A Genetic Algorithm Based Source Encoding Scheme for Distinguishing Incoming Signals in Large-scale Space-invariant Optical Networks

Hongki Sung, Yoonkeon Moon and Hagyu Lee

Abstract

Free-space optical interconnection networks can be classified into two types, *space variant* and *space invariant*, according to the degree of space variance. In terms of physical implementations, the degree of space variance can be interpreted as the degree of sharing beam steering optics among the nodes of a given network. This implies that all nodes in a totally space-invariant network can share a single beam steering optics to realize the given network topology, whereas, in a totally space variant network, each node requires a distinct beam steering optics. However, space invariant networks require mechanisms for distinguishing the origins of incoming signals detected at the node since several signals may arrive at the same time if the node degree of the network is greater than one. This paper presents a signal source encoding scheme for distinguishing incoming signals efficiently, in terms of the number of detectors at each node or the number of unique wavelengths. The proposed scheme is solved by developing a new parallel genetic algorithm called distributed asynchronous genetic algorithm (DAGA). Using the DAGA, we solved signal distinction schemes for various network sizes of several topologies such as the hypercube, the mesh, and the de Bruijn.

I. Introduction

Free-space optical interconnection networks can be classified into two types, *space variant* and *space invariant*, according to the degree of space variance[1]. The degree of space variance determines the network's complexity and regularity. A totally space variant network allows a completely arbitrary interconnection between nodes, whereas a totally space invariant network has a definite and regular structure with all the nodes having the same connection patterns. In terms of physical implementations, the degree of space variance can be interpreted as the degree of sharing beam steering optics among the nodes of a given network[2,3]. In other words, all nodes in a totally space-invariant network can share a single beam steering optics (considering multiple fanouts as one steering optics) to realize the given network topology, whereas, in a totally space variant network, each node requires a distinct beam steering optics. This is one of the reasons why space variant networks require complex optical implementations that often result in low interconnection density and high cost.

However, space invariant networks require mechanisms for distinguishing the origins of incoming beams (or signals) detected at the node since several signals may arrive at the same time if the node degree of the network is greater than one. We can distinguish the origins of the incoming signals if each signal has unique wavelength (wavelength division multiplexing) or each signal arrives at uniquely assigned timeslot (time division multiplexing).

For an N-node network, this would require up to N different wavelengths or N timeslots. Therefore, for a large-scale interconnection network, the requirement for the prohibitively large number of wavelengths makes such solutions impractical since the number of wavelengths from the currently available light sources is very limited. Similarly, the use of large number of timeslots would result in unacceptably large communication latency. It should be noted that the incoming signal distinction problem can be easily solved in space variant networks since we can have each incoming beam hit on a difference detector within the receiving node. This can be done easily in totally space variant networks because each incoming signal has its own

This research was supported by the Korean Science and Engineering Foundation(KOSEF) under project No. 961-0907-042-2.
Manuscript received July 14, 1995; accepted May 18, 1996.
H.K. Sung and Y.K. Moon are with Pacific Numerix Corp.
H.G. Lee is with Department of Computer Engineering, Hallym University.

beam steering optics. The incoming signal distinction by the use of a unique detector for each incoming signal is based on the concept of space division multiplexing. In a totally space-invariant network with N nodes, a trivial space division multiplexing scheme always exists if each node has N-1 detectors. In this paper, we present a method for reducing the required number of detectors per node for incoming signal distinction in a space-invariant network. The problem is first formalized in terms of graph theory. We find that the given problem is NP-complete so that we use a genetic algorithm (GA) to get a solution. GAs have recently received much attention as robust stochastic searching algorithms for various optimization problems[4,5,6]. This class of methods is based on the principles of natural selection and natural genetics that combine the notion of survival of the fittest, random and yet structured search, and parallel evaluation of nodes in the search space. Unfortunately, we are unable to get a solution using canonical GAs such as simple GAs[4] or parallelized GAs such as the Asynchronous GA[7]. So we develop a new parallel GA called Distributed Asynchronous GA (DAGA). We applied the proposed method using the DAGA to several network topologies with various network sizes. The topologies considered includes the hypercube[8], the mesh[9], and the binary de Bruijn networks[10,11] which show many advantages over other topologies in the design of massively parallel computers.

II. Motivation and Problem Statement

This section discusses what motivated us to develop a scheme which can efficiently distinguish incoming signals in a large-scale space-invariant optical interconnection networks, and then, formalizes the problem using graph theory.

1. Motivation

Let us consider an example network of a linear array with 4 nodes as shown in Fig. 1. Fig. 1.a shows the topology of the 4-node linear array and Fig. 1.b presents a side view of a totally space invariant optical implementation. A beam steering optics is responsible for generating two fanouts, one fanout to the right neighboring node and the other to the left neighboring node. We can see from the figure that all nodes have the same connection pattern and, consequently, can share a single beam steering optics to provide all the required optical links. The source position of each node is different from those of the others so that the incoming signals can be distinguished by the positions of the detectors. Similar straightforward spatial encoding of source positions at the nodes is used in the design of optical hypercube networks[3,12].

However, such straightforward encoding scheme requires too many unused detectors for large-scale networks. For an N-node network with node degree k, only k different signals may simultaneously arrive at a node since the node is connected to k nodes. Thus, in the space division multiplexing technique, only k+1 (k detectors and 1 source) pixels per node are utilized and the remaining N-k-1 pixels are wasted. This area waste is significant if $N >> k$. For example, the straightforward spatial encoding in a ten-dimensional hypercube network would result in (1024 - 11 = 1013) unused pixels per node, which is not acceptable.

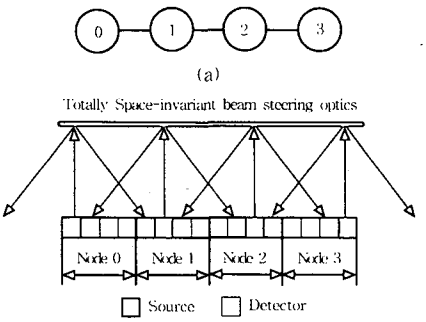


Fig. 1. An example network of a 4-node linear array. (a) the topology. (b) A totally space-invariant optical implementation. Each node is connected to neighboring nodes on both side except nodes on extreme ends.

2. Problem Statement

An efficient signal distinction scheme is to find the minimum number of detector pixels per node required for the incoming signal distinction in a given network. This problem can be re-stated if we represent a given network as a graph where a vertex and an edge represent a node and a link, respectively. Let V (E) represent the set of vertices (edges) in a graph, G.

Problem: Given a graph $G = (V,E)$, a positive integer $K \leq |V|$, which is a representation of a network, is G K-colorable? i.e., does there exist a function $f : V \rightarrow 0,1, \cdots, K\text{-}1$, for all i and j, such that $f(i) \neq f(i_1) \neq f(i_2) \cdots \neq f(i_j)$ where i_j represent all neighbors of node i.

The above problem is NP-complete since it can be reduced into the *Graph K-Colorability* problem which is known as NP-complete[13]. In general, NP-complete problems are very hard to solve unless the problem size is small enough. Using genetic algorithms, we solved the above modified colorability problem to derive efficient signal distinction schemes for the mesh, the hypercube, and the binary de Bruijn networks. Next section discusses in detail the genetic algorithm we used for the above problem and its implementation on a parallel computer.

III. A Genetic Algorithm for Optical Signal Distinction

In this section, we introduce fundamentals of genetic algorithms and their applications to deriving an efficient signal source encoding scheme for large-scale space-invariant optical interconnection networks.

1. General Description of Genetic Algorithms

Genetic algorithms (GAs) were developed to study the adaptive process of natural systems and to develop artificial systems that mimic the adaptive mechanism of natural systems[4,5,6,14,15,16,17]. GAs encode a potential solution to a specific problem on a simple chromosome-like data structure and apply such operators as selection, recombination (or crossover), and mutation to these structures in the hopes of getting closer to the solution. In a broader sense, a GA is any population-based model which uses selection and recombination operators to generate new sample points in a search space.

The first step in the implementation of GAs is to generate an initial population randomly in most cases. Each member of this population will be a binary string of length L which corresponds to the problem encoding. Each string is sometimes referred to as a *genotype* or a *chromosome*. On the execution of the GA, it starts with the *current population*. The initial population becomes the current population in the beginning or in the first generation. Selection is applied to the current population to create an intermediate population. Then recombination and mutation are applied to the intermediate population to create the *next population*. The process of going from the current population to the next population is called one generation in the execution of a GA. Now the next population becomes the current population for the next generation and repeats the above process until a member of the current population represents a solution to the given problem.

2. A Genetic Algorithm for the Proposed Signal Distinction Scheme

In Subsec. 3.1, we described a canonical GA referred to as a simple genetic algorithm[18]. Part of the biological metaphor used to motivate genetic search is that it is inherently parallel. In natural populations, thousands or even millions of individuals exist in parallel. This suggests a degree of parallelism that is directly proportional to the population size used in the genetic search. Many parallel genetic algorithms such as GENITOR II and Asynchronous Genetic Algorithms (AGA) have been proposed so far [5,7,14,19].

In the AGA[7], fitness evaluations are distributed to a number of processors and the whole population is maintained in a single processor. However, in the AGA, advantages of the parallel execution decrease as the number of processors increases mainly due to communication overheads among processors. Thus, the AGA is not good for massively parallel computers. This motivated us to develop a new parallel genetic algorithm called Distributed Asynchronous Genetic Algorithm (DAGA) for massively parallel computers. It should be noted that we were unable to solve the problem in Subsec. 2.2 using the canonical simple GA or the AGA within a reasonable amount of time as the problem size grows.

The DAGA solves the communication overheads problem in the AGA and also have additional features as follows. Since the performance of GAs depends on control parameters such as selection methods, population sizes, crossover probability, and mutation probability, the DAGA minimizes the number of control parameters. For example, the DAGA employs the random selection which reduces selection time compared to a selection method based on evaluation.

In what follows, we discuss an implementation of the DAGA in detail for the proposed optical signal distinction problem. We first discuss the string representation for the problem encoding. Control parameters such as the fitness function, the selection method, and the genetic operators are then discussed. Finally, we show a complete genetic algorithm.

1) String Representation and Initial Population

A node for m possible colors is represented by an m-bit string and the node's color is determined by the position of a 1 in the bit string. This implies that only a single bit can be set to 1 in the m-bit string since a node can take only one color. Thus, colors of all nodes in an N-node network can be represented by an Nm-bit string where the first m bits are for the first node's color, next m bits for the second node's color, etc.

The DAGA assumes the use of a parallel computer with multiple processors. Each processor runs a GA agent. Each GA agent holds only a single chromosome (an Nm-bit string which represents node colors of the network) and executes GA operations explained in the following subsections. For the initial population, each GA agent creates a chromosome randomly. However, the chromosome should be a *valid* string representation. By *valid* we mean that each group of m bits in the chromosome should have only a single bit set to 1.

2) Fitness and Selection

Fitness is defined as the number of nodes which have no color conflicts with neighboring (directly connected) nodes. A node with no color conflicts is the one which complies with coloring conditions discussed in Subsec. 2.2. Thus, for an N-node network, N is the maximum fitness or the *optimum*. It should be noted that *evaluation* in our DAGA implementation means the process of calculating the fitness

values for chromosomes.

In the DAGA, each GA agent randomly selects a mating partner from one of other GA agents. Unlike canonical GAs, the DAGA's crossover operation generates only a single offspring as will be discussed in detail in the following subsection.

3) Genetic Operators

Two genetic operators, crossover and mutation, are used to generate an offspring.

Crossover:

We use a two-point crossover operation, which is performed as follows. Let M_i and P_i be m-bit binary strings, where m is the number of colors that a node can take and $1 \leq i \leq N$ for an N-node network. Suppose that M_i and P_i represent colors of node i. Then, two chromosomes, I_m (*myself*) and I_p (*partner*), can be represented as follows:

$$I_m = M_1|M_2| \cdots |M_{N-1}|M_N$$

$$I_p = P_1|P_2| \cdots |P_{N-1}|P_N$$

where '|' denotes a string concatenation operation.

With the crossover probability p_{cross}, we choose two crossover points out of *N-1* possible points. Suppose that c_1 and c_2, where $1 \leq c_1 < N-1$ and $c_1 < c_2 \leq N - 1$, are chosen. If the length of the substring $M_{c1+1}|M_{c1+2}| \cdots |M_{c2}$ is less than that of the substring $M_1|M_2| \cdots |M_{c1}$ plus the length of the substring $M_{c2+1}| \cdots |M_N$, then an offspring I_o is constructed as

$$I_o = M_1| \cdots |M_{c1}|P_{c1+1}| \cdots |P_{c2}|M_{c2+1}| \cdots |M_N$$

Otherwise the offspring becomes

$$I_o = P_1| \cdots |P_{c1}|M_{c1+1}| \cdots |M_{c2}|P_{c2+1}| \cdots |P_N$$

The two-point crossover operation we employed in the DAGA is intended to generate an *offspring* who is always more similar to *myself*.

Mutation:

We adopted the concept of the next-point mutation[14], which determines the position of the next bit to be mutated with respect to the latest mutated bit position rather than by calculating mutations for every bit. This method reduces computation time compared to a normal mutation operation which scans all bits in the string and flips the bit with the given mutation probability.

In the DAGA implementation, with a mutation probability of p_{mutate}, a random number r in $[1, 1/p_{mutate}]$ is selected. Then we change the r-th node color of the offspring obtained by the crossover operation. This is only for the first mutation. If it's not for the first time, we mutate r-th away node from the previously mutated node. In case that r becomes greater than the number of nodes, we cannot find r-th away node in this chromosome. Mutation is then simply postponed till the next generation offspring is born. By virtually concatenating the current chromosome and its offspring, we determine r-th away node to do the mutation.

In the DAGA, a long string is virtually made along down the generations. Thus, we call this type of mutation as a *vertical* next-point mutation.

Assume that random numbers are chosen as r_1, r_2, r_3, $\cdots$. Then we mutate r_1-th, (r_1+r_2)-th, $(r_1+r_2+r_3)$-th node, $\cdots$ in the long virtual string. Once a node for mutation is determined, the node chooses a different color randomly. It should be noted that one offspring may have more than one mutation or none depending on the mutation probability p_{mutate} and the chromosome size.

4) A Complete Genetic Algorithm for Optical Signal Distinction

For an N-node network, a parallel computer with K processors is assumed. Each processor runs a GA agent as follows.

Algorithm for each GA agent:

Step 1 (Initialize): Each GA agent randomly generates a chromosome (say *myself*).
So the population size is equal to K, the number of the GA agents.

Step 2 (Evaluation of initial population): Each GA agent evaluates fitness of its own chromosome.

Step 3 (Selection and Crossover): Each GA agent randomly selects a partner from other GA agents, and does two-point crossover with the partner to generate an offspring,

Step 4 (Mutation): Each GA agent applies the vertical next-point mutation to the offspring.

Step 5 (Replacement): Each GA agent evaluates the offspring generated in Step 4 and replaces *myself* with the offspring only if the offspring's fitness is better than that of *myself* as well as that of the mating partner selected in Step 3.

Step 6 Each GA agent repeats Steps 3 through 5 until an optimum (fitness equals N) is found or the number of iterations reaches the given maximum value. When one of the two conditions is met, the GA agent signals to the others to stop running.

It should be noted that the above algorithm gives a coloring scheme with N colors for an N-node network when it find an optimum. If the algorithm is terminated by reaching the maximum iteration number, then we need to check whether at least one chromosome represents a coloring scheme without any color conflicts for all nodes. Then, such a scheme can be a solution even if it is not an optimum. As

will be shown in the following section for the simulation results, we hardly find an optimum.

3. Empirical Results

The DAGA discussed in the previous section was implemented and performed on the CM-5 parallel computer for the hypercube, the mesh, and the binary de Bruijn networks. The DAGA used the following parameters for the simulation.

- population size = 256. We used 256 processing elements in the CM-5.
- crossover probability = 0.7
- mutation probability = 0.005
- number of generations: shown in Table 1.

Table 1. Maximum number of generations for each GA run.

Network Size	Hypercube	Mesh	de Bruijn
16 nodes	40,000	40,000	30,000
64 nodes	80,000	80,000	80,000
256 nodes	2,000,000	800,000	1,000,000
1,024 nodes	16,000,000	5,000,000	7,000,000

Table 2 shows the minimum number of distinct colors or number of distinct pixels required using the DAGA for the mesh with wraparound connections, the hypercube, and the binary de Bruijn networks. A number in the paranthesis indicates the theoretical optimal number of pixels per node which is equivalent to the node degree (optimal number of detectors) plus 1 (one source). For example, for a 256-node hypercube (8-dimensional hypercube), the proposed scheme requires only 18 pixels for thespace division multiplexing technique.

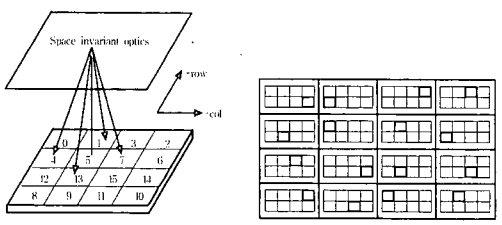
Table 2. Minimum number of pixels (a source and multiple detectors) per node required for a given size network. Note that node degree of the hypercube increases logarithmically with respect to the network size, whereas those of the mesh and the binary de Bruijn are constant. A number in the parenthesis indicates the theoretical optimal number of pixels. The mesh is assumed to have wraparound connections.

Network size	Hypercube	Mesh	de Bruijn
16 nodes	8 (5)	8 (5)	7 (5)
64 nodes	11 (7)	8 (5)	9 (5)
256 nodes	18 (9)	8 (5)	9 (5)
1,024 nodes	25 (11)	9 (5)	11 (5)

We compare the results out of the proposed signal distinction scheme with those out of the straightforward space division multiplexing schemes used in Ref. [3]. The straightforward space division multiplexing scheme would be as follows. For an N-node network, a node in the source plane is subdivided into N smaller pixels, and the address of the source node is encoded by the position of the pixel. A node in the detector plane is also subdivided into N pixels with a one-to-one correspondence between the subdivion of the source plane and the detector plane. In other words, each node mimics the encoding of the entire plane. Therefore, for an N-node network, regardless of the network topologies, the number of pixels per node required by the straightforward space division multiplexing scheme is equal to the number of nodes, N, in the given network. Table 3 shows how many pixels are saved by using the proposed signal distinction scheme.

Table 3. The number of pixels required for network topologies of given sizes. It shows that the proposed encoding scheme developed by the DAGA requires far less number of pixels compared to the straightforward scheme.

Network size	Number of pixels required			
	Proposed encoding scheme			Straightforward scheme
	Hypercube	Mesh	de Bruijn	Any topology
16 nodes	128	128	112	256
64 nodes	704	512	576	4,096
256 nodes	4,608	2,048	2,304	65,536
1,024 nodes	25,600	9,216	11,264	1,048,576



(a) (b)

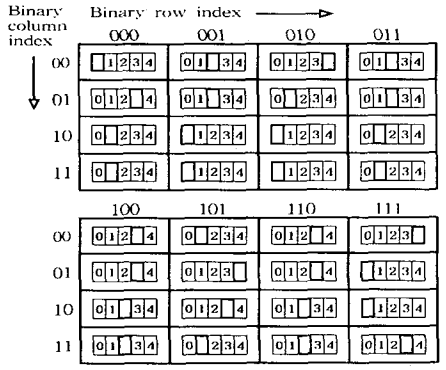
Fig. 2. An example for a hypercube network. (a) A 3D model of totally space-invariant beam steering optics for implementing optical hypercube networks. The beam steering optics generates 8 fanouts of which amounts of spatial shifts are $\pm 1col$, $\pm 1row$, $\pm 3col$, and $\pm 3row$, respectively. (b) Spatial encoding of source pixel positions in a 4-dimensional hypercube network with 16 nodes. Each node has 8 pixels; a source and 7 detectors. A dark pixel represents a source and the position of the source pixel is the spatial encoding of the corresponding node address as indicated by a number. White pixels represent detectors.

Fig. 2 demonstrates an example of the source position encoding, found by the DAGA for the four-dimensional hypercube network with 16 nodes. The beam steering optics for the totally space-invariant optical implementation of such

networks is based on Ref. [3]. A dark pixel and a white pixel represent a source and a detector, respectively. Note that the node address is spatially encoded by the position of the source pixel. The corresponding address is indicated by a number in the source pixel. For a four-dimensional hypercube network, the proposed scheme requires 8 pixels per node which is a 50% saving compared to 16 pixels per node required by the straight forwarding encoding scheme. Another example of the the source position encoding for five dimensional binary de Bruijn network is shown in Fig. 3, where beam steering optics is based on Ref. [20].

Node number (assigned spatial position)	Neighboring nodes	Node number (assigned spatial position)	Neighboring nodes
0 (pixel 0)	0 0 1 16	16 (pixel 1)	0 1 8 24
1 (pixel 2)	0 2 3 16	17 (pixel 0)	2 3 8 24
2 (pixel 4)	1 4 5 17	18 (pixel 0)	4 5 9 25
3 (pixel 2)	1 6 7 17	19 (pixel 1)	6 7 9 25
4 (pixel 3)	2 8 9 18	20 (pixel 2)	8 9 10 26
5 (pixel 1)	2 10 11 18	21 (pixel 3)	10 10 11 26
6 (pixel 3)	3 12 13 19	22 (pixel 2)	11 12 13 27
7 (pixel 4)	3 14 15 19	23 (pixel 0)	11 14 15 27
8 (pixel 3)	4 16 17 20	24 (pixel 1)	12 16 17 28
9 (pixel 2)	4 18 19 20	25 (pixel 0)	12 18 19 28
10 (pixel 1)	5 20 21 21	26 (pixel 0)	13 20 21 29
11 (pixel 2)	5 21 22 23	27 (pixel 1)	13 22 23 29
12 (pixel 3)	6 22 24 25	28 (pixel 2)	14 24 25 30
13 (pixel 4)	6 22 26 27	29 (pixel 1)	14 26 27 30
14 (pixel 3)	7 23 28 29	30 (pixel 2)	15 28 29 31
15 (pixel 0)	7 23 30 31	31 (pixel 3)	15 30 31 31

(a)



(b)

Fig. 3. An example for a binary de Bruijn network. (a) List of assigned pixel positions and neighboring node numbers in the five dimensional binary de Bruijn network. (b) Layout of the source/detector plane for the network.

IV. Concluding Remarks

Theoretically, a free space space-invariant design approach requires less number of beam steering optics and can provide higher interconnect density compared to the space-variant approach. However, in practice, the space-invariant design (1) utilizes given power less efficiently than the space-variant approach, and in addition, the space-invariant approach often (2) results in low resource utilization for distinguishing incoming signals; i.e., the approach may require many unused detectors when space division multiplexing technique is used for signal distinction.

In this paper, we have studied the second problem in the above that should be solved to make the space-invariant approach practical. We have presented an efficient scheme for distinguishing incoming signals. We first formalized the problem as a modified graph colorability problem and solved it using a genetic algorithm. Since we were unable to solve the problem simply using canonical genetic algorithms or existing parallelized genetic algorithms, we developed a new type of parallelized genetic algorithms called the Distributed Asynchronous Genetic Algorithm. The proposed scheme significantly reduces the number of the unused detectors, which enables the use of space division multiplexing technique in large-scale optical networks.

The proposed signal distinction scheme can also be used in wavelength division multiplexing and/or time division multiplexing techniques by considering the spatial positions as distinct wavelengths or timeslots, respectively. With wavelength or time division multiplexing techniques, each node has a single detector unlike the multiple detectors with the space division multiplexing technique. A receiving node distinguishes the origin of the incoming signal by identifying the wavelength of the signal or the timeslot the signal occupies.

References

[1] G. E. Lohman and K. H. Brenner, "Space-invariance in optical computing systems", *Optik*, vol. 89, pp. 123-134, 1992.

[2] A. Louri and H. Sung, "3d optical interconnects for high-speed interchip and interboard communications", *Computer*, vol. 27, pp. 27-37, Oct. 1994.

[3] A. Louri and H. Sung, "Efficient implementation methodology for three-dimensional space-invariant hypercube-based free-space optical interconnection networks", *Applied Optics*, vol. 32, pp. 7200-7209, Dec. 1993.

[4] J. H. Hollandm *Adaptation in Natural and Artificial Systems: an Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence.* Ann Arbor, MI: University of Michigan Press, 1975.

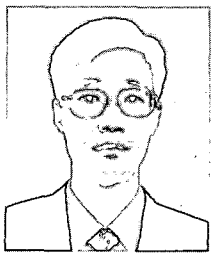
[5] D. Whitley and T. Starkweather, "Genitor ii : A distributed genetic algorithm", *Journal of Expert Theory and Artificial Intelligence*, vol. 2, pp. 189-214, Jan. 1994.

[6] M. Srinivas and L. M. Patnaik, "Genetic algorithms: A survey", *Computer*, vol. 27, pp. 17-26, June 1994.

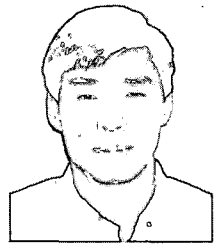
[7] B. P. Zeigler and J. Kim, "Asynchronous genetic algorithm on parallel computer", in *Proceeding of the 5th International Conference on Genetic Algorithms*, (Urbana-Champaign, IL), Univ. of Illinois Urbana-

Champaign, July 1993.
[8] J. P. Hayes and T. Mudge, "Hypercube supercomputers", *Proceedings of the IEEE*, vol. 77, pp. 829-1841, 1989.
[9] K. Hwang, *Advanced Computer Architecture : Parallelism, Scalability, Programmability*. New York, NY: McGraw-Hill, 1993.
[10] N. G. de Bruijn, "A combinational problem", *Koninklije Nedderlandse Academie van Wetenshappen Proc*, vol. Ser A49, pp. 758-764, 1946.
[11] J.-C. Bermond and C. Peyrat, "De bruijn and kautz networks: a competitor for the hypercube?", in *Proceedings of the First European Workshop on Hypercube and Distributed Computers*, (Rennes, France), pp. 279-293, Elsevier Science Publishers, Oct. 4-6 1989.
[12] Y. Sheng, "Space invariant multiple imaging for hypercube interconnections", *Applied Optics*, vol. 29, pp. 1101-1105, 1990.
[13] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York, NY: W. H. Freeman and Company, 1979.
[14] V. S. Gordon and D. Whitley, "Serial and parallel genetic algorithms as function optimization", in *Proceeding of the 5th International Conference on Genetic Algorithms*, (Urbana-Champaign, IL), pp. 177-183, Univ. of Illinois Urbana-Champaign, July 1993.
[15] Z. Miachalewicz, *Genetic Algorithm + Data Structure = Evolution Programming*. New York, New York: Springer-Verlag, 1992.
[16] T. Back and H. P. Schwefel, *An Overview of Evolutionary Algorithms for Parameter Optimization*, vol. 1, pp. 1-23. MIT Press, 1993.
[17] J. L. R. Filho and P. C. Treleaven, "Genetic-algorithms programming environments", *Computer*, vol. 27, pp. 28-43, June 1994.
[18] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*. Reading, MA: Addison-Wesley, 1989.
[19] S. Baluja, "Structure and performance of fine-grain parallelism in genetic search", in *Proceeding of the 5th International Conference on Genetic Algorithms*, (Urbana-Champaign, IL), pp. 155-162, Univ. of Illinois Urbana-Champaign, July 1993.
[20] A. Louri and H. Sung, "Optical binary de bruijn networks for massively parallel computing: Design methodology and feasibility study", *Applied Optics*, vol. 35, pp. 6714-6722, Oct. 1995.

Hongki Sung received the B.S. degree in Electronics Engineering from the Seoul National University, Korea, in 1984, the M.S. degree in Computer Engineering from the University of Southwestern Louisiana, U.S.A., in 1990, and the Ph.D. degree in Electrical Engineering from the University of Arizona, U.S.A., in 1994. He was a Member of Technical Staff at ETRI, Korea, from 1984 to 1989, and a professor at the Hallym University, Korea, from 1995 to 1997. He is now a Senior Staff Engineer at the Pacific Numerix Corp, Scottsdale, Arizona, U.S.A. His current research interests include simulation and modeling for VLSI characterization, genetic algorithms, and optical interconnects.



Hagyu Lee received the B.S., M.S., and Ph.D degrees in Computer Engineering from Seoul National University, Korea, in 1987, 1989, and 1994, respectively. He is currently a professor in the Department of Computer Engineering, Hallym University, Korea. His research interests include natural language processing, information retrieval, and Korean information processing.



Yoonkeon Moon received the B.S. and M.S. degrees in Control and Instrumentation Engineering from Seoul National University, Korea, in 1982 and 1984, respectively. From 1984 to 1989, he worked at the Central Research Lab. of Goldstar Industrial Systems Company in Korea as a research engineer. He received the Ph.D. degree in Electrical Engineering from the University of Arizona, U.S.A., in 1996. He is currently working at Pacific Numerix Corp. as a senior staff engineer. His research interests include discrete event system modeling/simulation, genetic algorithms, watershed modeling/simulation, and electronic design automation.