Copyright ⓒ 2008 KSII

Effect of Changing the Basis in Genetic Algorithms Using Binary Encoding

Yong-Hyuk Kim[1] and Yourim Yoon[2], *Non-members*
[1] Department of Computer Science and Engineering, Kwangwoon University
447-1, Wolgye-dong, Nowon-gu, Seoul, 139-701, Korea
[e-mail: yhdfly@kw.ac.kr]
[2] School of Computer Science and Engineering, Seoul National University
Sillim-dong, Gwanak-gu, Seoul, 151-744, Korea
[e-mail: yryoon@soar.snu.ac.kr]
*Corresponding author: Yourim Yoon

Received June 17, 2008; revised July 30, 2008; accepted August 4, 2008; published August 25, 2008

Abstract

We examine the performance of genetic algorithms using binary encoding, with respect to a change of basis. Changing the basis can result in a change in the linkage structure inherent in the fitness function. We test three simple functions with differing linkage strengths and analyze the results. Based on an empirical analysis, we show that a better basis results in a smoother fitness landscape, hence genetic algorithms based on the new encoding method provide better performance.

Keywords: Genetic algorithms, binary encoding, change of basis, coordinate-change, non-singular binary matrix

The authors would like to thank the anonymous referees for their helpful comments and suggestions that improved the quality of this paper. The present research has been conducted through the Research Grant of Kwangwoon University in 2008. The ICT at Seoul National University provided the research facilities for this study. A preliminary version of this paper appeared in Proceedings of the Genetic and Evolutionary Computation Conference, ACM SIGEVO, pp. 1117-1118, 2008.

DOI: 10.3837/tiis.2008.04.002

1. Introduction

Traditional approaches dealing with binary encoding generally use the inherent standard basis. When we consider a basis other than the standard one, the linkage structure between basis elements and the ruggedness of the problem space can be completely different from the original ones. Via a change of basis, a complex problem can be transformed into a simple one, and vice versa. In this paper, we provide an idea for changing the basis in binary encoding. We also investigate and analyze its effect on genetic algorithms.

The paper is organized as follows. In Section 2 we explain the method of changing the basis in the vector space $Z_2^n = (\{0,1\}^n, \oplus)$.[1] We introduce previous work related to the concept of changing the basis in Section 3. In Section 4 we introduce test functions with different linkage structures, and new bases for the functions, and in Section 5 we analyze the experimental results. We provide discussion including future work in Section 6.

2. Change of Basis

2.1 Binary Matrix

A matrix $\mathbf{A}$ is defined as *binary* if $\mathbf{A} \in M_{n\times n}(Z_2)$. Binary matrices have been widely used to deal with the adjacency of a graph [1][2][3]. In particular, Anderson and Feil [1] transformed the *light bulb puzzle* into the problem of solving a linear system $\mathbf{Ax} = \mathbf{b}$ from its graph structure, where $\mathbf{A} \in M_{n\times n}(Z_2)$ and $\mathbf{x}, \mathbf{b} \in Z_2^n$. Then, the solution is obtained by computing the inverse of $\mathbf{A}$, i.e., $\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$. Binary matrices are also useful in dealing with the *cut/cycle* subspace of a graph, which is a vector space over Z_2 [2]. They can also be used to represent a change of basis of a vector space over Z_2 in combinatorial problems. In this paper, we present this new application of binary matrices.

2.2 Change of Basis in Z_2^n

A *basis* for a vector space of dimension n is a sequence of n vectors, where every vector in the space can be uniquely expressed as a linear combination of basis vectors. Since it is often desirable to utilize more than one basis for a vector space, it is important to understand the means of easily transforming coordinate-wise representations of vectors and linear transformations, with respect to one basis, to their equivalent representations, with respect to another basis. Such a transformation is called a *change of basis*. The following theorem is easily derived from the basic theory of linear algebra [7].

Theorem 1. *Let* B_1 *and* B_2 *be two bases for* Z_2^n. *Then there exists a nonsingular matrix* $\mathbf{T} \in M_{n\times n}(Z_2)$ *such that for every* $\mathbf{v} \in Z_2^n$, $\mathbf{T}(\mathbf{v})_{B_1} = (\mathbf{v})_{B_2}$, *where* $(\mathbf{v})_B$ *is the representation of* $\mathbf{v}$ *with respect to the basis* B.

[1] $\oplus$ is the exclusive-or (XOR) operator. $(a_1, a_2, ..., a_n) \oplus (b_1, b_2, ..., b_n) = (a_1 \oplus b_1, a_2 \oplus b_2, ..., a_n \oplus b_n)$

The matrix $\mathbf{T}$ in Theorem 1 is called *coordinate-change* matrix from the basis $\mathcal{B}_1$ to $\mathcal{B}_2$. The standard basis $\mathcal{B}_s$ for Z_2^n is $\{e_1, e_2, ..., e_n\}$, where $e_j = (0,...,0,1,0,...,0)$ is the element of Z_2^n with 1 only in the j-th place and 0s in all other positions. Given a nonsingular matrix $\mathbf{T} \in M_{n\times n}(Z_2)$, the matrix $\mathbf{T}$ can be regarded as a coordinate-change matrix from the standard basis to another basis $\mathcal{B}_\mathbf{T}$ related to $\mathbf{T}$. For every $\mathbf{v} \in Z_2^n$, $\mathbf{T}(\mathbf{v})_{\mathcal{B}_s} = (\mathbf{v})_{\mathcal{B}_\mathbf{T}}$ and then $\mathcal{B}_\mathbf{T} = \{\mathbf{T}e_1, \mathbf{T}e_2, \ldots, \mathbf{T}e_n\}$. Hence, the problem of finding a new basis is equivalent to that of finding a proper nonsingular binary matrix.

3. Prior Work

In this section, we provide a brief literature survey on the basis change in real-coded genetic algorithms as well as binary-coded ones.

Chryssomalakos and Stephens [5] theoretically dealt with the bases of function space on Z_2^n. However, in this paper, we investigate the bases of more a fundamental space, i.e., those of the vector space Z_2^n.

Gene reordering [4][8][10] can be considered as a special case of a change of basis. If $\mathbf{T}$ is just a permutation matrix, a change of basis means a reordering of gene positions in encoding. The concept of changing the basis is much more general than that of gene reordering. Coordinate changes based on *eigenspace* and *orthogonalization* have been studied [12][13]. But, they focused on real-code representation. These results are not applicable to problems using binary encoding, because of the following proposition.

Proposition 1. *Let* $\mathbf{T}$ *be a nonsingular binary matrix. Then, if* $\mathbf{T}$ *is not the identity matrix* $\mathbf{I}$, *the eigenspace of* $\mathbf{T}$ *does not span* Z_2^n.

Proof. The spectrum of $\mathbf{T}$ is a subset of Z_2. Since $\mathbf{T}$ is nonsingular, zero cannot be an eigenvalue of $\mathbf{T}$. If $\mathbf{T}$ has an eigenvalue, it must be one. Suppose that the eigenspace of $\mathbf{T}$ spans Z_2^n. Then there exist linearly-independent n eigenvectors with eigenvalues of one. Let $\mathbf{A}$ be the matrix in columns of which has such n eigenvectors. Then $\mathbf{TA} = \mathbf{A}$. Since $\mathbf{A}$ is nonsingular, $\mathbf{T} = \mathbf{I}$. This is a contradiction.

To the best of the author's knowledge, the only non-trivial change of basis in a vector space over Z_2 is from [2][6]. In *cut space*, which is a subspace of Z_2^n, there are other bases corresponding to spanning trees of the given connected graph, besides the standard basis.

4. Test Functions

We tested n-dimensional functions of binary variables. These functions are simplified adaptations of Kauffman's NK landscapes [9]. Kauffman defined a function with n bits, in which each bit's fitness contribution depends on its k neighbors. NK landscapes thus have "tunable ruggedness" and are often used to test genetic algorithms. We defined three functions, where $k = 1, 2, n-2$, respectively. Each variable's fitness contribution depends on its next k variables. Three test functions are provided in the following.

$$\mathrm{F}_2(x) = \sum_{i=1}^{n-1} x_i \oplus x_{i+1} + x_n \oplus x_1,$$

$$\mathrm{F}_3(x) = \sum_{i=1}^{n-2} x_i \oplus x_{i+1} \oplus x_{i+2} + x_{n-1} \oplus x_n \oplus x_1 + x_n \oplus x_1 \oplus x_2, \text{ and}$$

$$\mathrm{F}_{n-1}(x) = \sum_{i=1}^{n} x_i \oplus \cdots \oplus x_{i-1} \oplus x_{i+1} \oplus \cdots \oplus x_n,$$

where each $x_i \in Z_2$.

Suppose that $\mathbf{T}$ is a nonsingular binary matrix. Let $\mathbf{x} = (x_1\ x_2 \cdots x_n)^T$ be $(\mathbf{v})_{\mathbb{B}_S}$ and $\mathbf{y} = (y_1\ y_2 \cdots y_n)^T$ be $(\mathbf{v})_{\mathbb{B}_T}$, where $\mathbf{v} \in Z_2^n$. Then $\mathbf{Tx} = \mathbf{y}$. Hence, $\mathrm{F}_2(\mathbf{x}) = \mathrm{F}_2(\mathbf{T}^{-1}\mathbf{y})$, $\mathrm{F}_3(\mathbf{x}) = \mathrm{F}_3(\mathbf{T}^{-1}\mathbf{y})$, and $\mathrm{F}_{n-1}(\mathbf{x}) = \mathrm{F}_{n-1}(\mathbf{T}^{-1}\mathbf{y})$.

For the function F_2, we select the coordinate-change matrix $\mathbf{T} = (t_{i,j})$ such that $t_{i,i} = t_{i,i+1} = 1$ and $t_{i,j} = 0$ in all other positions. For the function F_3, we select the matrix $\mathbf{T} = (t_{i,j})$ such that $t_{i,i} = t_{i,i+1} = t_{i,i+2} = 1$ and $t_{i,j} = 0$ in all other positions. We can easily check that these two $\mathbf{T}$s are nonsingular for all n. For the function F_{n-1}, we select the matrix $\mathbf{T} = (t_{i,j})$ such that $t_{i,i} = 0$ and $t_{i,j} = 1$ in all other positions. Then $\mathbf{T}$ is nonsingular when n is even, because $\mathbf{T}^2 = (\mathbf{1} - \mathbf{I})^2 = \mathbf{O} - 2 \cdot \mathbf{1} + \mathbf{I} = \mathbf{I}$ where $\mathbf{1}$ is an $n \times n$ binary matrix in which all elements are 1 and $\mathbf{O}$ is an $n \times n$ binary matrix in which all elements are 0. Given a coordinate-change matrix $\mathbf{T}$, its inverse $\mathbf{T}^{-1}$ can be efficiently computed by the Gaussian elimination method. Hence we can express the functions F_2, F_3, and F_{n-1} on the new basis $\mathbb{B}_T$ as follows:

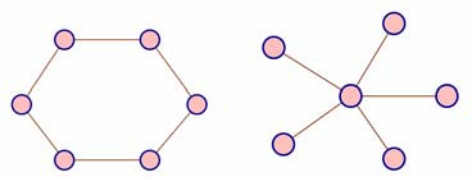
$$\mathbf{F}_2(\mathbf{T}^{-1}\mathbf{y}) = \sum_{i=1}^{n-1} y_i + \bigoplus_{i=1}^{n-1} y_i,$$

$$\mathbf{F}_3(\mathbf{T}^{-1}\mathbf{y}) = \sum_{i=1}^{n-2} y_i + \bigoplus_{i \in S_1} y_i + \bigoplus_{i \in S_2} y_i, \text{ and}$$

$$\mathbf{F}_{n-1}(\mathbf{T}^{-1}\mathbf{y}) = \sum_{i=1}^{n} y_i,$$

where each $y_i \in Z_2$ and S_ks are proper subsets of $\{1, 2, \ldots, n\}$.

Here, we examine the test functions before and after changing the basis. For convenience, we set the number of binary variables n to 6. **Fig. 1**, **2**, and **3** show the linkage structures among terms of the functions F_2, F_3, and F_{n-1}, respectively. They also show coordinate-change matrices ($\mathbf{T}$ and its inverse $\mathbf{T}^{-1}$). Here, each node denotes one term in the corresponding function. Each edge between two nodes denotes that the values of the two terms corresponding to the two nodes are dependent on their common variables. If the resultant graph has a complex topology, the corresponding function seems to be difficult to solve. Conversely, a simple sparse topology suggests that it is an easy problem. It is clear that the topologies obtained after changing the basis are simpler than those obtained beforehand. In particular, in the case of F_{n-1}, a completely-linked topology can be changed into a completely discrete one, just by changing the basis. In the next section, we show that such a change of topology intuitively affects the performance of a genetic algorithm.

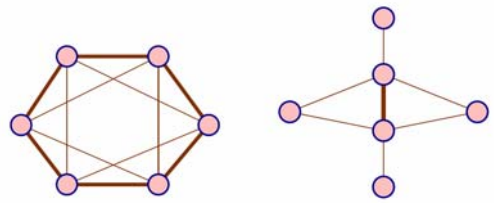


(a) $F_2(\mathbf{x})$ (b) $F_2(\mathbf{T}^{-1}\mathbf{y})$

$$\begin{pmatrix} 1&1&0&0&0&0\\ 0&1&1&0&0&0\\ 0&0&1&1&0&0\\ 0&0&0&1&1&0\\ 0&0&0&0&1&1\\ 0&0&0&0&0&1 \end{pmatrix} \qquad \begin{pmatrix} 1&1&1&1&1&1\\ 0&1&1&1&1&1\\ 0&0&1&1&1&1\\ 0&0&0&1&1&1\\ 0&0&0&0&1&1\\ 0&0&0&0&0&1 \end{pmatrix}$$

(c) $\mathbf{T}$ (d) $\mathbf{T}^{-1}$

Fig. 1. Linkage among terms of F_2 ($n = 6$)

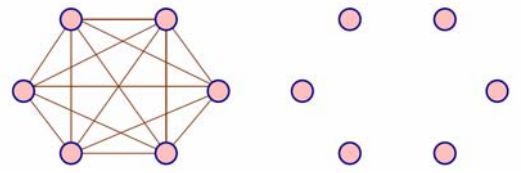


(a) $F_3(\mathbf{x})$ (b) $F_3(\mathbf{T}^{-1}\mathbf{y})$

$$\begin{pmatrix} 1&1&1&0&0&0\\ 0&1&1&1&0&0\\ 0&0&1&1&1&0\\ 0&0&0&1&1&1\\ 0&0&0&0&1&1\\ 0&0&0&0&0&1 \end{pmatrix} \qquad \begin{pmatrix} 1&1&0&1&1&0\\ 0&1&1&0&1&1\\ 0&0&1&1&0&1\\ 0&0&0&1&1&0\\ 0&0&0&0&1&1\\ 0&0&0&0&0&1 \end{pmatrix}$$

(c) $\mathbf{T}$ (d) $\mathbf{T}^{-1}$

Fig. 2. Linkage among terms of F_3 ($n = 6$)
(Thick edges indicate stronger linkage than thin ones)



(a) $F_{n-1}(\mathbf{x})$ (b) $F_{n-1}(\mathbf{T}^{-1}\mathbf{y})$

$$\begin{pmatrix} 0&1&1&1&1&1\\ 1&0&1&1&1&1\\ 1&1&0&1&1&1\\ 1&1&1&0&1&1\\ 1&1&1&1&0&1\\ 1&1&1&1&1&0 \end{pmatrix} \qquad \begin{pmatrix} 0&1&1&1&1&1\\ 1&0&1&1&1&1\\ 1&1&0&1&1&1\\ 1&1&1&0&1&1\\ 1&1&1&1&0&1\\ 1&1&1&1&1&0 \end{pmatrix}$$

(c) $\mathbf{T}$ (d) $\mathbf{T}^{-1}$

Fig. 3. Linkage among terms of F_{n-1} ($n = 6$)

4. Empirical Results

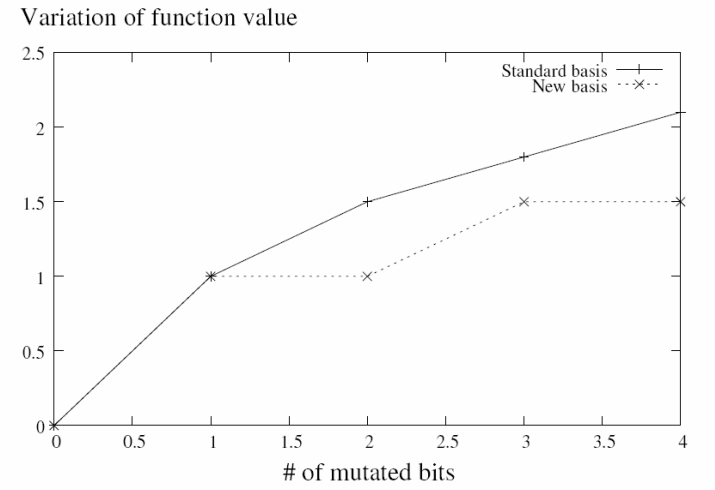
First, we investigated the ruggedness of test functions. When we mutated one bit in the encoding using the standard basis, more than one bit could be changed in the encoding of the new basis $\mathbb{B}_{\mathbf{T}}$. Thus, the neighborhood structure could differ greatly depending on the selected basis. For six test cases with 40 variables, i.e., $n = 40$, we examined the function values with respect to mutations of k arbitrary bits from a randomly generated solution, where $k = 1, 2, 3, 4$. **Table 1** shows the results, and **Fig. 4** depicts their average values. The figures were obtained from 10,000 trials. After changing the basis, we were able to ensure that all the test functions were smoother, i.e., the fitness landscape of $\mathrm{F}_{\#}(\mathbf{T}^{-1}\mathbf{y})$ is smoother than that of $\mathrm{F}_{\#}(\mathbf{x})$. In particular, it is clear that the new basis of F_{n-1} completely removes deceptiveness inherent in the function using the standard basis.

Table 1. Variation of Function Value

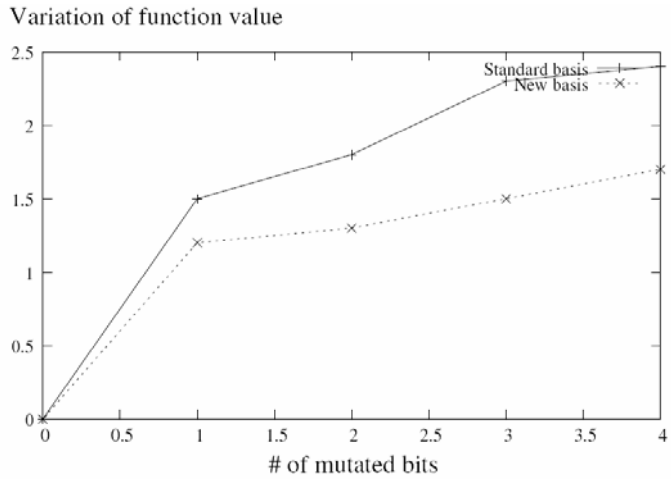
Test function	Perturbation strength	Maximum	Average	Standard deviation
$\mathbf{F}_2(\mathbf{x})$	1 bit	2	1.0	1.0
	2 bits	4	1.5	1.3
	3 bits	6	1.8	1.6
	4 bits	8	2.1	1.7
$\mathbf{F}_2(\mathbf{T}^{-1}\mathbf{y})$	1 bit	2	1.0	1.0
	2 bits	2	1.0	1.0
	3 bits	4	1.5	1.3
	4 bits	4	1.5	1.3
$\mathbf{F}_3(\mathbf{x})$	1 bit	3	1.5	0.9
	2 bits	6	1.9	1.6
	3 bits	9	2.3	1.6
	4 bits	12	2.4	2.0
$\mathbf{F}_3(\mathbf{T}^{-1}\mathbf{y})$	1 bit	3	1.2	1.0
	2 bits	4	1.3	1.0
	3 bits	5	1.5	1.2
	4 bits	6	1.7	1.3
$\mathbf{F}_{n-1}(\mathbf{x})$	1 bits	25	5.0	3.7
	2 bits	2	1.0	1.0
	3 bits	25	4.9	3.6
	4 bits	4	1.5	1.3
$\mathbf{F}_{n-1}(\mathbf{T}^{-1}\mathbf{y})$	1 bit	1	1.0	0.0
	2 bits	2	1.0	1.0
	3 bits	3	1.5	0.9
	4 bits	4	1.5	1.3

* Results from 10,000 trials.

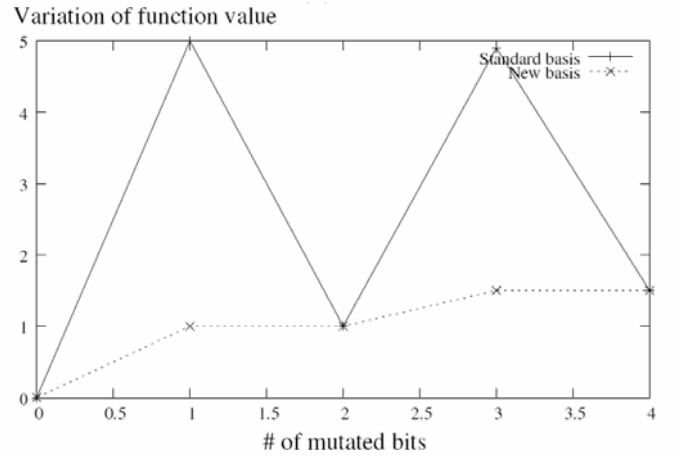
For the tests, we used the genetic algorithm of [11] with typical parameters. The genetic algorithm maximizes the test functions provided in Section 4. All genetic parameters except for the evaluation function are the same, and these are provided in **Table 2**.



(a) F_2



(b) F_3



(c) F_{n-1}

Fig. 4. Variation of function value

Table 3 shows the results for the same genetic framework. The optimal value of each test function is 40. The table figures were obtained from 100 trials. In every case, after changing the standard basis into another good one, the performance was significantly improved. In particular, the test on the function F_{n-1} shows that the new basis is much better than the standard basis. This is a natural result of the fact that the new basis completely removes the strong linkage structure inherent in the function represented by the standard basis. Also, it is clear that the performance of each function is proportional to the smoothness provided in **Table 1**.

Table 2. Input Size and Genetic Parameters

Type	Value
Number of variables (n)	40
Population size	50
Selection	Tournament selection with size 2
Crossover rate Crossover type	1.0 One-point crossover
Mutation rate Mutation type	0.05 Gene-wise mutation
Replacement proportion	0.5
Maximum number of generations	500

Table 3. Results ($n = 40$)

Test function	Best	Average (Average %-gap)	Std (Std %-gap)	CPU †	Optimal value
$\mathbf{F}_2(\mathbf{x})$	**40**	37.08 (7.30)	1.43 (3.58)	0.55	40
$\mathbf{F}_2(\mathbf{T}^{-1}\mathbf{y})$	**40**	**39.08 (2.30)**	1.12 (2.79)	0.54	40
$\mathbf{F}_3(\mathbf{x})$	39	36.68 (8.30)	1.45 (3.62)	0.55	40
$\mathbf{F}_3(\mathbf{T}^{-1}\mathbf{y})$	**40**	**38.60 (3.50)**	1.04 (2.61)	0.55	40
$\mathbf{F}_{n-1}(\mathbf{x})$	37	31.45 (21.38)	2.19 (5.49)	0.89	40
$\mathbf{F}_{n-1}(\mathbf{T}^{-1}\mathbf{y})$	**40**	**39.54 (1.15)**	0.61 (1.53)	0.53	40

* Results from 100 trials.
The value of %-gap is computed as follows: 100 ×(optimum − output)/optimum.
†Average CPU seconds on Intel® Xeon™ CPU 2.40GHz.

5. Discussion

In this paper, we provided the simple but important idea of changing the basis in binary representation. The paper was based on the fact that a problem encoded by a binary vector has multiple representations according to the selected basis. We also tested some functions, which demonstrated the advantages of changing the basis. To the best of the author's knowledge, this is the first paper establishing the importance of changing the basis in binary encoding. However, this paper does not provide any indications about which basis should be selected to ensure that the search space is smooth. More studies about the mechanism for finding a good basis, i.e., a good coordinate-change matrix are needed. To this end, it would be good to begin with an investigation of the space of nonsingular binary matrices, i.e., general linear groups

over Z_2^n. Some problems may be barely transformed by a change of basis. Research about which problems are transformed by changing the basis and the degree to which the problems are transformed are topics for future study. In the future, this approach can be extended into the case of k-ary encodings.

References

[1] M. Anderson and T. Feil, "Turning lights out with linear algebra," *Mathematics Magazine*, 71(4):300-303, 1998.

[2] N. Biggs, *Algebraic Graph Theory*, Cambridge University Press, second edition, 1994.

[3] R. A. Brualdi and H. J. Ryser, *Combinatorial Matrix Theory*, Cambridge University Press, 1991.

[4] T. N. Bui and B. R. Moon, "Genetic algorithm and graph partitioning," *IEEE Transactions on Computers*, 45(7):841-855, 1996.

[5] C. Chryssomalakos and C. R. Stephens, "What basis for genetic dynamics?" In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1018-1029, 2004.

[6] R. Diestel, *Graph Theory*, Springer-Verlag, Heidelberg, third edition, 2005. Graduate Texts in Mathematics, Volume 173.

[7] S. H. Friedberg, A. J. Insel, and L. E. Spence, *Linear Algebra*, Prentice-Hall International, Inc., third edition, 1997.

[8] I. Hwang, Y.-H. Kim, and B.-R. Moon, "Multi-attractor gene reordering for graph bisection," In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1209-1215, 2006.

[9] S. Kauffman, "Adaptation on rugged fitness landscapes," *Lectures in the Science of Complexity*, pages 527-618, 1989.

[10] Y.-H. Kim, Y.-K. Kwon, and B.-R. Moon, "Problem-independent schema synthesis for genetic algorithms," In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1112-1122, 2003.

[11] K. Sastry, "Single and multiobjective genetic algorithm toolbox in C++," Technical Report 2007016, IlliGAL, University of Illinois at Urbana-Champaign, June 2007.

[12] D. Whitley, M. Lunacek, and J. Knight, "Ruffled by ridges: How evolutionary algorithms can fail," In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 294-306, 2004.

[13] D. Wyatt and H. Lipson, "Finding building blocks through eigenstructure adaptation," In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1518-1529, 2003.

Yong-Hyuk Kim received the B.S. degree in computer science and the M.S. and Ph.D. degrees in computer science and engineering from Seoul National University (SNU), Seoul, Korea, in 1999, 2001, and 2005, respectively. From March 2005 to February 2007, he was a Postdoctoral Scholar in SNU and also a research staff member at the Inter-University Semiconductor Research Center in SNU. Since March 2007, he has been an assistant professor at Department of Computer Science and Engineering in Kwangwoon University, Seoul, Korea. His research interests include algorithm design/analysis, discrete mathematics, optimization theory, combinatorial optimization, evolutionary computation, operations research, and data/web mining. Dr. Kim has served as a Committee Member of GECCO 2005-2006 and a reviewer for journals (IS, TKDE, TPDS, TC, TSE) of IEEE Computer Society since 2003.

Yourim Yoon received the B.S. degree in computer engineering from Seoul National University (SNU), Seoul, Korea, in 2003. She is currently a Ph.D. candidate and working toward the Ph.D. degree at School of Computer Science and Engineering (CSE) in SNU. She is also a chief of the Optimization Laboratory at School of CSE in SNU. Her research interests include optimization theory, combinatorial optimization, evolutionary computation, discrete mathematics, and operations research. She served as a reviewer for BIC-TA 2007.