DETECTION OF THE PERMUTATION SYMMETRY IN PATTERN SETS

DONG JI-YANG AND ZHANG JUN-YING

Received 19 February 2006; Accepted 1 May 2006

Symmetry is a powerful tool to reduce the freedom degrees of a system. But the applicability of the symmetry tool strongly depends on the ability to calculate the symmetries of the system. There exists an interesting algorithmic problem to search for the symmetry of a high-dimensional system. In this paper, a genetic algorithm-based permutation symmetry detection approach is proposed for pattern set. Firstly, the permutation symmetry distance (PSD) is defined to measure the similarity of a pattern set before and after being transformed by a permutation operator. Secondly, the permutation symmetry detection problem is converted into an optimization problem by taking the PSD as a fitness function. Lastly, a genetic algorithm-based approach is designed for the symmetry detection problem. Computer simulation results are also given for five pattern sets of different dimensionality, which show the efficiency and speediness of the proposed detection approach, especially in high-dimensional cases.

1. Introduction

Symmetry, as a powerful tool, may reveal the intrinsic relations inherent in the objects or phenomena that seem to be uncorrelated and is used widely to almost all areas of the natural science [2, 6, 9]. The study of symmetry begins in the early stage of the neural networks research [21], and recently more and more researchers devoted themselves to this field. For example, Baldi demonstrated in his paper [1] that the global properties of an individual network could be found from symmetry considerations of the invariance group of the specific pattern set stored by some learning rules. Reimann showed in his paper [17] that the symmetry of the network structure is already determined by the symmetry of the set of test sequences, indicating that learning a set of elements applied is concerned with finding invariant relations inherent in this set. He also showed how to design the artificial autoassociative neural networks using group theoretical methods [18]. All this research reveals the importance and usefulness of the symmetry method in the study of neural networks.

The applicability of the symmetry method, however, strongly depends on the potential ability of calculating the symmetry group of the system efficiently. In general, the symmetry group is usually calculated by the generators of the symmetry group of the given system itself. When the generators are unknown, one has to calculate the symmetry group by the method of exhaustive search, which is impractical in fact for a high-dimensional case because of the tremendous search space. For example, the order of the symmetric group S_n grows with $n!$, that is, there are $n!$ potential solutions in an n-dimensional system [18]. There is, to our knowledge, no report of using an efficient algorithm to detection the permutation symmetry of a high-dimensional system. Almost all of the up-to-the-date symmetry studies in neural networks done at the low-dimensional cases because of the symmetry calculation problem. There still exists an interesting algorithmic problem when the system is sufficiently large.

We define in this paper a permutation symmetry distance (PSD) to measure the similarity of a pattern set before and after being transformed by a permutation operator, then convert the permutation symmetry detection problem into an optimization problem. There are many stochastic approaches for the optimization problem [11–13, 15, 19, 20], the genetic algorithm is a simple and effective one. We do not want to study the algorithm itself here. Our intent is to test the feasibility of such a global random search algorithm in the permutation symmetry detection problem. Instead of developing a new search algorithm, we use the standard genetic algorithms (GAs) to search for the symmetric permutation operators of a given pattern set in this paper. A permutation operator is encoded by a set of consecutive integers (chromosome). The PSD is taken as the fitness function. The three chosen genetic operators (crossover, mutation, selection) are related to those in some works of GAs for the traveling salesman problem, which have the same encoding method to our work. Results are presented for five different dimensional binary pattern sets. When compared with exhaustive search, genetic algorithms show greater efficiency, especially in high-dimensional cases.

The organization of this paper is as follows. Section 2 gives some definitions used in this paper and presents the detailed definition of permutation symmetry measure. In Section 3, we outline the search algorithm and the corresponding operators. In Section 4, we present the computer simulation results. Conclusions and discussions are given in Section 5.

2. Measure of the permutation symmetry

2.1. Permutation symmetry. We begin our discussion with some important definitions [3].

Definition 2.1. A set $V = \{v_1, v_2, \dots, v_n\}$ consists of n elements. Then *a permutation operator* (or called *permutation* for short) of V is a bijection $\sigma : V \to V$, which reindexes the set. When omitting the symbol v, the set V can be understood as a set of consecutive integers (or an order set), that is, $(1, 2, \dots, n)$, and the permutation can be written as $\sigma = \begin{pmatrix} 1 & 2 & \cdots & n \\ i_1 & i_2 & \cdots & i_n \end{pmatrix}$, $i_j \in \{1, 2, \dots, n\}$, where it is understood that σ maps 1 to i_1, 2 to i_2, and so forth.

Definition 2.2. The set of all permutation operators on the set $\{1,2,\dots,n\}$ is denoted by S_n, which is called the symmetric group of degree n.

The order of S_n is $n!$ as is easily seen using the "two-row" method to write a permutation operator. For example, let $V = \{1,2,3\}$, there are six permutation operators for this set, namely,

$$\begin{pmatrix}1&2&3\\1&2&3\end{pmatrix},\begin{pmatrix}1&2&3\\1&3&2\end{pmatrix},\begin{pmatrix}1&2&3\\2&1&3\end{pmatrix},\begin{pmatrix}1&2&3\\2&3&1\end{pmatrix},\begin{pmatrix}1&2&3\\3&2&1\end{pmatrix},\begin{pmatrix}1&2&3\\3&1&2\end{pmatrix}. \tag{2.1}$$

The permutation operator $\left(\begin{smallmatrix}1&2&\cdots&n\\1&2&\cdots&n\end{smallmatrix}\right)$ is called the identity element and is denoted as e.

Definition 2.3. Let $x = (x_i)$, $i = 1,2,\dots,n$, be an n-dimensional pattern vector. The action of a permutation operator $(s \in S_n)$ on x is defined as $(s \cdot x)_i = x_{s(i)}$, which reindexes the components of the pattern vector.

For example, let $x = (a,b,c,d)$, that is, $x_1 = a$, $x_2 = b$, $x_3 = c$, $x_4 = d$. Let the permutation operator be $s = \left(\begin{smallmatrix}1&2&3&4\\3&2&4&1\end{smallmatrix}\right)$, then $s \cdot x = (x_{s(1)},x_{s(2)},x_{s(3)},x_{s(4)}) = (x_3,x_2,x_4,x_1) = (c,b,d,a)$.

Definition 2.4. Let $X = \{x^1,x^2,\dots,x^m\}$ be an m-pattern set. The action of permutation operator (s) on the set X is defined as $s \cdot X = \{s \cdot x^1, s \cdot x^2,\dots,s \cdot x^m\}$. If $s \cdot X = X$, the permutation operator (s) is a symmetric permutation operator of the set X.

The set of all symmetric permutation operators of the set X is denoted by S_X. S_X is a group and is called the permutation symmetry group of X, or permutation symmetry of X in brief (readers desiring a more exactly definition can refer to the book [3]).

For example, let $X = \{(2,1,2),(2,2,1),(2,1,1)\}$. The permutation symmetry of X is $S_X = \{e, \left(\begin{smallmatrix}1&2&3\\1&3&2\end{smallmatrix}\right)\}$, which consists of 2 symmetric permutation operators.

Definition 2.5. The action of a permutation operator (s) on a matrix $w = (w_{ij})$ is defined as $(s \cdot w)_{ij} = x_{s(i)s(j)}$, for all $s \in S_n$, which reindexes the rows and columns of the matrix simultaneously. If $s \cdot w = w$, the permutation operator s is a symmetric permutation operator of w. The set of all symmetric permutation operators of w is called the permutation symmetry group of w, denoted as S_w.

Property 2.6. Let $G = \{g_i\}$ be a symmetry group, and $g_i, g_j \in G$. If $g_k = g_i \cdot g_j$, then $g_k \in G$.

For example, the pattern set $V = \{(1,1,0,0),(1,0,0,1),(0,0,1,1),(0,1,1,0)\}$, which has 8 symmetric permutation operators. Assume that we know two of the symmetric permutation operators, $s_1 = \left(\begin{smallmatrix}1&2&3&4\\1&4&3&2\end{smallmatrix}\right)$ and $s_2 = \left(\begin{smallmatrix}1&2&3&4\\2&1&4&3\end{smallmatrix}\right)$. Then the other symmetric permutation operators can be obtained by

$$\begin{aligned}
s_3 = s_1 \cdot s_2 &= \begin{pmatrix}1&2&3&4\\4&1&2&3\end{pmatrix}, & s_4 = s_2 \cdot s_1 &= \begin{pmatrix}1&2&3&4\\2&3&4&1\end{pmatrix},\\
s_5 = s_3 \cdot s_1 &= \begin{pmatrix}1&2&3&4\\4&3&2&1\end{pmatrix}, & s_6 = s_4 \cdot s_2 &= \begin{pmatrix}1&2&3&4\\3&2&1&4\end{pmatrix},\\
s_7 = s_5 \cdot s_2 &= \begin{pmatrix}1&2&3&4\\3&4&1&2\end{pmatrix}, & s_8 = s_1 \cdot s_1 &= \begin{pmatrix}1&2&3&4\\1&2&3&4\end{pmatrix}.
\end{aligned} \tag{2.2}$$

Then $S = \{s_1,s_2,s_3,s_4,s_5,s_6,s_7,s_8\}$ is the permutation symmetry group of the pattern set V.

Property 2.7. Let $G = \{g_i\}$ be a group, and $g_i \in G$, then $g_k = g_i^{-1} \in G$.
For example, if $s_1 = \left(\begin{smallmatrix} 1 & 2 & 3 & 4 \\ 1 & 4 & 3 & 2 \end{smallmatrix}\right) \in G$, then $s_1^{-1} = \left(\begin{smallmatrix} 1 & 2 & 3 & 4 \\ 1 & 4 & 3 & 2 \end{smallmatrix}\right) \in G$.

2.2. Measure for permutation symmetry. Symmetry is treated as a binary feature in the exact mathematical definition (an object is either symmetric or nonsymmetric). However, the exact definition of symmetry is inadequate to describe and quantify neither the symmetries found in the natural world nor those found in the visual world. For example, we say that the equilateral triangle has "more" symmetry than the isosceles triangle. Thus, although symmetry is usually considered a binary feature, we view symmetry as a continuous feature where intermediate values of symmetry denote some intermediate "amount" of symmetry. This concept of continuous symmetry is in accord with our perception of symmetry as can be seen in [22].

Zabrodsky et al. [22] present a concept "symmetry distance" to measure the symmetry in an image. But they treat the shape of an image as a point sequence, that is, an order-dependent points set. So the symmetry distance is limited to measure the symmetries of a sequence. There are no reports of the continuous measure of the permutation symmetry of a pattern set which is order independent to its patterns.

In this section, we aim to define a measure for permutation symmetry in a pattern set.

Let a set $V = \{v^k \in \mathbb{R}^n,\ k = 1, 2, \ldots, m\}$ contain m patterns, and let each pattern be an n-dimensional vector $v^k = (v_1^k, v_2^k, \ldots, v_n^k)$, then the out-product matrix of V is denoted as W,

$$W_{ij} = \sum_{k=1}^{m} v_i^k \cdot v_j^k. \tag{2.3}$$

$W = (W_{ij})$ is an $n \times n$ real matrix.

For example, the pattern set $V = \{(2,3,2), (2,2,3)\}$ has 2 patterns. According to formula (2.3), the out-product matrix of V would be

$$W = \begin{pmatrix} 4 & 6 & 4 \\ 6 & 9 & 6 \\ 4 & 6 & 4 \end{pmatrix} + \begin{pmatrix} 4 & 4 & 6 \\ 4 & 4 & 6 \\ 6 & 6 & 9 \end{pmatrix} = \begin{pmatrix} 8 & 10 & 10 \\ 10 & 13 & 12 \\ 10 & 12 & 13 \end{pmatrix}. \tag{2.4}$$

We have the following theorem.

Theorem 2.8. *Let S_V be the permutation symmetry group of the pattern set $V = \{v_i^k\}$, and let W be the out-product matrix of V, then all permutation operators in S_V are also the symmetric permutation operators of W, that is,*

$$s \cdot W = W, \quad \forall s \in S_V. \tag{2.5}$$

Proof. We prove this theorem by contradiction.

Assume there is a permutation operator s of S_V, and s is not the symmetric permutation operator of W, that is,

$$s \cdot W \neq W. \tag{2.6}$$

Assume $s \cdot v^1 = v^{i_1}$, $s \cdot v^2 = v^{i_2}, \dots, s \cdot v^m = v^{i_m}$, because $s \in S_V$, then $\{v^{i_1}, v^{i_2}, \dots, v^{i_m}\} = \{v^1, v^2, \dots, v^m\}$.

We can rewrite (2.3) as

$$W = \sum_{k=1}^{m} (v^k)^t v^k, \tag{2.7}$$

where $(v^k)^t$ is the transposed vector of v^k.

So

$$\begin{aligned} s \cdot W &= \{(s \cdot v^1)^t \cdot (s \cdot v^1) + (s \cdot v^2)^t \cdot (s \cdot v^2) + \cdots + (s \cdot v^m)^t \cdot (s \cdot v^m)\} \\ &= \{(v^{i_1})^t \cdot v^{i_1} + (v^{i_2})^t \cdot v^{i_2} + \cdots + (v^{i_m})^t \cdot v^{i_m}\} \\ &= \{(v^1)^t \cdot v^1 + (v^2)^t \cdot v^2 + \cdots + (v^m)^t \cdot v^m\} = W, \end{aligned} \tag{2.8}$$

that is, the permutation operator s is a symmetric permutation operator of W. It is opposite.

So all permutation operators in S_V are also the symmetric permutation operators of W. □

Theorem 2.8 implies $S_V \subseteq S_W$. In fact, $S_V = S_W$ can easily be satisfied by modifying the out-product matrix as

$$W_{ij} = \sum_{k=1}^{m} (v_i^k + \alpha)(v_j^k + \alpha), \tag{2.9}$$

where α is a real parameter. In general, $S_V = S_W$ when $v_i^k + \alpha \geq 0$ for any i and k.

So we can search for the permutation symmetry of a given pattern set by its out-product matrix.

Let W be the out-product matrix of a given pattern set V, and let W^s be the permutated out-product matrix by the permutation operator s. We define the *permutation symmetry distance* (PSD) of the permutation operator s on the given pattern set V as a quantifier of the difference between W and W^s, that is,

$$\text{PSD}(s) = \sum_{i=1}^{n} \sum_{j=1}^{n} (W_{ij}^s - W_{ij})^2. \tag{2.10}$$

We can normalize by scaling the matrix element W_{ij} so that the maximum PSD of a permutation operator is a given constant, for example, 1 or n. Thus the PSD value is limited in range, where $\text{PSD}(s) = 0$ for perfectly symmetric permutation operator s. What we do in this work is to search out for all the permutation operators, whose PSD are equal to 0, of a given pattern set.

For example, the out-product matrix of the pattern set $V = \{(2,3,2),(2,2,3)\}$ is $W = \begin{pmatrix} 8 & 10 & 10 \\ 10 & 13 & 12 \\ 10 & 12 & 13 \end{pmatrix}$. Let the permutation operator be $s = \begin{pmatrix} 1 & 2 & 3 \\ 1 & 3 & 2 \end{pmatrix}$, then $W^s = \begin{pmatrix} 8 & 10 & 10 \\ 10 & 13 & 12 \\ 10 & 12 & 13 \end{pmatrix}$. Because $W^s = W$, $\text{PSD}(s) = \sum_{i=1}^{n} \sum_{j=1}^{n} (W_{ij}^s - W_{ij})^2 = 0$. So $s = \begin{pmatrix} 1 & 2 & 3 \\ 1 & 3 & 2 \end{pmatrix}$ is a symmetric permutation operator of the set V.

In the next section, we describe the genetic algorithm for searching for the symmetric permutation operators of a pattern set.

3. Genetic algorithm-based approach

Genetic algorithms are search algorithms based on the mechanics of natural selection and natural genetics and are used to search for large, nonlinear search spaces where expert knowledge is lacking or difficult to encode and where traditional optimization techniques fall short [8]. The basic principles of GAs were first designed by Holland [10]. They work with a population of individual strings (chromosomes), each representing a possible solution to a given problem. Each chromosome is assigned a fitness value according to the result of the fitness function. Highly fit chromosomes are given more opportunities to reproduce and the offspring share features taken from their parents.

The permutation symmetry detection problem gives all the symmetric permutation operators, whose PSD are equal to 0, of a given pattern set. We know that a pattern set often has more than one symmetric permutation operator. That is, permutation symmetry detection is a multioptimization problem. We aim to use the genetic algorithm to find as many different symmetric permutation operators as possible in a single run. So we modify the standard genetic algorithm [23] as follows.

(1) *Initialization*: a starting population with N individuals is (randomly) generated.
(2) *Evaluation:* every individual of the initial population is evaluated.
(3) *Recombination*: relatively "fit" individuals are selected for recombination. Then a new generation with N parents and N children is created using crossover and mutation.
(4) *Evaluation*: these new individuals are evaluated.
(5) *Save the symmetric permutation operators:* save the individuals whose fitness values are 0, then replace them with individuals randomly generated.
(6) *Selection*: choose the best N individuals to propagate to the next generation using the CHC selection [7].
(7) *Termination check*: if a given amount of time (a number of generations) has elapsed, the algorithm stops. Otherwise, it goes back to step (3) and continues.

We present the algorithm operators in detail as follows.

3.1. Representation of the chromosomes.

3.1. Representation of the chromosomes. Coding all possible solutions into different chromosomes is a key problem of genetic algorithms. The chromosome often is a binary string in traditional coding, but the binary coding scheme is not suitable for our problem.

A permutation operator can be written in the form of "two rows," for example, $\begin{pmatrix} 1 & 2 & \cdots & n \\ i_1 & i_2 & \cdots & i_n \end{pmatrix}$, when omitting the top line, the permutation operator is written as $[i_1 \quad i_2 \quad \cdots \quad i_n]$, where it is understood that the permutation operator maps 1 to i_1, 2 to i_2, and so forth. For example, the chromosome $[3 \quad 4 \quad 1 \quad 2]$ stands for the permutation operator $\begin{pmatrix} 1 & 2 & 3 & 4 \\ 3 & 4 & 1 & 2 \end{pmatrix}$. This coding method is used widely in GAs for TSP [7, 8, 14]. For an n-dimensional pattern set, we initialize the population by randomly placing 1 to n into n length chromosomes and guaranteeing that each number appears exactly once. Thus chromosomes stand for legal permutation operator.

3.2. Crossover. Because the chromosomes must be a sequence of 1 to n, the traditional single-point crossover is not suitable. Partially mapped crossover (PMX) which is a good crossover operator for this kind of chromosome is proposed by Glodberg and Lingle [8]. It works as follows.

First two crossover points are selected at random, for example,

(i) Parent 1: [1 2 **3 4 5 6** 7 8 9],

(ii) Parent 2: [2 5 **1 9 6 7** 3 4 8].

They define a *matching section* (in shadow and bold).

The corresponding elements of the matching section define an interchange mapping, for example,

$$\{3 \Longleftrightarrow 1, 4 \Longleftrightarrow 9, 5 \Longleftrightarrow 6, 6 \Longleftrightarrow 7\}, \tag{3.1}$$

which is applied pointwise to the parents to get the offspring, the other positions are filled with "#,"

(i) Offspring 1: [# # **1 9 6 7** # # #],

(ii) Offspring 2: [# # **3 4 5 6** # # #].

The elements duplicated with the mapped matching section define *the conflicting genes*.

Copy the remains elements except for the conflicting genes to the original positions from the corresponding parents,

(i) Offspring 1: [# 2 **1 9 6 7** # 8 #],

(ii) Offspring 2: [2 # **3 4 5 6** # # 8].

Collect the conflicting genes from the parents in a list,

(i) Conflicting genes in offspring 1: [**1 9 7**],

(ii) Conflicting genes in offspring 2: [**5 3 4**].

Then, the conflicting genes define another interchange mapping, for example,

$$\{1 \Longleftrightarrow 5, 9 \Longleftrightarrow 3, 7 \Longleftrightarrow 4\}. \tag{3.2}$$

Fill in the "#" positions one by one according to the mapping,

(i) Offspring 1: [5 2 **1 9 6 7** 4 8 3],

(ii) Offspring 2: [2 1 **3 4 5 6** 9 7 1].

The PMX operator can preserve efficiently the similarity between a parent and its offspring. In this way, the better permutation operator sections in parents may be inherited and combined for the offspring, which would speed the search process.

3.3. Mutation. For the same reason that we do not use the traditional crossover operator, we cannot use the traditional mutation operator. Instead of using the traditional mutation operator, we randomly select two genes in one chromosome and *swap* them. Thus, we still have legal permutation operator after swap mutation. For example,

(i) Parent: [1 2 3 **4** 5 6 7 8 **9**],

(ii) Offspring: [1 2 3 **9** 5 6 7 8 **4**].

3.4. Selection. When using traditional roulette wheel selection, the best individual has the highest probability of survival but does not necessarily survive. We use CHC selection to guarantee that the best individual will always survive in the next generation [7]. In CHC selection, if the population size is N, we generate N children by using roulette wheel selection, then combine the N parents with the N children, sort these $2N$ individuals according to their fitness value, and choose the best N individuals to propagate to the next generation. To prevent convergence to a local optimum, we save the top 10% individuals and reinitialize the rest of the population randomly if the population has converged [14].

4. Experimental results

In order to test the effectiveness of the genetic algorithms described above, we performed experiments on data with different dimensions and with different parameter values on a DELL computer (Optiplex G1, C400/128M SDRAM/Windows 2000 Server). The program is coded in Visual C++ 6.0. Here we present some experimental results (Table 4.1–Table 4.6) of some pattern set, in which the number of the symmetric permutation operators is known in advance, that is, the test sets have D_n-symmetry which has $2n$ (n is the dimension of the pattern) symmetric permutation operators [4, 5]. To restrict length, we just list three test pattern sets ($n = 8,9,10$) as follows.

For $n = 8$, the test pattern set is

$$\begin{gathered}\{(1,1,1,1,0,0,0,0),(1,1,1,0,0,0,0,1),(1,1,0,0,0,0,1,1),(1,0,0,0,0,1,1,1),\\(0,0,0,0,1,1,1,1),(0,0,0,1,1,1,1,0),(0,0,1,1,1,1,0,0),(0,1,1,1,1,0,0,0)\}.\end{gathered}\tag{4.1}$$

For $n = 9$, the test pattern set is

$$\begin{gathered}\{(1,1,1,1,1,0,0,0,0),(1,1,1,1,0,0,0,0,1),(1,1,1,0,0,0,0,1,1),\\(1,1,0,0,0,0,1,1,1),(1,0,0,0,0,1,1,1,1),(0,0,0,0,1,1,1,1,1),\\(0,0,0,1,1,1,1,1,0),(0,0,1,1,1,1,1,0,0),(0,1,1,1,1,1,0,0,0)\}.\end{gathered}\tag{4.2}$$

For $n = 10$, the test pattern set is

$$\begin{gathered}\{(1,1,1,1,1,0,0,0,0,0),(1,1,1,1,0,0,0,0,0,1),\\(1,1,1,0,0,0,0,0,1,1),(1,1,0,0,0,0,0,1,1,1),\\(1,0,0,0,0,0,1,1,1,1),(0,0,0,0,0,1,1,1,1,1),\\(0,0,0,0,1,1,1,1,1,0),(0,0,0,1,1,1,1,1,0,0),\\(0,0,1,1,1,1,1,0,0,0),(0,1,1,1,1,1,0,0,0,0)\}.\end{gathered}\tag{4.3}$$

In Tables 4.1–4.6, "m" is the number of symmetric permutation operators found by the program, "*time*" is the corresponding search time in seconds, "u" is the population size, "*maxgen*" is the maximum iteration generation. Table 4.1 shows the results of exhaustive search for 6 different pattern sets. Tables 4.2–4.6 show the results of our approach with different parameter values for 5 different pattern sets, respectively. The crossover and mutation rates are 0.7 and 0.01, respectively, in all experiments.

Table 4.1. Search results of 6 different dimensional pattern sets by exhaustive search.

Dimension	8	9	10	11	12	13
m	16	18	20	22	24	26
time	≈ 0.3	4	49	605	8379	107932

Table 4.2. Search results of 5 different parameter values for an 8-dimensional set by our approach.

u	20	20	20	30	100
maxgen	200	1000	500	1000	1000
m	8	13	11	13	16
time	< 1	3	2	5	13

Table 4.3. Search results of 5 different parameter values for a 10-dimensional set by our approach.

u	20	20	50	50	20
maxgen	1000	2000	1000	2000	4000
m	15	18	16	18	18
time	3	7	10	20	15

Table 4.4. Search results of 5 different parameter values for a 12-dimensional set by our approach.

u	50	50	80	100	100
maxgen	1000	2000	1000	1000	1200
m	15	18	16	19	21
time	12	25	20	25	30

5. Conclusions and discussions

From the simulation results shown in Tables 4.1–4.6, we can see the following.

(1) All of the symmetric permutation operators of a pattern set can be found by exhaustive search in theory, but it is so time consuming that it becomes impossible in fact for a high-dimensional case. This is seen from Table 4.1. The program requires more than 12-times search time to search for the symmetric permutation operators when the dimension of pattern set increases one. The search time is about 1.2 days for $n = 13$. At this rate, the program has to spend at least 5.5 years to find out all the symmetric permutation operators of a 16-dimensional pattern set. It is obvious that exhaustive search can only be used in small-dimensional case in practice.

(2) Our approach is not superior to exhaustive search in small-dimensional cases, that is, $n < 10$. However, when the dimension of the test pattern set is bigger than 10, our approach is faster than exhaustive search. For example, $n = 12$ in Table 4.4, genetic algorithm can find out 21 symmetric permutation operators in 30 seconds, but the exhaustive search has to spend 8379 seconds to find out all the 24 symmetric permutation operators

Table 4.5. Search results of 5 different parameter values for a 14-dimensional set by our approach.

u	100	100	100	80	70
maxgen	1000	2000	4000	2000	3000
m	14	19	23	17	23
time	33	66	133	53	70

Table 4.6. Search results of 5 different parameter values for a 16-dimensional set by our approach.

u	100	200	50	300	100
maxgen	2000	500	2000	1000	4000
m	16	8	9	10	23
time	84	43	41	134	169

(see Table 4.1). The most important advantage of our approach is that the search time increases slower than the exhaustive search as the dimensionality of the pattern set increases. For example, when the dimension $n = 16$ (see Table 4.6), the program can find out the majority of symmetric permutation operators in 169 seconds, it is less than 3 times of that for $n = 14$.

Of course, there is a shortcoming in our approach, for example, there are not any criteria to know whether all of the symmetric permutation operators have been found, so the search results are often incomplete. However, the permutation symmetry of a pattern set is a group, so we can apply Properties 2.6 and 2.7 to obtain more or even all the symmetric permutation operators.

(3) The total computational time needed is the product of single-measure time and the evaluation times needed in the global search algorithm. There are many other global optimization algorithms which may be better than the traditional genetic algorithms described in this paper, for example, immune algorithms [12] and quantum algorithms [20]. This paper illustrates with the traditional genetic algorithms how global optimization algorithms can be applied to search for the permutation symmetry of a pattern set.

(4) Moreover, as to effectiveness, our approach is relatively similar to the genetic algorithm for TSP, which is well documented, because both problems have the same genetic operations and representation. The dimension of pattern set corresponds to the city number in TSP, which can be several thousands [16].

(5) There are many algorithms to solve the optimization problem, for example, simulated annealing [19] and evolutionary strategies [15]. In recent years, more and more efficient algorithms have been presented, such as immune algorithms [12] and quantum algorithms [20]. Many of those algorithms are proven to be more effective than the standard genetic algorithms in all kinds of optimization problems. It is believed that it would be more effective when such algorithms are utilized instead of the standard genetic algorithm used in this paper for the permutation symmetry detection problem.

In this paper, we define a measure of permutation symmetry that transforms the symmetry detection problem to an optimization problem, which is the main contribution of this paper, and show how the genetic algorithms can be applied to detect the permutation symmetry of a given pattern set, which overcomes the computing complexity of permutation operators search and makes it possible to study the high-dimensional system with the symmetry tool, for example, designing of artificial neural networks.

Acknowledgment

This work is supported by the National Science Fund of China under Grant no. 60574039.

References

[1] P. Baldi, *Symmetries and learning in neural network models*, Physical Review Letters **59** (1987), no. 17, 1976–1978.

[2] D. M. Bishop, *Group Theory and Chemistry*, Clarendon Press, Oxford, 1973.

[3] J. Q. Chen, *Group Representation Theory for Physicists*, World Scientific, New Jersey, 1989.

[4] J.-Y. Dong, S. Xu, and B. Wu, *Symmetry in classification scheme of DHNN with even classification*, Journal of Xiamen University (Natural Science) **39** (2000), no. 3, 329–335.

[5] ______, *The symmetry of the structure of DHNN with even classification*, Journal of Xiamen University (Natural Science) **39** (2000), no. 1, 46–51.

[6] J. P. Elliott and P. G. Dawber, *Symmetry in Physics*, Macmillan Press, London, 1979.

[7] L. J. Eshelman, *The CHC adaptive search algorithm: how to have safe search when engaging in nontraditional genetic recombination*, Foundations of Genetic Algorithms-1 (G. J. E. Rawlins, ed.), Morgan Kaufmann, California, 1991, pp. 265–283.

[8] D. Goldberg and R. Lingle, *Alleles, loci and the traveling salesman problem*, Proceedings of the 1st International Conference on Genetic Algorithms, Lawrence Erlbaum Associates, New Jersey, 1985, pp. 154–159.

[9] B. Gruber and I. Iachello, *Symmetries in Science*, Plenum Press, New York, 1988.

[10] J. H. Holland, *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Michigan, 1975.

[11] A. Homaifar, S. Guan, and G. E. Liepins, *A new approach on the traveling salesman problem by genetic algorithms*, Proceedings of the 5th International Conference on Genetic Algorithms, Morgan Kaufmann, California, 1993, pp. 460–466.

[12] L. Jiao and L. Wang, *A novel genetic algorithm based on immunity*, IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans **30** (2000), no. 5, 552–561.

[13] P. Jog, J. Y. Suh, and D. Van Gucht, *The effects of population size, heuristic crossover and local improvement on a genetic algorithm for the traveling salesman problem*, Proceedings of 3rd International Conference on Genetic Algorithms (J. D. Schaffer, ed.), Morgan Kaufmann, California, 1989, pp. 110–115.

[14] S. J. Louis and G. Li, *Genetic algorithms with memory for traveling salesman problems*, Proceedings of the 1st International Conference on Genetic Algorithms (ICGA '85), Pennsylvania, July 1985, pp. 160–168.

[15] H. Muhlenbein, M. Gorges-Schleuter, and O. Kramer, *Evolution algorithms in combinatorial optimization*, Parallel Computing **7** (1988), no. 1, 65–85.

[16] Y. Nagata and S. Kobayashi, *Edge assembly crossover: a high-power genetic algorithms for the traveling salesman problem*, Proceedings of the 7th International Conference on Genetic Algorithms (ICGA '97) (T. Back, ed.), Morgan Kaufmann, California, 1997, pp. 450–457.

[17] S. Reimann, *Symmetry and network structure*, Neural Processing Letters **6** (1997), no. 1-2, 1–10.

[18] ______, *On the design of artificial auto-associative neuronal networks*, Neural Network **11** (1998), no. 4, 611–621.

[19] P. Ruján, *Searching for optimal configurations by simulated tunneling*, Zeitschrift für Physik. B. Condensed Matter **73** (1988), no. 3, 391–416.

[20] D. R. Simon, *On the power of quantum computation*, Proceedings of 35th Annual Symposium on Foundations of Computer Science (Santa Fe, NM, 1994), IEEE Comput. Soc. Press, California, 1994, pp. 116–123.

[21] C. Tianren, *Group theory method in artificial neural networks designing*, Discovery of Nature **27** (1989), no. 1, 21–29.

[22] H. Zabrodsky, S. Peleg, and D. Avnir, *Symmetry as a continuous feature*, IEEE Transactions on Pattern Analysis and Machine Intelligence **17** (1995), no. 12, 1154–1166.

[23] S. Zickenheiner, T. Nicolaou, A. Bleck, and K. Waldschmidt, *Genetic processing of the traveling salesman problem with the associative architecture AM*^{3}, Proceedings of the 20th EUROMICRO Conference on System Architecture and Integration (EUROMICRO '94), Liverpool, September 1994, pp. 111–116.

Dong Ji-Yang: Department of Physics, Xiamen University, Xiamen, Fujian 361005; National Key Lab for Radar Signal Processing, Xidian University, Xi'an, Shaanxi 710071, China
E-mail address: jydong@xmu.edu.cn

Zhang Jun-Ying: School of Computer Science and Technology, Xidian University, Xi'an, Shaanxi 710071; National Key Lab for Radar Signal Processing, Xidian University, Xi'an, Shaanxi 710071, China
E-mail address: jyzhang@xidian.edu.cn