Convergence analysis of the new hybrid genetic algorithm for job the shop scheduling problem

Nguyen Huu Mui
Department of Information Technology
Hanoi University of Education
Hanoi, Vietnam
muinh@hnue.edu.vn

Vu Dinh Hoa
Department of Information Technology
Hanoi University of Education
Hanoi, Vietnam
hoavudinh@gmail.com

Luc Tri Tuyen
Institute of Information Technology
Vietnam Academy of Science and Technology
Hanoi, Vietnam
tuyenlt@ioit.ac.vn

Abstract - In our recent paper [9], we proposed a new hybrid genetic algorithm (NHGA) for the job shop scheduling problems. The method of encoding we used is Natural coding instead of traditional binary coding. This manner of coding has a lot of advantages but its convergence is still an open issue for years. This paper analyzes the convergence property of the NHGA by applying properties of Markov chains. Based on the Markov chain analysis of genetic algorithm, we find out the proposed method leads to the convergence to the global optimum in the case of Natural Coding.

Keywords - genetic algorithm; convergence; Markov; jobshop.

I. INTRODUCTION

Genetic algorithms (GA) simulate self-adapting evolutionary processes of biological populations of nature in order to optimize the objective functions. These algorithms were developed by John Holland [5], characterized by searching strategies based on the population and genetic operations such as selection, mutation and crossover. Nakano and Yamada [11] were among the first people who applied the canonical GA to represent each solution as a binary series for job shop scheduling problems (JSP). They then proposed some methods of combining GA with various searching techniques and also gained many significant achievements in conquering the JSP. Ulder and others [10] firstly proposed a searching method as local genetic method. This method was the combination of GA with the local searching technique of the JSP. More recently, many researchers such as Lee Hui Peng at al [6], F. Guerriero [3], Rui Zhang at al [12], Yamada [14], etc., proposed various hybrid methods combining GA with many other searching techniques to address this complicated problem. Until now, however, there are not any method which addresses completely this problem especially in the case of many machines and many jobs.

Based on studies of the JSP and solutions applying to this problem, researchers recognized that the method which combines GA with other techniques to solve the JSP is a potential way to research. In our recent publication, we presented some new hybrid genetic algorithms for the JSP. For more detail about these algorithms, readers can refer to [9]. In this article, we only outline the basic points of the algorithms as the first part of presenting the main issue that is still open for years, that is a proof of the convergence of an upgraded hybrid genetic algorithm for the Job Shop scheduling problem.

II. THE NEW HYBRID GENETIC ALGORITHM FOR JOB SHOP SCHEDULING PROBLEM

A. Description of the problem (*)

I) The problem

The problem of the Job Shop scheduling is described as below:

Given $n$ jobs $\{J_i\}_{j=1, \ldots, n}$ which are processed by the set of $m$ machines $\{M_j\}_{j=1, \ldots, m}$ satisfying the constrains:

1. At a specific time, each machine is only able to process one job.

2. Each machine $M_j$ can process any job $J_i$, the part of work of $J_i$ processed by $M_j$ is defined by section $O_{ij}$. For each of the jobs, $J_i$, the order of processing $O_{ij}$ on the machines $\{M_j : j = 1, \ldots, m\}$ is given and called industrial serial.

3. The time mark of beginning and finishing the section $O_{ij}$ are denoted by $s_{ij}$ and $c_{ij}$ respectively.

4. The duration for processing of section $O_{ij}$ is denoted by $p_{ij}$.

5. The total of time for finishing all jobs is called the make-span and denoted by $C_{max} = \max \{c_{ij}\}$.

For example, a $3 \times 3$ JSP is given in Table 1. The given data includes the industrial serial of jobs and processing time for each job related to each machine.

<table>
<thead>
<tr>
<th>Jobs</th>
<th>Machines (processing time)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1 (3)</td>
</tr>
<tr>
<td></td>
<td>2 (3)</td>
</tr>
<tr>
<td>2</td>
<td>1 (2)</td>
</tr>
<tr>
<td></td>
<td>3 (3)</td>
</tr>
<tr>
<td>3</td>
<td>2 (3)</td>
</tr>
<tr>
<td></td>
<td>1 (2)</td>
</tr>
<tr>
<td></td>
<td>3 (1)</td>
</tr>
</tbody>
</table>

---

Supported by Hanoi University of Education, Vietnam.
Table 1 can be replaced by a matrix of industrial serials \( \{T_i\} \) and a matrix of processing time \( \{p_{ik}\} \) as follow:

\[
\begin{bmatrix}
1 & 2 & 3 \\
1 & 3 & 2 \\
2 & 1 & 3
\end{bmatrix}
\begin{bmatrix}
3 & 3 & 3 \\
2 & 3 & 4 \\
3 & 2 & 1
\end{bmatrix}
\]

2) **Encoding a solution due to Natural numbers**

For a Job Shop Scheduling Problem with many jobs and many machines, the way of binary coding requires a very large number of calculations. It then leads to the complexity of the algorithm as well as the lag of computers. Therefore, we introduce a new way of encoding called Natural coding and then we find out that the algorithm also converges to the global optimum.

Assume that there are \( n \) jobs processed by \( m \) machines. The total of sections which needs to be processed for all jobs equals to \( m \times n \). We encode the sections of \( J_1 \) by \( 1 \) to \( m \), \( J_2 \) by \( m + 1 \) to \( 2m \), and \( J_3 \) by \( (n - 1)m + 1 \) to \( m \times n \). Hence, each solution is a permutation of the sequence of natural numbers \( \{1, 2, 3, \ldots, m \times n\} \) satisfying the constrains of the problem. For instance, for \( 3 \times 3 \) JSP given in Table 1, the sections are encoded by natural numbers as in Table 2.

<table>
<thead>
<tr>
<th>Jobs</th>
<th>Coding of sections</th>
</tr>
</thead>
<tbody>
<tr>
<td>( J_1 )</td>
<td>1 2 3</td>
</tr>
<tr>
<td>( J_2 )</td>
<td>4 5 6</td>
</tr>
<tr>
<td>( J_3 )</td>
<td>7 8 9</td>
</tr>
</tbody>
</table>

So, a regular solution of the problem might be a form of Figure 1.

\[
\begin{bmatrix}
1 & 4 & 8 & 7 & 6 & 2 & 5 & 3 & 9
\end{bmatrix}
\]

**Figure 1: A regular solution of 3 \times 3 JSP**

B. **The traditional evolutionary algorithm**

Begin
- \( t = 0 \)
- Initialize \( P(t) \) with a super individual
- Perform selection
- Determine the fitness of each individual
- Perform crossover
- Perform mutation
- Determine the fitness of each individual
- Perform selection
- Determine the fitness of each individual
- Until some stopping criterion applies

End.

This traditional algorithm does not converge to the global optimum [4]. Therefore, we improve the algorithm by an additional, say, super individual which does not take part in the evolutionary process.

C. **Evolutionary algorithm with super individuals**

In this section, we only present the algorithm summarily. The details of the algorithm can be found in [9].

The super individual is an individual that has special properties. It does not take part in the processes including crossover, mutation and selection. After performing on three basic operations in the population, we perform a new operation, say copy operation. This operation copies the individual which has higher adaptation than the super individual in the same state to the position of that super individual. The evolutionary algorithm with super individuals is as follow:

Begin
- \( t = 0 \)
- Initialize \( P(t) \) with a super individual
- Perform selection
- Determine the fitness of each individual
- Perform crossover
- Perform mutation
- Determine the fitness of each individual
- Perform selection
- Determine the fitness of each individual
- Until some stopping criterion applies

End.

III. **FINITE MARKOV CHAINS**

A. **Definitions of a Markov Chain**

Let us consider an economic or physical system \( S \) with \( m \) possible states, represented by the set \( I \):

\[ I = \{1, 2, \ldots, m\} \]

Let the system \( S \) evolve randomly in discrete time \( t = 0, 1, 2, \ldots, n, \ldots \), and let \( J_n \) be the random variable (r.v) representing the state of the system \( S \) at time \( n \).

**Definition 1.** The random sequence \( \{J_n, n \in N\} \) is a Markov chain if and only if for all \( j_0, j_1, \ldots, j_n \in I \):

\[
P(J_{n+1} = j_{n+1} / J_0 = j_0, \ldots, J_n = j_n) = P(J_{n+1} = j_{n+1} / J_n = j_n) \quad (1)
\]

(provided this probability has meaning).

**Definition 2.** A Markov chain \( J_n; n \geq 0 \) is homogenous if and only if probabilities (1) do not depend on \( n \) and are non-homogenous in other cases.

For this regard, we will only consider the homogenous case, for which we write:

\[
P(J_{n+1} = j_{n+1} / J_n = j_n) = p_{j_0} \quad \text{and we introduce matrix } P \text{ defined as: } P = [p_{j_0}].
\]

The elements of matrix \( P \) have the following properties:
(i) \( p_{ij} \geq 0 \), for all \( i, j \in I \) and (ii) \( \sum_{j=1}^{I} p_{ij} = 1 \).

A matrix \( P \) satisfying these two conditions is called to be a Markov matrix or transition matrix.

To fully define the evolution of a Markov chain, it is also necessary to fix an initial distribution for state \( J_0 \), i.e. a vector \( \Gamma^{(0)} = [\pi_1^{(0)}, \pi_2^{(0)}, \ldots, \pi_m^{(0)}] \), such that: \( \pi_i^{(0)} \geq 0, i \in I \) and \( \sum_{i=1}^{m} \pi_i^{(0)} = 1 \).

For all \( i, \pi_i \) represents the initial probability of starting from \( i: \pi_i = P(J_0 = i) \).

In the same manner, if we denote \( \Gamma^{(n)} \) as the distribution for state \( J_n \), i.e. a vector \( \Gamma^{(n)} = [\pi_1^{(n)}, \pi_2^{(n)}, \ldots, \pi_m^{(n)}] \) on the meaning of \( \pi_i^{(n)} = P(J_n = i) \). We also have: \( \pi_i^{(n)} \geq 0, i \in I \) and \( \sum_{i=1}^{m} \pi_i^{(n)} = 1 \).

We now introduce the transition probabilities of order \( n \), \( \Gamma_{ij}^{(n)} \), defined as:

\[
\Gamma_{ij}^{(n)} = P(J_{n+j} = j|J_i = i).
\]

Similarly, we have a transition matrix after \( n \) steps:

\[
\Gamma^{(n)} = [\Gamma_{ij}^{(n)}]
\]

It has been found out that a homogenous Markov chain is characterized by the couple \((\Gamma^{(0)}, P)\).

**B. Basic properties of Markov Chains**

**Theorem 1.** ([113], page 409)

Let \( P \) be the transition matrix of a Markov chain, element \( \Gamma_{ij}^{(n)} \) of matrix \( P^n \) is also the probability of the transition from state \( i \) to state \( j \) of the system after \( n \) steps. In other words, the transition matrix after \( n \) steps of a Markov chain is the power of \( n \) of the Markov matrix: \( \Gamma^{(n)} = [\Gamma_{ij}^{(n)}] = P^n \).

For \( \Gamma^{(0)} \), from \( \Gamma^{(n)} = \Gamma^{(0)} \times P^n \), we have:

\[
\Gamma^{(n)} = \Gamma^{(0)} \times P^n.
\]

Hence, the probability distribution of the states of a process with Markov property only depends on the couple \((\Gamma^{(0)}, P)\).

There are two typical types of Markov Chains:

- Absorbing Markov Chain: State \( i \) is said to be absorbing if when the system reaches state \( i \), it cannot transit to any other state (i.e \( p_{ii} = 1 \)). An absorbing Markov chain is a Markov chain which has at least one absorbing state and from any non-absorbing states, it can reach an absorbing state (unnecessary in one step).

- Ergodic Markov Chain: Section below will learn for more detail of the Ergodic Markov Chains, which its properties have a very important role in investigating the GA algorithm convergence.

C. **Ergodic Markov Chain**

**Definition 3.** A Markov chain is said to be Ergodic Markov chain if from any state, it can reach to any other state (unnecessary in one step), i.e. \( \exists k \in N: p_{ij}^k > 0 \) for all \( i, j \in I \).

Therefore, a primitive Markov chain is an Ergodic Markov chain.

Now we focus on some important properties of Ergodic Markov chains used for proving the convergence of the JSP problem in this paper.

**Theorem 2.** ([7], page 123) Let \( P \) be a primitive stochastic matrix. Then \( P^k \) converges as \( k \to \infty \) to a positive stable stochastic matrix \( P^\infty = 1' \Pi^\infty \) where \( 1' \) is a matrix which has all elements are 1 and \( \Pi^\infty = \Pi^{(0)} \lim_{k \to \infty} P^k = \Pi^{(0)} \cdot P^\infty \) is a vector of probability distribution has nonzero entries and is unique regardless of the initial distribution.

Theorem 2 leads to a conclusion that the probability distribution of the states of an Ergodic Markov chain will be stable as time tends to be infinite regardless of the initial distribution \( \Pi^{(0)} \). This is the most important property of Markov chain using to prove convergence of GA algorithm.

Here we have another important theorem.

**Theorem 3.** ([7], page 126) Let \( P \) be an \( n \times n \) be a reducible stochastic matrix, meaning \( P = \begin{pmatrix} C & 0 \\ R & T \end{pmatrix} \). \( C: m \times m \) be a primitive stochastic matrix and \( R, T \neq 0 \), we have:

\[
P^\infty = \lim_{k \to \infty} P^k = \lim_{k \to \infty} \begin{pmatrix} C^k & 0 \\ \sum_{i=0}^{k} R^i C^i T^k & T^k \end{pmatrix} = \begin{pmatrix} C^\infty & 0 \\ R^\infty & 0 \end{pmatrix}
\]

is a stable matrix where \( P^\infty = 1' \Pi^\infty \) and \( \Pi^\infty = \Pi^{(0)} \times P^\infty \) are stable row vectors of \( P^\infty \) that do not depend on \( \Pi^{(0)} \) and satisfy following conditions:

\[
\pi_i^\infty \geq 0 \text{ for all } 1 \leq i \leq m \text{ and } \pi_i = 0 \text{ for all } m < i \leq n.
\]

**Lemma 1.** Let \( C: M \) and \( S \) be stochastic matrices, where \( M \) is positive and \( S \) is column allowable (in which has at least one positive entry in each column). Then the product \( CMS \) is a positive matrix.

For more details of Markov Chains, reader can see in [8].

3
IV. CONVERGENCE ANALYSIS OF THE NEW HYBRID GENETIC ALGORITHM FOR JOB SHOP SCHEDULING PROBLEM

A. Traditional genetic algorithm

The problem is assumed for $n$ jobs and $m$ machines with given industrial serial. Each solution counts as an individual, each job processed on each machine counts as a gene. Let $N$ be number of individuals of the population. Then, each element of the state space $i$ can be regarded as a natural number series with the length of $n \times m \times N$, where the projection $\pi_i(i)$ picks up the $k$-th individual in the population. The research space has a total of $(n!m^nN)$ states.

1) Transition matrix of the NHGA caused by crossover

Consider hybrid operations, let $C$ be the transition matrix caused by crossover, we have:

$$
C = \begin{pmatrix}
c_{11} & \cdots & c_{1(n!m^nN)} \\
\vdots & \ddots & \vdots \\
c_{(n!m^nN)} & \cdots & c_{(n!m^nN)(n!m^nN)}
\end{pmatrix}
$$

where $c_{ij}$ is the probability of the population transiting from state $i$ to state $j$ with probability $p_c$ of the hybridization.

For the inputs of the traditional genetic algorithm, hybrid probability $p_c \in [0, 1]$, the hybrid processes happen by choosing any parent from the population at current state $i$ with the probability of $p_c$. Then perform crossover for three individuals by GT law randomly [14]. After these hybrid processes, the population can be at state $j$ with a probability of $c_{ij}$ and $\sum_{j=1}^{(n!m^nN)} c_{ij} = 1$.

This $c_{ij}$ does not depend on what order and what time the population is, but only depends on $p_c$ given due to the inputs of algorithm.

Therefore, transition matrix caused by crossover $C$ is a specific stochastic matrix which does not depend on current state as well as the time that the population is.

2) Transition matrix of NHGA caused by mutation

Let $i$ be the state at the time $t$, and $\pi_i(i)$ be the $k$-th individual in the population of $N$ individuals. $\pi_k(\pi_i(i))$ be the $h$-th machines in $\pi_k(i)$ and $\pi_l(\pi_h(\pi_k(i)))$ be the gene at the $l$-th location. All of these are described as in Figure 2.

The mutation algorithm for $\pi_k(i)$ can be described shortly as follow:

- Choose $\pi_k(i)$ to make mutation with a probability of $p_m > 0$ (very small). So the probability that the mutation does not occur is $1 - p_m$.
- When $\pi_k(i)$ occurred mutation, the mutation process as follow:

For machines $\pi_h(\pi_k(i)); h = 1, \ldots, m$, at each of these machines:

- Do not make the mutation with probability of $p > 0$ (close to 1).
- Inversely, replace $h$-th machine with any permutation, different from the identity with the same ability.

So the probability for each of these changes is $\frac{1-p}{n!-1}$.

For this algorithm, we can show that transition probabilities caused by mutation are positive. Indeed, let $m_{ij}$ be the probability that the system transits from state $i$ to state $j$ caused by mutation, then it can be calculated as below.

Consider any two states $i, j$. Assume that $i$ and $j$ have $K$ identical individuals and $N - K$ different individuals at the same order in the population, then:

- The probability for the $K$ individuals (mutation do not occur) is $(1 - p_m)^K$.
- For $N - K$ couple of different individuals, $\pi_k(i)$ and $\pi_k(j)$, for $t = 1, \ldots, N - K$. Let $L_k$ be the number of machines that have the same genes between $\pi_k(i)$ and $\pi_k(j)$, so $m - L_k$ be the number of machines do not.

- The probability for the $L_k$ identical machines is $p^{L_k}$.
- The probability for $m - L_k$ remaining different machines is $\left(1 - \frac{p}{n!-1}\right)^{m-L_k}$.

Therefore, $m_{ij} = (1 - p_m)^K \cdot p_m^{N-K} \prod_{t=1}^{N-K} \left(p^{L_t} \cdot \left(1 - \frac{p}{n!-1}\right)^{m-L_t}\right) > 0$

Hence, $M = [m_{ij}]$ is an positive matrix.

3) Transition matrix of NHGA caused by selection

The third operator that is performed to generate a new population from the previous one is selection operator. Let $S$ be the transition matrix caused by selection, we have:

$$
S = \begin{pmatrix}
s_{11} & \cdots & s_{1(n!m^nN)} \\
\vdots & \ddots & \vdots \\
s_{(n!m^nN)} & \cdots & s_{(n!m^nN)(n!m^nN)}
\end{pmatrix}
$$
For \( s_i \), the probability of the transition from state \( i \) to state \( j \) caused by selection, we also have: 
\[
\sum_{j=1}^{(mN)^N} s_{ij} = 1.
\]

The probability, for individual \( \pi_k(i) \) is preserved in its population after selection, is: 
\[
f(\pi_k(i))/\sum_{h=1}^{N} f(\pi_h(i)).
\]

Hence, the probability for current population preserving state \( i \) after selection will be 
\[
s_i = \frac{\prod_{k=1}^{N} f(\pi_k(i))}{\sum_{k=1}^{N} f(\pi_k(i))}.
\]

Since \( f \) is a positive function we derive \( s_i > 0 \). So, the transition matrix \( S \) of the NHGA caused by selection is a stochastic, column allowable matrix and does not depend on current states as well as the time of the population.

After all, let \( P \) be the transition matrix of the NHGA throughout the process of the traditional genetic algorithm, we have:
\[
P = \begin{pmatrix}
P_{11} & \cdots & P_{1(nmN)} \\
\vdots & \ddots & \vdots \\
P_{(nmN)1} & \cdots & P_{(nmN)(nmN)}
\end{pmatrix} = C \times M \times S.
\]

Since the matrices \( C, M \) and \( S \) are stochastic matrices that do not depend on the time and current state of the population, the matrix \( P \) to do so. On the other hand, \( C, M \) and \( S \) are the stochastic matrices, \( M \) is positive and \( S \) is column allowable, according to Lemma 1 then \( P \) is a positive transition matrix. So this algorithm can be considered as an Ergodic Markov chain.

### B. Genetic algorithm with super individuals

With the appearance of super individuals, the cardinality of the population increases to \( N + 1 \). The cardinality of the state space grows from \((n!)^{mN}\) to \((n!)^{m(N+1)}\).

We locate the super individual's location at the beginning of the encoded series with the length of \((N+1) \times m \times n\) and it can be accessible by \( \pi_o(i) \) from a population at state \( i \). We arrange the state space in decreasing order of selection of super individuals, i.e \( i < j \rightarrow f(\pi_o(i)) > f(\pi_o(j)) \).

Then the transition matrices caused by crossover, mutation and selection have the size of \((n!)^{m(N+1)} \times (n!)^{m(N+1)}\). Denote \( C^+, M^+, S^+ \) are the transition matrices caused by crossover, mutation and selection respectively of the population with super individuals. Since the super individuals are not affected by genetic operations, the matrices \( C^+, M^+, S^+ \) have the form of:
\[
C^+ = \begin{pmatrix}
C & \cdots \\
\vdots & \ddots \\
C & \cdots 
\end{pmatrix},
M^+ = \begin{pmatrix}
M \\
\vdots \\
M 
\end{pmatrix},
S^+ = \begin{pmatrix}
S \\
\vdots \\
S 
\end{pmatrix}
\]

where each diagonal of them includes \((n!)^m\) matrices \( C, M, S \) considered in the traditional genetic algorithm without super individuals. From that we have:
\[
C^+M^+S^+ = \begin{pmatrix}
C.M.S \\
\vdots \\
C.M.S
\end{pmatrix} = \begin{pmatrix}
P \\
\vdots \\
P
\end{pmatrix}
\]

The copy operation is represented by an upgrade matrix \( U \) with the size of \((n!)^{m(N+1)} \times (n!)^{m(N+1)}\), in which its elements are defined as follow.

Consider a population with a super individual at state \( i \). Let \( \pi_o(i) \) be the individual with the highest adaptation among the usual individuals, i.e \( f(\pi_o(i)) = \max\{f(\pi_j(i))/k = 1, \ldots, N\} \), then \( u_{ij} = 1 \) if \( f(\pi_o(i)) < f(\pi_j(i)) \) with \( j = \{\pi_1(i), \pi_2(i), \ldots, \pi_N(i)\} \), otherwise \( u_{ij} = 1 \).

The copy operation replaces the super individual by the usual individual which has higher adaptation into super individual's location. If there are not any usual individuals satisfying, the population is preserved.

It is clear that the matrix \( U \) also only depends on \( f \) function's values. So the matrix \( U \) for a given-size NHGA is determinant. Since the order of the states is decreasing arrangement of the adaptability of the super individuals, matrix \( U \) has the form as:
\[
U = \begin{pmatrix}
U_{11} & \cdots \\
\vdots & \ddots \\
U_{(nmN)1} & \cdots & U_{(nmN)(nmN)}
\end{pmatrix}
\]

where the matrices \( U_{ij} \) have the size of \((n!)^{mN} \times (n!)^{mN}\).

To ease the presentation, assume that problem (*) has only one global optimum. Then only \( U_{11} \) is a unit matrix whereas all matrices \( U_{\alpha\alpha} \) with \( \alpha \geq 2 \) are unit matrices with some zero diagonal entries. So the transition matrix \( P = C \times MS \) for the NHGA becomes
\[
P^* = C^+ \times M^+ \times S^+ \times U = \begin{pmatrix}
P & \cdots \\
\vdots & \ddots \\
P & \cdots 
\end{pmatrix} \times \begin{pmatrix}
U_{11} & \cdots \\
\vdots & \ddots \\
U_{(nmN)1} & \cdots & U_{(nmN)(nmN)}
\end{pmatrix}
\]
\[
= \begin{pmatrix}
PU_{11} & \cdots \\
\vdots & \ddots \\
PU_{(nmN)1} & \cdots & PU_{(nmN)(nmN)}
\end{pmatrix}
\]

Then we can prove the NHGA with super individuals converges to the global optimum as the number of generations tends to be infinite.
Indeed, because $U_{11}$ is a unit matrix then $PU_{11} = P$ is a positive stochastic matrix gathers the transition probabilities for states containing a globally optimal super individual (globally optimal states). Since the matrices $U_{aa}$ with $a \geq 2$ are unit matrices with some zero diagonal entries so $PU_{aa} \neq 0$ for all $k \in \{2, \ldots, (n1)^m\}$. Therefore, the sub-matrices $PU_{a1}$ with $a \geq 2$ may be gathered in a rectangular matrix $R \neq 0$ and the adjunct matrix of $P$, $T$, is also not equal 0:

$$T = \begin{pmatrix}
PU_{22} & \cdots & PU_{2j} \\
\vdots & \ddots & \vdots \\
PU_{mj} & \cdots & PU_{jj}
\end{pmatrix}$$

Theorem 3 guarantees that the probability of staying in any non-globally optimal state converges to zero. It follows that the probability of being in any globally optimal state converges to one, so that the limit of $P(Z_t = f)$ converges to one for $t \to \infty$ as well:

$$\lim_{t \to \infty} P\{Z_t = f\} = 1.$$ 

V. CONCLUSIONS

This article gives a proof of the convergence to the global optimum of a new hybrid genetic algorithm we proposed for the Job Shop Scheduling Problems [9]. The proof of the convergence of GA in the case of binary coding was solved by Günter Rudolph in 1994. However, encoding by binary coding keeps much restriction in applying Genetic algorithms to complex problems in real life. In this regard, we proved the convergence to the global optimum of GAs using natural coding based on Markov chain analysis, which has been an open issue for years.

ACKNOWLEDGMENT

We would like to thank Günter Rudolph for his proof of the convergence of Canonical Genetic Algorithms in the case of binary coding. We also thank to colleagues in the Department.

This paper is an extended and revised version of [9].

REFERENCES