Evaluation Of Flowshop Structures for Scheduling

Xianyi Zeng and Michel Happiette
GEMTEX ENSAIT
2, Place des Martyrs de la Résistance
59070 Roubaix
FRANCE
e-mail: Xian.Zeng@univ-lille1.fr

Abstract: This paper presents an algorithm for evaluating the Flowshop structures in terms of scheduling admissibility. These Flowshop structures are generated from a procedure of optimal layout of production. A learning automaton is used in this approach. The scheduling behaviors of the Flowshop structures can be evaluated by this learning automaton without calculating specific scheduling solutions.

Keywords: learning automaton, layout of production, evaluation, scheduling

Xianyi Zeng received the M.Sc. degree in Computer Engineering from the University of Qinghua, P.R.China, and the Ph.D degree in Automation from the University of Lille I, France.
He has been an Assistant Professor of Automation at ENSAIT since 1993. His current research interests include intelligent techniques, soft computing and their applications in textile industry.

Michel Happiette received the M.Sc. degree in Textile Engineering from the Ecole Nationale Supérieure des Arts et Industries Textiles (ENSAIT) of Roubaix, France, and the Ph.D in Automation from the University of Lille I, France.
He has been an Assistant Professor of Production Management at ENSAIT since 1984. His current research interests include production management and its applications in textile industry.

1. Introduction

New economic markets, under the impacts of international competitions, ask for more performant procedures in industry to solve scheduling problems and optimal layout of production [1], [2], [5]. In general, a workshop can be characterized by either Jobshop or Flowshop structure [3], [4], [6], [7]. In a Jobshop structure, the processing order of each item on machines is well defined and it may be different for different items. But in a Flowshop structure, this order is identical for all items.

A great number of successful algorithms have been proposed to solve scheduling problems in Flowshop structures [8]. Jobshop scheduling problems, which are considered more complex, can be solved using the existing Flowshop scheduling algorithms. In [17], a general algorithm was proposed for decomposing the original complex Jobshop problem into simpler Flowshop scheduling subproblems. This decomposition permits to bring about a double classification of machines and products in order to define homogeneous production subsystems. The combination of the solutions to these subproblems leads to a general solution to the original scheduling problem.

Each Flowshop structure corresponds to a linear production family (an ordered subset of machines) where the scheduling of all items belonging to this family is realized without any backward move of flow of products. Such a structure, which has been studied in our previous papers [9] [13], has the sequence of the range of production of each item in this family included in the sequence of the corresponding layout of production.

To determine the minimal number of machines in the layout of production, we have built a tree structure which is conditioned at each of its nodes by applying two rules to reduce the expansion of the searching space. This tree structure has been presented in literature for the detection of order-consistency on a set of syntactic patterns [6], [10]. Similar structure was also defined in [11] to solve the string correction problem.

The general scheme of the decomposition of Jobshop into Flowshop structures is shown in Figure 1. It starts from creating an initial partition of items of production. The algorithm which we proposed in [13] permits to define an hierarchical classification by minimizing an internal index which is defined as linear combination of the compactness inside each class and the separability between different classes [12]. This procedure makes us obtain the initial partitions for both items of production and machines.

After performing the procedure of layout of production (Figure 1), each class of items should correspond to a minimal number of machines, whose combination constitutes a solution of layout of production. If no layout solution is found, our algorithm will try to extend the space of layout solutions by integrating a minimal number of additional machines. The fail of the layout extension leads to exchanging items between classes or to eliminating some items so that the new partition of the remaining items leads to successful solutions of layout for each class.

Once several solutions of layout of production are found together, we select one solution according to an evaluation criterion, and start the scheduling procedure in order to find admissible solutions. The fail of the scheduling makes the algorithm

return to the layout procedure and repeat the previous three operations: adding a minimal number of machines, exchanging items between classes and eliminating some items.

The operations shown in Figure 1 are performed in uncertain environments and they often deal with imprecise data. The uncertainties of these operations can be summarized as follows:. 1) Layout procedure: we do not know the best method to obtain the optimal layout solutions with a minimal number of calculations. 2) Layout solution: we do not know what is the layout solution leading to admissible scheduling solutions. 3) Classification: we do not know what is the best partition of items so that each class corresponds to a Flowshop structure with a minimal number of machines.

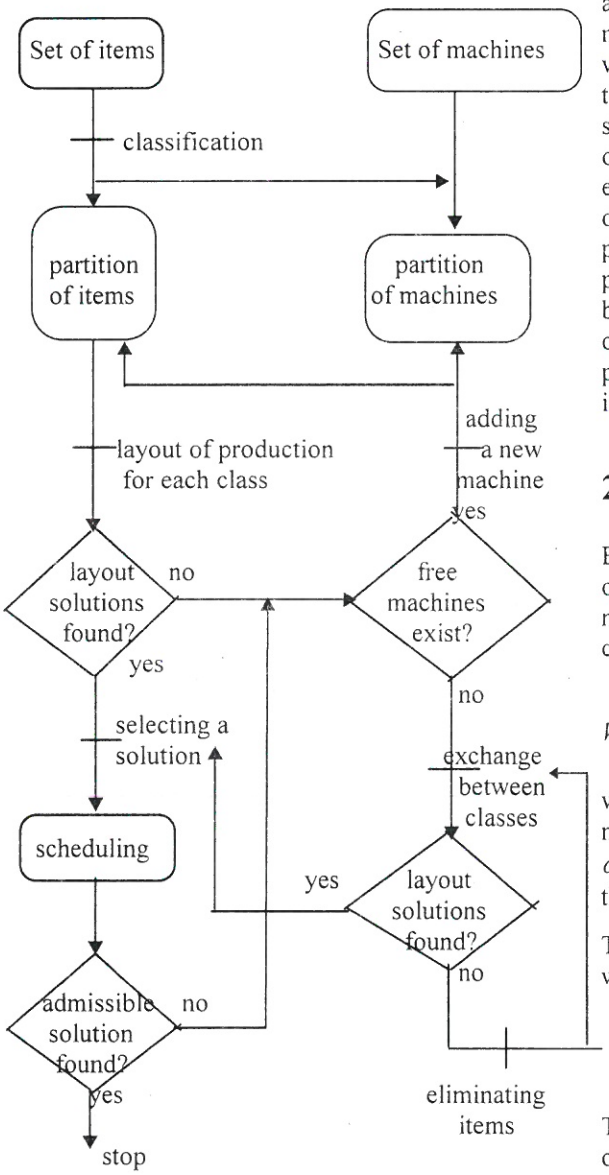


Figure 1. General Scheme of the Decomposition

Under these uncertainties, the operations of the decomposition risk heavy calculations or even no solution. To overcome this inconveniency, we propose an adaptive heuristic strategy in order to control the alternation between the scheduling procedure and the optimal layout of production. According to this adaptive strategy, at least one admissible scheduling solution is delivered to the workshop planning maker.

In this paper, we present only the procedure of construction and evaluation of Flowshop structures for a given family of items of production. Two operations of the general algorithm will be processed: a) Optimal layout procedure of production, which permits to generate solutions of layout. This operation improves the idea presented in [12] by introducing a new procedure of adding a minimal number of machines. b) Evaluation of solutions of layout, which permits to select one solution of layout so that the corresponding scheduling can be successful with maximal probability. This operation utilizes a learning automaton in order to evaluate the scheduling behaviors of each solution of layout and to generate the corresponding probability (action probability) for the current period. In the next period, a solution of layout will be selected at random according to the distribution of all the action probabilities. This operation permits to converge to the best solution of layout in the sense of scheduling admissibility.

2. Problem Modelling

Each of the items to be produced is defined by the ordered set of the machines required for its manufacturing (repeat of machines is possible). It can be described by a word as follows:

$$W=\alpha_1^{n\alpha_1}\alpha_2^{n\alpha_2}\ldots\ldots\alpha_m^{n\alpha_m}=(\alpha_i^{n\alpha_i}),\quad i=1,\ldots\ldots,m$$

where α_i is a letter representing a machine for the manufacturing of the item W and we have $\alpha_i \in ALPHA$, a set containing all the letters. $n\alpha_i$ is the multiplicity of the machine α_i.

The total multiplicity of a given machine α in the word W is then

$$N\alpha(W)=\sum_{\alpha_i=\alpha} n\alpha_i$$

The set $H(W)=\{(\alpha, N\alpha)\}$ is the minimal alphabet on which the word W can be constructed.

For example, let W be the word describing the following working process: $W=a^2bcac^3d$, the corresponding $H(W)=\{(a, 3), (b, 1), (c, 4), (d, 1)\}$.

We can see from this example that each item or word can be composed of several machines of the same type. In this paper, we are interested in the possibility of building a set of Flowshop production structures where each of them is served by a linear conveyor. In this way, the statement of the problem is the following:

Searching for the possibility of grouping, within the same production structure, the machines' succession (words) relative to different items. The chosen type of conveyor then imposes a condition of compatibility between the words grouped together: the order of the machines laid out along the conveyor must necessarily correspond to the order imposed by each of the machines' succession.

According to [9], for a given class C, the items (words) are manufactured by the minimal set of the machines corresponding to C, defined by: $H(C)=\{(\alpha, N\alpha)\}$ where $\alpha \in \{\cup M_i, i \in C\}$ and $N\alpha = max\{N\alpha_i, i \in C\}$.

For example, let $C=\{W_1, W_2, W_3, W_4\}$ with $W_1=aba^2c$, $W_2=bdcba$, $W_3=adabc$, $W_4=cbac$. We have $H(C)=\{(a, 3), (b, 2), (c, 2), (d, 1)\}$ and those four items are order -consistent. The following layout of the machines allows the production of each of them without any backward move: $W(C)=abdacbac$.

3. Searching for Layout of Production

In order to search for the layout of production for the set $C=\{W_1, W_2, \ldots\ldots, W_n\}$, we adopt the algorithm for the detection of order consistency published in [9]. This algorithm is briefly presented below:

For each word W_i, an expansion is progressively constructed by setting at the k^{th} position either the first character of W_i which has not been used yet, or a joker. Our objective is to build a generalization which is common to all the words of C. The set is order-consistent iff all the characters have been located for each $W_i \in C$ and the obtained generalization uses only characters from $H(C)$. Our algorithm uses a searching procedure based on a tree of layout permitting, at each step, the elimination of useless nodes by applying two rules R_1 and R_2. If class C is order-consistent, on the terminal useful nodes can be found all the possible layouts associated with C. If C is not order-consistent, all the terminal nodes on the tree of layout are useless and there does not exist any layout solution associated with C.

The two rules R_1 and R_2 are given as follows:

R_1: Let s_k be a node such that: $\exists(\alpha, W_i) \in ALPHA \times C$ s.t. $N\alpha(C/s_k) < N\alpha(Wi/s_k)$. The success condition cannot be satisfied by any terminal node of the tree having s_k as a predecessor; s_k will be labelled as useless and its successors will not be explored [9].

R_2: The node s_k is useless if there exist two words $W_i, W_j \in C$ and two letters $\alpha, \beta \in ALPHA$ s.t. $(\alpha, 1), (\beta, 1) \in H(W_i/s_k)$ and $H(W_j/s_k)$ with $\alpha < \beta$ in W_i/s_k and $\beta < \alpha$ in Wj/s_k ("<" is the order between two letters in a sequence).

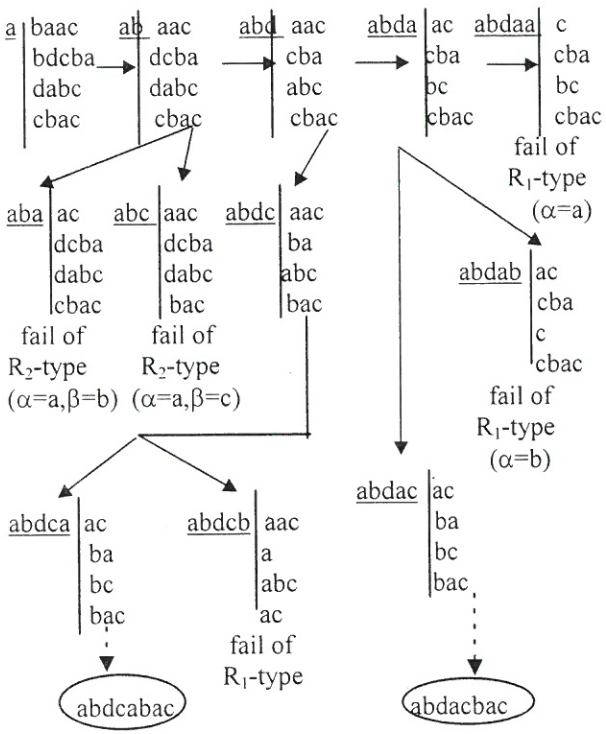


Figure 2. The Exploration Tree of Layout

According to R_2, one couple of subsequences (α β) and (β α) belonging to W_i and W_j respectively leads to a fail or a backward move.

Remark: In the rules R_1 and R_2, s_k is the common generalization of the expansions of the first sequences of the words in C. C/s_k represents the set of machines in the remaining resources of $H(C)$ and W_i/s_k represents the set of machines in the remaining needs of W_i.

One example of searching for layout of production is given as follows:

Assuming that we have 4 words: $W_1=abaac$, $W_2=bdcba$, $W_3=adabc$, $W_4=cbac$, the corresponding minimal set of machines and its total multiplicity can be calculated

$H(C)=\{(a,3)(b,2),(c,2),(d,1)\}$ and
$N(C)=3+2+2+1=8$

Two solutions of layout (*abdcabac* and *abdacbac*) can be found in Figure 2. Each of them corresponds to a Flowshop structure. For the solution *abdcabac*, we get the following Flowshop: $M_1=a$, $M_2=b$, $M_3=d$, $M_4=c$, $M_5=a$, $M_6=b$, $M_7=a$, $M_8=c$. The corresponding four items of production can be rewritten as: $J_1=(O_{11}\ O_{12}\ O_{15}\ O_{17}\ O_{18})$, $J_2=(O_{22}\ O_{23}\ O_{24}\ O_{26}\ O_{27})$, $J_3=(O_{31}\ O_{33}\ O_{35}\ O_{36}\ O_{38})$, $J_4=(O_{44}\ O_{46}\ O_{47}\ O_{48})$

If no solution of layout is obtained for a given set of items of production, i.e. all the terminal nodes in the exploration tree of layout fail, we add a new machine so that the optimal layout of production can continue. This procedure is presented below.

1) Choosing a terminal useless node according to the following priorities:

P_1: Taking the node where the number of fails (R_1-type or R_2-type) is minimal.

P_2: For different nodes, if the numbers of fails are equal, taking the node whose branch is the longest.

2) For the selected node, if the fail is R_2-type (α, β non order -consistent), then we add a new machine having the same type as that of the last letter (α or β) in the generalization string of this node. If the fail is R_1-type (the incompatible letter is α), then we add a machine α to the set of items.

Adding a new machine permits to continue the generalization procedure from the selected node. If no solution of layout is found in this node, we add another new machine according to the priorities defined a priori. This procedure repeats until at least one solution of layout is obtained.

4. Learning Automata

If several useful terminal nodes are found together from the tree of layout, we do not know in advance which solution of layout is the best for making the scheduling successful. It is necessary to perform an evaluation for each existing solution to find the one whose probability of successful scheduling is maximal. Each probability should be defined as a function of scheduling behaviors of the corresponding solution of layout.

In this paper, a learning automaton (L.A.) is used to select the "optimal" solution of layout in the sense of scheduling admissibility.

L.A. is an adaptive decision-making device operating on an unknown random environment and has been used as models of learning system [14], [15]. The L.A. has a finite set of actions and each action has a certain probability (unknown to the automaton) of getting rewarded by the environment. The aim is to learn to choose the optimal action (i.e. the action with the highest probability of being rewarded) through repeated interaction with the environment. If the learning algorithm is chosen properly, then the iterative process of interacting with the environment can be made result in the optimal action being selected with the highest probability.

A L.A. is a stochastic automaton in feedback connection with a random environment [15] (See Figure 3). The output of the L.A. (actions) is the input to the environment and the output of the environment (responses) is the input to the L.A.

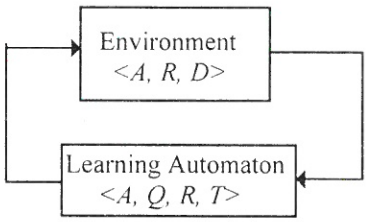


Figure 3. General Scheme of Learning Automaton

In general, a L.A. is defined by (A, Q, R, T) and the environment by (A, R, D), where

$A=\{\alpha_1, \alpha_2, \ldots\ldots, \alpha_r\}$ is the set of all actions of the automaton. The action of the automaton at instant k is denoted by $\alpha(k)$ and $\alpha(k) \in A$ for $k=0, 1, 2, \ldots\ldots$ Evidently, A is the set of outputs of the automaton and it is also the set of inputs to the environment.

R is the domain of responses from the environment. Let $\beta(k)$ denote the response received by the automaton at instant k where $\beta(k) \in R$, $\forall k$. $\beta(k)$ is the output of the environment at instant k and it is also the input to the automaton.

$D=\{d_1, d_2, \ldots\ldots, d_r\}$ is the set of reward probabilities, where

$d_i(k)=E[\beta(k) \mid \alpha(k)=\alpha_i]$

If the d_i's are independent of k, the environment is called stationary. Otherwise, it is nonstationary. The reward probabilities are unknown to the automaton.

Q is the state of the automaton defined by

$Q(k)=[P(k), \bar{D}(k)]$

where $P(k)=[P_1(k), P_2(k), \ldots\ldots, P_r(k)]$,

($\forall k$, we have $0 \le p_i(k) \le 1$ and $\sum_{i=1}^{r} p_i(k) = 1$)

is the action probability vector and

$$\bar{D}(k) = \left[\bar{d}_1(k), \ldots\ldots, \bar{d}_r(k)\right]$$

is the vector of estimates of the reward probabilities at the *k-th* instant.

T is the learning algorithm or the reinforcement scheme which is used by the automaton in order to update its state. At instant k, we get from T

$Q(k+1)=T(Q(k), \alpha(k), \beta(k))$

During the execution of the algorithm, the automaton selects an action $\alpha(k)$ from the set of actions A at each instant k. The selection of actions depends on its current action probability vector $P(k)$. The selected action $\alpha(k)$ becomes an input to the environment and the environment gives the input of the automaton a random response $\beta(k)$ whose expected value is d_i if $\alpha(k)=\alpha_i$. Next, the automaton calculates $Q(k+1)$ using the reinforcement scheme T.

This procedure is repeated until the optimal action to the environment is found, i.e.

$d_m=\max_j \{d_j\}$ where α_m is the optimal action.

The action α_m has the maximum probability of being rewarded. It is desired that the action probability corresponding to α_m (i.e. p_m) tends to unity as the time k goes to infinity.

Many criteria have been proposed to evaluate the performance of learning automata. One of them is ε-optimal criterion, which is frequently used to evaluate the asymptotic behavior of learning algorithm of the automata. It is defined as follows:

Let m be the index of the optimal action. A learning algorithm is said to be ε-optimal if

$$\lim_{k\to\infty} (\inf p_m(k))>1-\varepsilon$$

for any ε>0, by choosing sufficiently small values of the internal parameter of the learning algorithm.

For simplicity of notations, we do not distinguish between the reward probabilities and their estimates in the following discussion.

5. Evaluation of Solutions of Layout

5.1 The Reinforcement Scheme of the L.A.

In this paper, the universe of scheduling solutions is considered as random environment and each solution of layout is considered as an action whose probability is calculated from the evaluation of the scheduling behaviors. According to this idea, we give in Figure 4 the scheme of L.A.

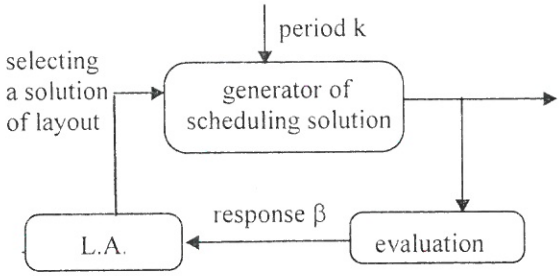


Figure 4. Learning Automaton in Feedback Connection with Scheduling

In the L.A., we adopt the Pursuit Algorithm [18] as a reinforcement scheme. It has been proved to be ε-optimal if the environment is stationary. The principle of this algorithm is illustrated as follows.

First, we define several notations:

$X_i(k)$: total reward obtained for the i-th action until k-th sampling period.

$n_i(k)$: number of times the i-th action is chosen until k-th sampling period.

Next, we apply the Pursuit Algorithm to the problem for evaluating the solutions of layout.

At sampling period 0 ($k=0$), set $X_i(k)=0$, $n_i(k)=0$ and $P_i(k)=1/r$ for $i=1, 2, \ldots, r$. Initialize $d(k)$ by picking each action for a small number of times and setting $d_i(k)$ to the average reaction obtained.

At sampling period k ($k\geq 0$),

i) $\alpha(k)=\alpha_i$ is selected according to the distribution of the action probabilities $P(k)$.

ii) Obtain at random a solution from the generator of scheduling solutions. The universe of the corresponding scheduling solutions is associated with the current solution of layout $\alpha(k)$.

iii) Calculate the response function $\beta(k)$ and update action probabilities $P(k)$ and reward probabilities $D(k)$, i.e.

$P(k+1)=P(k)+\mu(e_{M(k)}-P(k))$

where e_j is a r-dimensional vector with *j-th* component unity and all others zero, $M(k)$ is the index of the maximal reward estimate, i.e. $d_{M(k)}=d_j(k) \mid j=1, 2, \ldots, r\}$ and μ is the internal parameter (a positive real number) determining the convergence rate of the algorithm.

$X_i(k+1)=X_i(k)+\beta(k)$ $\quad$ $n_i(k+1)=n_i(k)+1$

where i is the number of the solution of layout applied at period k.

$X_j(k+1)=X_j(k)$ for $j\neq i$ $\quad$ $n_j(k+1)=n_j(k)$ for $j\neq i$

$d_j(k+1)=X_j(k+1)/n_j(k+1)$ for $j=1, 2, \ldots, r$.

5.2 Evaluation of Scheduling Solutions

The "best" solution of layout is found according to the distribution of the action probabilities of the L.A., which vary with the response function $\beta(k)$. In our algorithm, β is generated by a scheduling evaluation criterion and defined as a function of the admissibility state of the corresponding scheduling solutions.

Assuming that the current solution of layout corresponds to the Flowshop $(M_1, M_2, ..., M_m)$ and to the set of items of production $\{J_j=(O_{j1}, O_{j2}, ..., O_{jm}) \mid 1 \le j \le n\}$. Each item J_j is characterized by the earliest starting time r_j and the latest finishing time d_j. Each operation O_{ji} is characterized by the processing time on machine M_i:

$p_{ji}=p_i$ if J_j is processed by M_i and

$p_{ji}=0$ otherwise

Denote t_{ji} and f_{ji} the real starting time and the real finishing time of the operation O_{ji}. A necessary and sufficient condition for successful scheduling is $f_{jm} \le d_j$ for $\forall j \in \{1, 2, ..., n\}$. For a given scheduling solution S, t_{ji} and f_{ji} can be recursively calculated as follows:

Initiation:

$t_{ji}=0$ for $\forall i \in \{1, 2, ..., m\}$ and $\forall j \in \{1, 2, ..., n\}$

$t_{j1} = r_j$ if O_{j1} is the first operation on M_1

$= max\{r_j, f_{s1}\}$ if O_{i1} is another operation on M_1

where O_{s1} is the operation preceded by O_{j1}

$t_{ji} = max\{r_j, f_{j,i-1}\}$ if O_{ji} is the first operation on M_i

$= max\{r_j, f_{j,i-1}, f_{si}\}$ if O_{ji} is another operation on M_i

where O_{si} is the operation preceded by O_{ji}

$f_{ji}=t_{ji}+p_i$ if J_j is processed by M_i

$= t_{ji}$ otherwise

The above calculation permits to minimize the left margins of processing intervals of O_{ji}'s for all machines M_i 's and all items J_j 's. Under this calculation, if there exist right free margins of processing intervals, i.e. the condition $f_{jm} \le d_j$ is satisfied for $j=1, ..., n$, then the corresponding scheduling solution S is admissible.

According to the scheme of L.A., at period k, a solution of layout is selected at random according to the distribution of the action probabilities $P(k)=(P_1(k)\ P_2(k)\ ...\ P_r(k))$ (r is the total number of existing solutions of layout). By applying the selected action to the generator of scheduling solutions, we get $S(k)=(S_1, ..., S_h, ..., S_m)$ where S_h is a sequence of operations processed on M_h, denoted by $(O_{1h} ... O_{q(h),h})$ with $h \in \{1, ..., m\}$ and $q(h) \le n$. On the machine M_h, a necessary condition of scheduling admissibility is $D_{ih} \ge t_{ih}+p_h$

where $D_{ih}=d_i - \sum_{l=h+1}^{m} p_{li}$

In general, the evaluation of a scheduling solution can be made according to the following three principles [19]:

1) the condition of admissibility is satisfied;

2) the free margins of processing intervals are large enough;

3) the loads on machines are sufficient.

In this paper, only the two first principles are taken into account by the evaluation criterion.

The first principle leads to the definition of a series of integers $N_{1h}, N_{2h}, ..., N_{q(h)h}$. Each N_{ih} corresponds to the operation O_{ih} with

$$N_{ih} = \begin{cases} 1 & \text{if } t_{ih} + p_{ih} \le D_{ih} \\ 0 & \text{otherwise} \end{cases}$$

Then, the number of operations in S_h satisfying the condition of admissibility is

$$Eval_1(S_h) = \frac{c}{q(h)} \sum_{i=1}^{q(h)} N_{ih}$$

where c is a constant on the interval $[0, 1]$ adjusting the proportion between $Eval_1(S_h)$ and $Eval_2(S_h)$.

For the operation O_{ih}, the left free margin is 0. The second principle leads to the calculation of the right free margin $D_{ih}-t_{ih}-p_{ih}$. In order to obtain appropriate input to the L.A., we evaluate the right free margin using a continuous function $f(.)$.

$$Eval_2(S_h) = \frac{1}{q(h)} \sum_{i=1}^{q(h)} f(D_{ih} - t_{ih} - p_h)$$

where $f(x)$ is an ascending continuous function symmetric with respect to the origin.

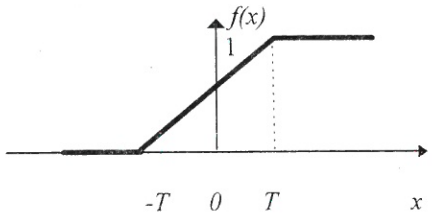


Figure 5. Evaluation Function $f(x)$

The response to the L.A. β is defined as the general evaluation on all the machines, i.e.

$$\beta(k)=Eval(S)=\frac{1}{2}\sum_{h=1}^{m} w_h[Eval_1(S_h)+Eval_2(S_h)]$$

w_h's are the weights adjusting the proportion between the evaluations on different machines ($w_h \in [0,1]$ and $\Sigma w_h=1$) and making the value of β be included in the interval $[0, 1]$.

If the value of β is close to 1, it means that the current solution of layout is good in the sense of scheduling admissibility. In this case, the probability of getting admissible scheduling solutions is high. If the value of β is close to 0, it means that the current layout solution is bad in the sense of scheduling admissibility. In this case, the probability of getting admissible solutions is low.

5.3 Generation of Scheduling Solutions

In Figure 4, the solution $S=(S_1\ S_2\ \ldots\ S_m)$ is generated at random by the generator of scheduling solutions. This solution is used to evaluate the scheduling behaviors of the corresponding solution of layout L. If the precedence constraints between items are not considered, S should be generated according to a uniform distribution, i.e. all the combinations of operations in the searching universe have the same probabilities. However, due to the multiplicity of these combinations, the searching universe is too large to find solutions reflecting the real behaviors of L. For example, let S^0 be the unique admissible solution corresponding to the Flowshop structure L. The generator can probably generate a series of S's far away from S^0. In this case, both the value of the response β and the action probability of L decrease. Another solution of layout may be selected and the admissible solution in the structure L can never be found.

From the previous analysis, we recognize that it is necessary to define a strategy for reducing the searching space of S by assigning different probabilities to different combinations of operations. The reduced space should permit to generate a small number of scheduling solutions with high probabilities. These solutions should be close enough to the admissible solution S^0. We give next this strategy on the machine M_h.

On the machine M_h, the searching space of scheduling solutions is based on the selected solutions on the machines M_1, ..., M_{h-1}. In this case, each operation O_{ih} is characterized by the interval $[R_{ih}, D_{ih}]$ and p_h with $R_{ih}=max\{r_i, f_{i,h-1}\}$. Let $A=[a_0, a_k]$ be the scheduling interval with $a_0=min\{R_{ih} \mid i=1, \ldots, q\}$ and $a_k=max\{D_{ih} \mid i=1, \ldots, q\}$. According to the idea of [16], we try to partition the interval A with a separator F_i. The costs (free margin loss cost and overflow cost) of O_{ih} separated by F_i are defined as follows:

1) When $R_{ih} \le F_i \le D_{ih}$, the left free margin loss cost $x1=(F_i-R_{ih})/(D_{ih}-R_{ih})$ and the left overflow cost $x2=(F_i-R_{ih})/p_h$. The right free margin loss cost $y1=(D_{ih}-F_i)/(D_{ih}-R_{ih})$ and the right overflow cost $y2=(D_{ih}-F_i)/p_h$.

2) When $F_i<R_{ih}$ or $F_i>D_{ih}$, the interval $[R_{ih}, D_{ih}]$ is not separated by F_i and $x1=x2=y1=y2=0$.

For the operation O_{ih}, the probability distribution of F_i can be defined from these costs, i.e.

$$\begin{aligned} P(O_{ih}, F_i) &= \omega x1 + \xi x2 \quad \text{when } F_i \in [R_{ih}, (R_{ih}+D_{ih})/2] \\ &= \omega y1 + \xi y2 \quad \text{when } F_i \in [(R_{ih}+D_{ih})/2, D_{ih}] \\ &= 0 \quad \text{when } F_i \in [a_0, R_{ih}] \text{ or } F_i \in [D_{ih}, a_k] \end{aligned}$$

where ω, ξ $(\omega+\xi=1)$ are positive real coefficients.

Under this definition, high probabilities are assigned to the values of F_i corresponding to big free margin loss and big overflow. Therefore, the probability distribution of F_i is centered on the processing interval of O_{ih} and the biggest probability corresponds to $F_i=(R_{ih}+D_{ih})/2$. A value F_i can be selected at random according to this distribution. For different O_{ih}'s, we can obtain different F_i's, which can be arranged in order, i.e. $F_1 \le F_2 \le \ldots\ldots \le F_{q(h)}$. This order permits to generate a scheduling solution $S=(O_{1h}\ O_{2h}\ \ldots\ldots\ O_{q(h),h})$. This solution is relatively close to the admissible solution S^0, because the precedence constraints between O_{ih}'s are taken into account in F_i's.

Remark: If $F_i=F_{i+1}=\ldots=F_j$, the order $(O_{ih}\ \ldots\ldots O_{jh})$ can be generated at random according to the uniform distribution.

We can further reduce the searching universe of S by imposing the following two constraints:

1) operations whose processing intervals are small (free margins are small) should be firstly processed;

2) operations whose earliest starting times are close to the starting time of M_h should be firstly processed.

For each operation O_{ih}, we define a real variable Z_{ih} taking into account these constraints:

$$Z_{ih}=(1-\lambda)\frac{D_{ih}-R_{ih}}{p_h}+\lambda\frac{R_{ih}-a_0}{p_h}$$

where λ is a random variable obeying the uniform distribution in $[0, 1]$.

Having arranged the Z_{ih}'s in an ascending order, we get $Z_{1h} \le Z_{2h} \le \ldots \le Z_{qh}$ and the scheduling solution $S_h=(O_{1h}\ O_{2h}\ \ldots\ O_{qh})$. Based on the solutions on the previous machines, S_h is generated by the random variable λ and its searching universe is largely reduced by this strategy.

6. Simulation

6.1 Convergence of the Algorithm

Theoretically, the algorithm of learning automaton presented in Section 5.1 converges to the optimal action with a probability high enough because the optimal solution of layout exists with respect to our criterion and then the environment is stationary. However, in practice, a stop condition should be given so that the algorithm stops in a reasonable duration.

In our simulation, this stop condition is defined by

$Cvg<\varepsilon$

where ε is the predefined precision threshold and $Cvg=\frac{1}{r}\sum_{i=1}^{r} V_i$, the average of the variances of all probabilities

with $V_j=\frac{1}{k}\sum_{i=1}^{k}(P_j(i+1)-P_j(i))^2$, variance of P_j.

V_j can be calculated with recurrence:

$$V_j(k)=\frac{k-1}{k}V_j(k\text{-}1)+\frac{1}{k}(P_j(k+1)-P_j(k))^2$$

According to our experiments, under this stop condition, we get the best action with minimal probability of error. Noises on P_j can be finally eliminated because V_j is calculated through k periods and $\{k\}$ is an increasing series. If the action probabilities become stable, Cvg will rapidly decrease.

6.2 Simulation Results

The proposed algorithm has been applied to several examples in order to perform an evaluation for a number of Flowshop structures. The simulation results of three examples are given in this Section. Example 1 illustrates the convergence behavior of the algorithm using different values of μ. In Example 2, a modification of the algorithm of L.A. is given. In Example 3, two equivalent optimal solutions of layout are processed.

In our simulation, the parameters of the algorithm are selected as follows:

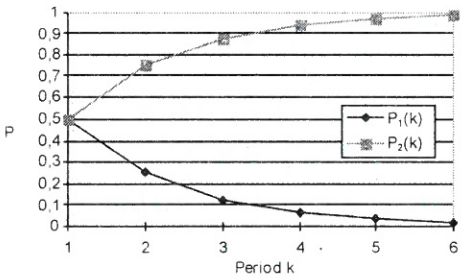
$T=3,\ c=0.5,\ \varepsilon=0.002.$

Example 1:

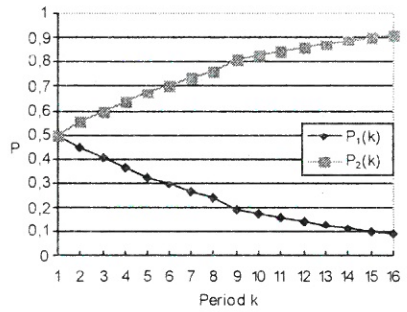
Assume that we have two items $W_1=ab$, $W_2=ba$, with $r_1=0$, $d_1=10$, $r_2=0$, $d_2=5$ and the processing times on a and b are $p_a=2$, $p_b=3$.

No solution of layout is generated from the exploration tree. By adding a machine of type a or a machine of type b, two solutions of layout: $L_1=aba$ and $L_2=bab$, can be obtained. Both the items and the solutions of layout are symmetric. However, the L.A. gives different action and reward probabilities to these two solutions of layout because of different scheduling parameters on L_1 and L_2.

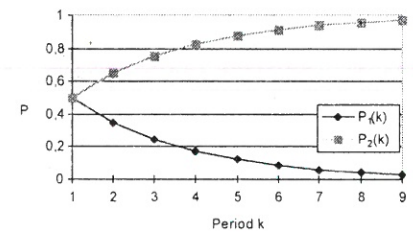
For different values of the internal parameter of the algorithm μ, we obtain



(a) μ=0.5



(b) μ=0.1



(c) μ=0.3

Figure 6. Evolution of Action Probabilities for Different Values of μ

It is shown in Figure 6 that the values of the parameter μ determine the convergence rate. In general, small values of μ lead to long duration of convergence but the convergence is more stable. And big values of μ lead to short duration of

convergence but some noises may appear during the convergence. In Example 1, the compromised value of μ is *0.3*. The evolution of the corresponding *Cvg* is shown as follows:

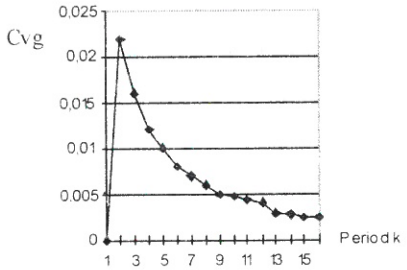
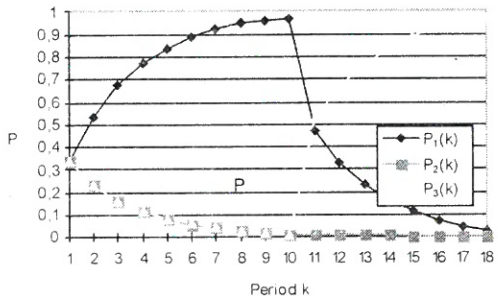


Figure 7. Evolution of *Cvg* (the Averaged Variance of All Action Probabilities)

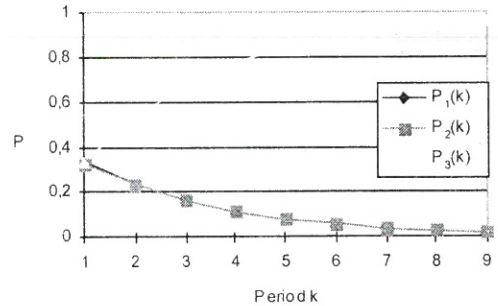
Example 2:

Assume that there exist *3* items $W_1=abc$, $W_2=acb$, $W_3=cab$. The scheduling parameters are the following: $p_a=5$, $p_b=2$, $p_c=1$ and $r_1=1$, $d_1=10$, $r_2=2$, $d_2=15$, $r_3=0$, $d_3=10$.

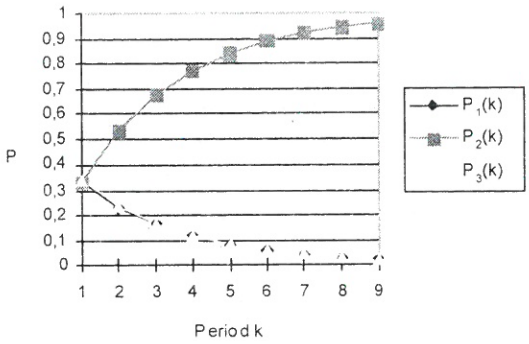
From the procedure of layout of production, we obtain 3 solutions of layout: $L_1=cabcb$, $L_2=cacbc$ and $L_3=acabc$. With these solutions of layout, a number of experiments on the proposed algorithm of L.A. have been done and we show the results of the three first experiments as follows.



(a) The First Experiment: the Duration of Convergence=*18* Periods



(b) The Second Experiment: the Duration of Convergence=9 Periods



(c) The Third Experiment: the Duration of Convergence=9 Periods

Figure 8. Evolution of Action Probabilities for 3 Different Experiments

In Example 2, different experiments lead to different results. In (a) and (b) of Figure 8 , $P_3(k)$ tends to *1* and L_3 is then selected. However, in (c) of Figure 8, $P_2(k)$ tends to *1* and L_2 is then selected. Having done *30* experiments on the algorithm, we obtain: the solution L_2 is selected *22* times and the solution L_3 is selected *8* times.

These results are not satisfying because some "false" optimal actions are selected by the algorithm of L.A. In this example, the best solution is L_3 instead of L_2 (see Section 6.3) but L_2 is more selected by the L.A. These results are due to the bad behavior of the estimates of the reward probabilities $D(k)$, i.e. some $d_i(k)$'s may be masked by some others until the stop condition of the algorithm is satisfied.

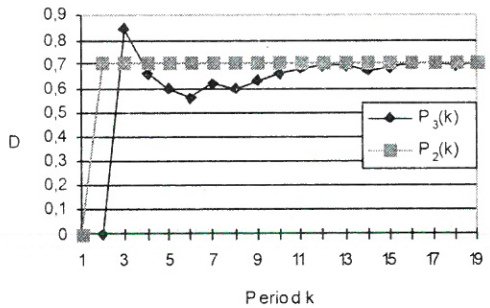


Figure 9. Evolution of the Reward Probabilities *D*

From Figure 9, we can see that the evolution of $d_2(k)$ is very variable and several local maxima can be obtained from the curve of $d_2(k)$. However, its value is never taken into account in the updating of the action probabilities $P_i(k)$'s because the value of $d_3(k)$ is always maximal and the value of $d_2(k)$ is then masked by $d_3(k)$ until the algorithm stops.

To solve this problem, we modify next the reinforcement scheme of the L.A. defined in Section 5.1.

The updating of the estimates of the reward probabilities:

$d_i(k+1)=X_i(k+1)/n_i(k+1)$ if α_i is applied at the period k

$d_j(k+1)=\delta \cdot d_j(k)$ for all $j \neq i$

where δ is the penalty coefficient with $0<\delta<1$.

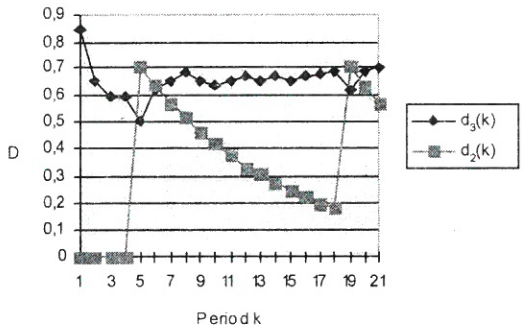


Figure 10. Evolution of the Reward Probabilities *D(k)* After the Modification *(δ=0.9)*

Under this modification, the maximal estimate of reward probabilities whose action has not been selected should be progressively penalised. In this example, both $d_2(k)$ and $d_3(k)$ are taken into account in the updating of the action probabilities. In *30* experiments after this modification, the correct solution L_2 was selected *28* times and L_3 was selected only *2* times.

This modification has been applied to all the examples of the simulation. According to our experiments, the uncertainty of selection of optimal action can largely be reduced by this modification.

In practice, the value of δ should be appropriately selected. Great values of δ make the convergence of the algorithm too long and small values of δ could lead to non stability of the algorithm.

Example 3:

Assume that there exist *4* items $W_1=abaac$, $W_2=bdcba$, $W_3=adabc$, $W_4=cbac$. The scheduling parameters are: $r_1=0$, $d_1=16$, $r_2=0$, $d_2=16$, $r_3=0$, $d_3=16$, $r_4=0$, $d_4=16$ and $p_a=2$, $p_b=3$, $p_c=4$, $p_d=5$.

From the procedure of layout of production, we obtain two solutions: $L_1=abdcabac$ and $L_2=abdacbac$. In the first experiment, the algorithm selects L_2 as optimal action through *21* periods. The result is given as follows:

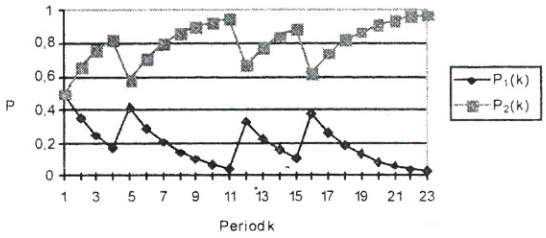


Figure 11. Evolution of The Action Probabilities

Having done a great number of experiments on the algorithm, we can observe that the number of selections of L_1 is almost equal to that of L_2. This is due to the fact that the solutions L_1 and L_2 are equivalent in the sense of scheduling admissibility (see Section 6.3).

6.3 Validation of the Results

The validation of the simulation results can be done using the scheduling algorithm presented in [8]. The objective of this algorithm is to find all scheduling solutions under the precedence constraints derived from scheduling parameters. Its application to the examples of Section 6.2 gives the following results:

Example 1:

Under the Flowshop structure $L_1=M_1M_2M_3$ ($M_1=a$, $M_2=b$ and $M_3=a$), W_1 and W_2 can be rewritten as: $W_1=(O_{11}\ O_{12})$ and $W_2=(O_{22}\ O_{23})$. From the scheduling algorithm, we obtain:

the unique admissible solution: (O_{11}) on M_1

$(O_{22}\ O_{12})$ on M_2

(O_{23}) on M_3

Under the Flowshop structure $L_2=M_1M_2M_3$ ($M_1=b$, $M_2=a$ and $M_3=b$), we have $W_1=(O_{12}\ O_{13})$ and $W_2=(O_{21}\ O_{22})$. Two admissible solutions can be obtained:

Solution 1:	Solution 2:
(O_{21}) on M_1	(O_{21}) on M_1
$(O_{12}\ O_{22})$ on M_2	$(O_{22}\ O_{12})$ on M_2
(O_{13}) on M_3	(O_{13}) on M_3

The solution of layout L_2 is better than L_1 in the sense of scheduling admissibility, which conforms to the results of Section 6.2.

Example 2:

Under the Flowshop structure $L_1=cabcb$ (*5* machines), we have $W_1=(O_{12}\ O_{13}\ O_{14})$, $W_2=(O_{22}\ O_{24}\ O_{25})$, $W_3=(O_{31}\ O_{32}\ O_{33})$. No admissible solution can be found.

Two best scheduling solutions can be obtained:

Solution 1:

M_1: (O_{31}) M_2: $(O_{12}\ O_{32}\ O_{22})$

M_3: $(O_{13}\ O_{33})$ M_4: $(O_{14}\ O_{24})$ M_5: (O_{25})

The total overflow $\sum_{i=1}^{m} N_i(d_i - f_i) = 7$

where N_i's have been defined in Section 5.2.

The number of non admissible items $\sum_{i=1}^{m} N_i = 2$

where f_i is the real finishing time of the item W_i.

Solution 2:

M_1: (O_{31}) M_2: $(O_{12}\ O_{22}\ O_{32})$

M_3: $(O_{13}\ O_{33})$ M_4: $(O_{14}\ O_{24})$ M_5: (O_{25})

The total overflow = 8

The number of non admissible items = 1

For L_2=*cacbc*, no admissible solution can be found. For the best scheduling solutions, we obtain the same total overflow and the same number of non admissible items.

For L_3=*acabc*, we have $W_1=(O_{11}O_{14}O_{15})$, $W_2=(O_{21}\ O_{22}\ O_{24})$, $W_3=(O_{32}\ O_{33}\ O_{34})$. One admissible scheduling solution can be found. We obtain

M_1: $(O_{11}\ O_{12})$ M_2: $(O_{32}\ O_{22})$ M_3: (O_{33})

M_4: $(O_{14}\ O_{34}\ O_{24})$ M_5: (O_{15})

with $f_1=9<d_1=10$, $f_2=14<d_2=15$ and $f_3=d_3=10$

Evidently, L_3 is the best solution of layout in the sense of scheduling admissibility, which also conforms to the results of Section 6.2.

Example 3

For L_1=*abdcabac* (8 machines), we have $W_1=(O_{11}\ O_{12}\ O_{15}\ O_{1}\text{-}\ O_{18})$, $W_2=(O_{22}\ O_{23}\ O_{24}\ O_{26}\ O_{2}\text{-})$, $W_3=(O_{31}\ O_{33}\ O_{35}\ O_{36}\ O_{38})$, $W_4=(O_{44}\ O_{46}\ O_{4}\text{-}\ O_{48})$,

No admissible solution can be found. One of the best scheduling solutions is

M_1: $(O_{11}\ O_{31})$ M_2: $(O_{22}\ O_{12})$ M_3: $(O_{23}\ O_{33})$

M_4: $(O_{24}\ O_{44})$ M_5: $(O_{15}\ O_{35})$ M_6: $(O_{46}\ O_{26}\ O_{36})$

M_7: $(O_{4}\text{-}\ O_{1}\text{-}\ O_{2}\text{-})$ M_8: $(O_{48}\ O_{18}\ O_{38})$

The total overflow = 8

The number of non admissible items = 3

For L_2=*abdacbac*, we have $W_1=(O_{11}\ O_{12}\ O_{14}\ O_{1}\text{-}\ O_{18})$, $W_2=(O_{22}\ O_{23}\ O_{25}\ O_{26}\ O_{2}\text{-})$, $W_3=(O_{31}\ O_{33}\ O_{34}\ O_{36}\ O_{38})$, $W_4=(O_{44}\ O_{46}\ O_{4}\text{-}\ O_{48})$

No admissible solution can be found. One of the best scheduling solutions is

M_1: $(O_{11}\ O_{31})$ M_2: $(O_{22}\ O_{12})$ M_3: $(O_{23}\ O_{33})$

M_4: $(O_{14}\ O_{34})$ M_5: $(O_{45}\ O_{25})$ M_6: $(O_{46}\ O_{26}\ O_{36})$

M_7: $(O_{4}\text{-}\ O_{1}\text{-}\ O_{2}\text{-})$ M_8: $(O_{48}\ O_{18}\ O_{38})$

The total overflow = 8

The number of non admissible items = 3

Therefore, the solutions of layout L_1 and L_2 are equivalent in the sense of scheduling admissibility, which also conforms to the results of Section 6.2.

7. Conclusion

The general scheme defined in Section 1 enables the partitioning of the items of production in a Jobshop structure into different classes so that the items belonging to each class should obey a Flowshop structure and admissible solution be obtained using a Flowshop scheduling algorithm.

In this paper, we discuss only two problems of this general scheme: 1) building Flowshop structures (solutions of layout of production) from a set of machines and a set of items; 2) evaluating these Flowshop structures using a learning automaton.

The solutions of layout are evaluated according to the criterion of scheduling. A L.A. is designed to find, according to the distribution of the action probabilities and the reward probabilities, the optimal solution of layout in the sense of scheduling admissibility. The selected solution of layout does not necessarily lead to an admissible scheduling solution but its probability of getting admissible scheduling solutions is the highest among all the solutions of layout. In general, the heavy calculation of the scheduling procedure can largely be reduced by this evaluation.

However, this evaluation procedure is significant only when there exists the unique optimal solution of layout and the environment of the L.A. is stationary. In this case, the convergence of the algorithm has been proved. If there exist several equivalent optimal solutions of layout, this evaluation is not significant.

This evaluation procedure can also be applied to the clustering operation of the scheme of Section 1, i.e. exchanging items between different classes so that admissible scheduling solution can be found in each class. In general, a clustering procedure is based on measures of similarity between items inside each class and of dissimilarity between different classes. In our problem, these two measures, which cannot be explicitly defined, depend on the procedure of layout of Section 3 and the scheduling behaviors. Therefore, each precise measure of similarity and dissimilarity should be obtained by running a

scheduling algorithm, which leads to very heavy calculation. For simplicity, our evaluation procedure of solutions of layout could be used to estimate the measures of similarity and dissimilarity.

This evaluation procedure can also be used in other fields for making rough evaluation and for simplifying the complexity of calculations.

REFERENCES

1. BUFF, E.S., ARMOUR, G.C. and VOLLMAN, T.E., **Allocating Facilities With CRAFT**, HARVARD BUSINESS REVIEW, 42, March/April, 1964.
2. CARRIE, A.S., **Computer-Aided Layout Planning – The Way Ahead**, INT J. PROD.RES., Vol.18, No.3,1980, pp.283-294.
3. CHARALAMBOUS, O. and HINDI, K.S., **A Knowledge-based Jobshop Scheduling System**, PRODUCTION PLANNING AND CONTROL, Vol.4, 1993, pp.304-312.
4. FISHER, M.L., LAGEWEG, B.J. , LENSTRA J.K. and RINNOY, K., **Surrogate Duality Relaxation for Jobshop Scheduling**, DISCRE. APPLIED MATH, No.5, 1983, pp. 65-75.
5. FORTENBERRY, J.C. and COX, J.F., **Multiple Criteria Approach To the Facilities Layout Problem**, INT. J. PROD. RES., Vol.23, No.4, 1985, pp. 773-782.
6. FU, K.S., **Syntactic Method in Pattern Recognition,** ACADEMIC PRESS, New York, 1974.
7. GAREY, M.R., JOHNSON, D.S. and SETHI, R., **The Complexity of Flowshop and Jobshop Scheduling**, Technical Report, No.168, Computer Science Dept., Pennsylvania State University.
8. HAPPIETTE, M. and ZENG, X., **Ordonnancement pour le probleme Flowshop**, REVUE FRANÇAISE APII, No.6, 1995.
9. HAPPIETTE, M. and STAROSWIECKI, M., **An Algorithm for the Detection of Order-Consistency On A Set of Syntactic Patterns: Application To Workshop Organisation**, SYSTEMS SCIENCE, Vol.16, No.3, 1990.
10. MICLET, L., **Méthodes structurelles pour la reconnaissance de formes**, ÉDITION EYROLLES, 1984.
11. WAGNER, R. A. and FISHER, M.J., **The String To String Correction Problem**, JOURNAL ACM, Vol.21, No.1, 1974, pp. 168-173.
12. ZENG, X. and VASSEUR, C., **Searching for the Best Partition By Clustering,** Uncertainty in Intelligence Systems, ELSEVIER SCIENCE PUBLISHERS, North-Holland, 1993.
13. ZENG, X., HAPPIETTE, M. and VASSEUR, C., **A Solution To the Scheduling Problem by Decomposing Jobshop into Flowshop Structures**, Proceedings of the 12th International Conference on Systems Science, Wroclaw, Poland, September 1995, pp. 301-308.
14. NARENDRA, K.S. and THATHACHAR, M. A. L., **Learning Automata A Survey**, IEEE TRANS. ON SYSTEMS, MAN, CYBERNETICS, Vol.14, 1974, pp. 323-334.
15. NARENDRA, K. S. and THATHACHAR, M. A. L., **Learning Automata: An Introduction**, PRENTICE HALL, Englewood Cliffs, NJ, 1989.
16. HAPPIETTE, M. and ZENG, X., **Temporal Decomposition By Hierarchical Procedure for Scheduling**, Proceedings of the 2nd IFAC/IFIP/IFORS Workshop on Intelligent Manufacturing Systems, Vienna, Austria, June 1994, pp. 261-266.
17. ZENG, X., HAPPIETTE, M. and VASSEUR, C., **Decomposition of Jobshop into Flowshop Structures for Scheduling**, Proceedings of the International Conference IPMU'96, Granada, Spain, July 1996.
18. OOMMEN, B.J. and LANCTOT, J. K., **Discretized Pursuit Learning Automata**, IEEE TRANS. ON SYSTEMS, MAN, CYBERNETICS, Vol.20, 1990, pp. 931-938.
19. BLAZEWICZ, J., ECKER, K.H., SCHMIDT, G. and WEGLARZ, J., **Scheduling in Computer and Manufacturing Systems**, SPRINGER VERLAG, Berlin, 1994.