8

Analytic Process of State and Linear Combined Priority

Wu Xiao-Bing
Faculty of Engineering; University of Passo Fundo
99001.970 - Passo Fundo - RS - Brazil
e-mail: wu@vitoria.upf.tche.br

Abstract

Analytic Process of State (APS) is a new approach to deal with the complex discrete processes in the real world, such as the production process in a manufacturing system. It decomposes a complex process into a series of simple processes through a concept known as "basic entity". These simple processes are solved with an analysis of process state. In the problem-solving process, APS just concentrates on the points in which some decision is necessary for the resolution, known as decision points. In this paper, a linear combined priority model is introduced and applied to make decisions in the decision points. A local search algorithm is proposed to build such a priority model. An example of application of APS with combined priority to a flexible manufacturing system scheduling problem is presented.

Keywords

Analytic process of state, priority model, local search algorithm, FMS, scheduling.

1 INTRODUCTION

Analytic Process of State (APS) is an approach proposed recently to solve the complex discrete processes in practice, such as the production process in a manufacturing system (Wu 1993, 1995; Wu and Armentano 1994). It decomposes a complex process into a series of simple processes. These simple processes are solved with an analysis of process state, and the solving process is focused on the points in which some decisions are required to make, known as decision points. In this way, APS provides a simple way to deal with the difficult problems, especially the practical ones in industry.

The development of APS is based on the necessity of resolution of large scale production scheduling problems in a practical manufacturing system. Initially, a Multi-Objetive Linear Combined Priority (MOLCP) model presented in Wu (1987) is used to make decision in the problem-solving process with APS (Wu 1993, 1995, Wu and Armentano, 1994). A method based on Analytic Hierarchy Process (Saaty 1980) is developed to build an MOLCP model, which combines the different factors associated with conventional production scheduling rules

L. M. Camarinha-Matos et al. (eds.), *Balanced Automation Systems II*

and deal with the multiobjective problems in a simple way. More detail about the conventional scheduling rules can be found in Panwalkar and Iskander (1977). The application of dispatch rules to solve the flexible manufacturing system scheduling problems can be found in Stecke and Solberg (1981), Montazeri and Van Wassenhove (1990).

In this paper, we give a simplified introduction of APS, provided a linear combined priority model for the decision-making in APS, and develop a local search algorithm to build such a priority model. An example of application of APS and the combined priority model in FMS scheduling is presented. The simulation results are analyzed.

2 ANALYTIC PROCESS OF STATE

2.1. Basic concepts

Basic entity: denotes the smallest individual entity, or "moving particle", in a system which moves through the system and causes its state changes. It may be real or fictitious.
Process: denotes the whole moving sequence of operations or activities with which the basic entities move through a system.
State: denotes a special situation or period of a basic entity in which some properties or attributes of a basic entity are not changed.
Transition of state: denotes a change of state from one to another.

2.2. Classification of state and transition

According to the viewpoint of time and function, the states of a basic entity can be classified into the following types:
a). Initial state: a state to denote the beginning of a process. It is the first state of a basic entity, and no transition from another state to it will happen.
b). Control state: a state used for some logical control function in a process. The time spent in this state is zero.
c). Determinate timed state: a state which is timed, and the staying time of a basic entity in the state is predefined.
d). Indeterminate timed state: a state which is timed, and the staying time of a basic entity in the state is not predefined.
e). End state: a state to denote the end of a process. This is the last state of a basic entity, and no transition from it to any other state will happen.
f). Subset state: a state composed of some simple state types described above. This state is used to simplify the model for some complex or independent processes.

Based on the resource viewpoint, the states can be classified into the following types:
a). Non-resource state: a state in which a basic entity does not require resources.
b). Resource selecting state: a state in which a basic entity has more than one available resource to select for its necessity.
c). Resource waiting state: a state in which a basic entity is competing with other basic entities for the same resource.
d). Resource processing state: a state in which a basic entity is being processed on the required resource.

Transitions from one state to another one are very important in a process. They can be classified into two basic kinds:

a) Direct transition: a transition which is realized independently of any decision making.
b) Indirect transition: a transition which is dependent on some decision making.

2.3. Description of APS

States in APS can be represented by the following symbols.

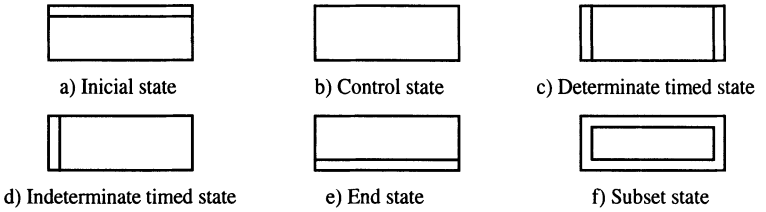

Figure 1. Symbols of states

If a state is also a resource waiting state, we add three small circles at the left of its symbols. If a state is also a resource selecting state, we add three small circles at the right of its symbols. Some examples are shown in Figure 2.

Figure 2. Examples of resource related states

The symbols of transitions are as follows.

Figure 3. Symbols of transitions

2.4. Properties of APS

Definition 2.4.1 : given a finite set of states, Σ , and a finite set of transition T, if $\alpha, \beta \in \Sigma$, $t \in T$, and t is the transition from α to β, then we say that t changes state from α to β. It can be written as: t: $\alpha \rightarrow \beta$; α is called the input state of transition t, and β is called the output state of transition t.
Definition 2.4.2 : given a transition $t \in T$, and states $\alpha, \beta \in \Sigma$; if: t: $\alpha \rightarrow \beta$, and the transition t is a direct transition, then the point at which the transition t happens in a process is called a No Decision-Making Point (NDMP) of the process.

Definition 2.4.3 : given a transition $t \in T$, and states $\alpha, \beta \in \Sigma$, if: $t: \alpha \rightarrow \beta$, and the transition t is an indirect transition, then the point at which the transition t happens in a process is called a Decision-Making Point (DMP) of the process.
Definition 2.4.4 : a transition is called a defined transition if it is a no decision-making point, or it is a decision-defined decision-making point.
Definition 2.4.5 : a process is called Defined Process (DP) if all the transitions in it are defined transitions.
Definition 2.4.6 : given an objective set $O = \{ O_i / i = 1, ..., n \}$, a process is called a multiobjective defined process based on O if it is a defined process and can satisfy O.

Property 2.4.1: a transition is direct or indirect according to its input state.
Property 2.4.2: if the input state of a transition is a control, and a non-resource or a resource processing state, then the transition is a direct transition.
Property 2.4.3: if the input state of a transition is a determinate timed, and a non-resource or resource processing state, then the transition is a direct transition.
Property 2.4.4: if the input state of a transition is a resource waiting state, then the transition is an indirect transition.
Property 2.4.5: if the input state of a transition is a resource selecting state, then the transition is an indirect transition.

2.5. The procedure to solve a problem with APS

The basic idea of APS to solve a problem is to change the DMPs into the decision-defined points in order to get a defined process, and then, to simulate it in order to get a desired solution. One of the key points is to build a decision-making model and to change the DMPs into the defined points directly and uniquely in a multiobjective way, if a multiobjective solution is desired. The detailed procedure is shown as follows:
1). Define the problem, the basic entity and its process;
2). Define the states and transitions in the process;
3). Analyze each possible state transition in order to recognize the NDMPs and the DMPs;
4). Build up a priority model for each DMP in order to change it into a defined point;
5). Simulate the process and get the desired solution.

3 LINEAR COMBINED PRIORITY

Dispatch rule is a rapid, simple and practical methodology to solve a production scheduling problem, especially for the practical manufacturing systems. But scheduling rules are very sensitive to the variation of manufacturing systems, such as the layout, the production process, as well as the scheduling objectives. It is a well known fact that no a single scheduling rule is discovered to be suitable to produce a good schedule for every case. To combine the advantages of various simple rules in order to form a good priority model and produce a good schedule for any given system or objective, a linear combined priority model is proposed. A general form of a combined priority model can be described as follows:

$$P = \alpha_1 F_1 + \alpha_2 F_2 + ... + \alpha_n F_n \qquad (1)$$

where α_i (i = 1, 2, ..., *n*) and F_i (i = 1, 2, ..., *n*) are combinatorial parameters and the factors associated with a scheduling rule i correspondingly. For example, if rule 1 is SPT (Shortest Processing Time), F_1 represents the processing time of a part on a machine. *P* is a linear combined priority model associated the simple production scheduling rules 1, 2, ... , *n*.

In this way, the priority model *P* considers various factors when it is used to define a priority value, in order to make a suitable decision and produce a better schedule. To build a priority model *P*, the key point is to determine the combinatorial parameters α_i (i = 1, 2, ..., *n*).

4 LOCAL SEARCH ALGORITHM

To build a combined priority model defined in (1), we have to search a better combination of the scheduling rules 1, 2, ..., *n*. In another word, we need to search a better combinatorial value for each parameter α_i (i = 1, 2, ..., *n*). For this motivation, we develop a local search algorithm to get the desired value of these combinatorial parameters.

The basic idea of this algorithm is that from an initial priority model *P0*, with initial values of the combinatorial parameters $\alpha_i(0)$ (i = 1, 2, ..., *n*), we search these parameters in their neighbor to find out a better combination according to the defined criteria. The search process is carried out with a "par search", that is, for each time, we take two factors to find a best combination when the other factors are fixed; then we search another par of factors which is not considered, until all of the parameters are searched. The best combination defines the desired priority model.

To simplify the explication of algorithm, we consider initially one priority model with two factors:

$$P = \alpha_1 F_1 + \alpha_2 F_2 \tag{2}$$

The following notation is utilized in the description of the algorithm :

z(P) : criteria utilized to measure a schedule produced by priority model P
τ : number of search points in each direction
δ : initial search step, $0 < \delta, \tau\,\delta < \min(\alpha_1, \alpha_2),\ \tau\,\delta < 1 - \max(\alpha_1, \alpha_2)$
δ_m : minimum search step
λ : reducing percent of step δ
k : index of iteration
S_0 : list of non-searched elements (α_1, α_2, δ, z)
S_1 : list of searching element
S_2 : list of local minimum solution found in a neighbor of the searching points

Algorithm I

Step 1. Let $P_0 = P$, $z_0 = z(P_0)$ e $\delta_0 = \delta$. Put the element (α_1, α_2, δ_0, z_0) in a list S_0.

Step 2. If S_0 is empty, go to step 10.
If not, remove an element ($\alpha_{1,s}$, $\alpha_{2,s}$, δ_s, z_s) $\in S_0$ from the list S_0 and put it in a list S_1. Let k = 0, $z^* = z_s$, $\alpha_1^* = \alpha_{1,k} = \alpha_{1,s}$, $\alpha_2^* = \alpha_{2,k} = \alpha_{2,s}$, $\delta^* = \delta_k = \delta_s$

Step 3. Define $\alpha_{1,k+1}$, $\alpha_{2,k+1}$ and build a new model P_{k+1} with the following way:

$$\alpha_{1,k+1} = \alpha_{1,k} + \delta_k$$

$$\alpha_{2,k+1} = \alpha_{2,k} - \delta_k$$
$$P_{k+1} = \alpha_{1,k+1} F_1 + \alpha_{2,k+1} F_2$$
$$\delta_{k+1} = \delta_k$$

Evaluate the performance of the new schedule and let $z_{k+1} = z(P_{k+1})$.

If $z_{k+1} < z^*$, let $z^* = z_{k+1}$, $\alpha_1^* = \alpha_{1,k+1}$, $\alpha_2^* = \alpha_{2,k+1}$, $\delta^* = \delta_{k+1}$.

Step 4. $k = k + 1$.

If $k < \tau$, go to step 3.

Step 5. $\delta^* = \lambda \delta^*$. Se $\delta^* > \delta_m$, put $(\alpha_1^*, \alpha_2^*, \delta^*, z^*)$ in list S_0.

If not, put $(\alpha_1^*, \alpha_2^*, \delta^*, z^*)$ in list S_2.

Step 6: Remove the element $(\alpha_{1,s}, \alpha_{2,s}, \delta_s, z_s) \in S_1$ from the list S_1.

Let $k = 0$, $z^* = z_s$, $\alpha_1^* = \alpha_{1,k} = \alpha_{1,s}$, $\alpha_2^* = \alpha_{2,k} = \alpha_{2,s}$, $\delta^* = \delta_k = \delta_s$

Step 7. Define $\alpha_{1,k+1}$, $\alpha_{2,k+1}$ and build a new model P_{k+1} in the following way:

$$\alpha_{1,k+1} = \alpha_{1,k} - \delta_k$$
$$\alpha_{2,k+1} = \alpha_{2,k} + \delta_k$$
$$P_{k+1} = \alpha_{1,k+1} F_1 + \alpha_{2,k+1} F_2$$
$$\delta_{k+1} = \delta_k$$

Evaluate the performance of the new schedule and let $z_{k+1} = z(P_{k+1})$.

If $z_{k+1} < z^*$, $z^* = z_{k+1}$, $\alpha_1^* = \alpha_{1,k+1}$, $\alpha_2^* = \alpha_{2,k+1}$, $\delta^* = \delta_{k+1}$

Step 8. $k = k + 1$.

If $k < \tau_e$, go to step 7.

Step 9. $\delta^* = \lambda \delta^*$. If $\delta^* > \delta_m$, put $(\alpha_1^*, \alpha_2^*, \delta^*, z^*)$ in list S_0.

If not, put $(\alpha_1^*, \alpha_2^*, \delta^*, z^*)$ in list S_2.

Go to Step 2.

Step 10. Select the element $(\alpha_1^*, \alpha_2^*, \delta^*, z^*) \in S_2$ with $z^* \leq z$ for any $(\alpha_1, \alpha_2, \delta, z) \in S_2$.

Then, α_1^*, α_2^*, represent the better parameters of model P in the searching neighbor.

The algorithm searches the points in two different directions, initialized with a known value of parameter α_1. Firstly, the search is done in the right, then in the left. The best points found in both directions serve as the new starting points for further search, with reduced search step, until the limit of minimum search step. When it happens, we get a local minimum solution and put it in list S_2. The best local minimum solution we found in list S_2 corresponds the final result.

In this algorithm, step 1 defines the initial conditions. In the step 2, we remove a non-searched point, $(\alpha_1, \alpha_2, \delta_d, z)$, from list S_0 and put it in list S_1 for further exploring. Step 3 and 4 search the points in the right of α_1 (left of α_2) and try to find a better point. Step 5 reduces the search step and compares it with a known minimum valor. If it is bigger, we put the best point found in step 3 and 4 in the list S_0 for further search. If not, this point is local minimum we found in the neighbor of the initial point and is put in list S_2. This way, we complete the searching process in the right of α_1 and start the searching process in the left of α_1 , which is described in step 6 to step 9. Step 10 selects the best local minimum solution we found to form a better priority model with two factors.

The algorithm I can be generalized to a case with n factors (see equation 1) in the following way.

Algorithm II

Step 1. Given a factor set, $F = \{F_1, F_2, ..., F_n\}$, a set of parameters $\alpha = \{\alpha_1, \alpha_2, ..., \alpha_n\}$, an empty list S, criteria z, the maximum number of iteration M and search step δ. Let $k = 0$, $P_0 = P$, and $\alpha_{i,0} = \alpha_i$.

Step 2. Select a par of factors F_i, $F_j \in F$ and $(F_i, F_j) \notin S$. If it does not exist, end.

Step 3. Utilize the algorithm I to get the best parameter α_i^*, α_j^*, associated with the factors F_i, F_j, in their neighbor. Define $\alpha_{r,k+1}$ ($r = 1, 2, ..., n$) and build a new model P_{k+1} in the following way:

$$\alpha_{i,k+1} = \alpha_i^*$$
$$\alpha_{j,k+1} = \alpha_j^*$$
$$\alpha_{r,k+1} = \alpha_{r,k} \ (r \neq i, r \neq j)$$
$$P_{k+1} = \sum_{i=1}^{n} \alpha_{i,k+1} \times F_i$$

Step 4. Add the par of factors (F_i, F_j) in list S. Let $k = k + 1$.
If $k < M$, go to step 2
If not, end.

It is noted that the par searching way simplifies the problem-solving process and avoids a very complex algorithm to search all the parameters simultaneously.

5 SOLVING FMS SCHEDULING WITH APS

5.1 Production scheduling problem

In a manufacturing system, production scheduling involves all the elements associated with a production process, principally the jobs and machines. Its main objective is to allocate the jobs to the machines in a determined time period in order to satisfy some desired criteria.

The main difference of scheduling between a conventional production system and a flexible manufacturing system (FMS) is caused by the flexibility and automatization in a modern manufacturing environment. In an FMS, one machine can process various different parts, and one operation can be processed in various different machines. These flexibilities increase significantly the feasible schedule space, complicate the FMS scheduling. This paper will discuss FMS scheduling problem under various alterable production routes.

5.2 Operation States

In a flexible manufacturing system, we consider a production operation as a basic entity. The production process of an operation can be divided into the following states: releasing state (RLS), part routing state (RTG), available state (AV), waiting state (WTG), processing state (PRC) and completion state (CPT).

1). Releasing state (RLS) : it represents the state in which an operation is released to a manufacturing system. It is terminated as soon as its all preceding operations are completed.

2). Part routing state (RTG) : it indicates the decision process which chooses one workcenter, or processing route, for an operation from the available candidates.

3). Available state (AV) : it denotes the state in which the part associated with an operation in this state needs a constant interprocessing time to be available in its desired workcenter.

4). Waiting state (WTG): it indicates the state in which an operation is waiting to be selected for machining.
5). Processing state (PRC) : it represents the state in which an operation is being processed.
6). Completion state (CPT) : it denotes the state in which an operation is completed.

5.3. Analysis and modeling

State RLS: there are two kinds of operations in this state: a) first operation of a part; b) non-first operation of a part. In the first case, the operation will enter its next state directly, i.e., no decision is needed. In the second case, the operation will leave the state RLS as soon as its immediate preceding operations are finished; it does not depend on any decision. Therefore the transition with state RLS as input state is a direct transition.
State RTG: in an FMS, there may be many workcenters which can process an operation. A workcenter, or a process route, is chosen for such an operation. It is clear this state is a control and resource selecting state. A transition with this state as an input state will be an indirect one. In order to maximize the utilization of machines, the least loaded workcenter is chosen.
State AV: the part associated with an operation in this state will need a constant time to be ready in the desired workcenter. After this time, the operation will leave this state. So this is a determinate timed and non-resource state. A transition with this state as an input state will be a direct transition.
State WTG: in this state, an operation is competing with other operations for a workcenter. The selection of operation will depend on some decision. So this state is an indeterminate timed and resource waiting state. Any transition with this state as an input state will be an indirect transition. A linear combined priority model is utilized to define this indirect transition (see the next section).
State PRC: an operation needs a constant time for its machining process in a workcenter. It leaves the state as soon as it is completed. So the state is a determinate timed and resource processing state. Any transition with it as an input state will be a direct transition.
State CPT: this is the last state an operation enters. So it is an end state; no transition will happen from it to any other state.

According to this analysis, we get our APS model for the FMS scheduling problem which is shown in Figure 4.

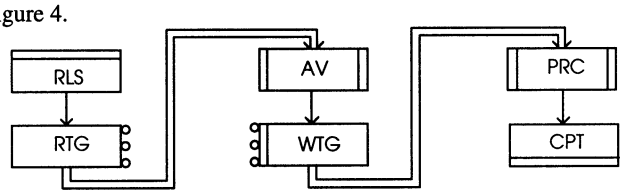

Figure 4. The APS model for FMS scheduling

6 IMPLEMENTATION AND ANALYSIS OF RESULTS

To schedule a flexible manufacturing system, a simulation software SFMS (Scheduler for Flexible Manufacturing Systems) is developed and implemented in SUN workstation (SPARC 10). Simulation program is written in C. With this software, we test a lot of examples with 100 parts (about 500 operations) and 7 workcenters. Each operation of a part can be processed

in until three alterable machining centers. Due date of each part is defined as 7 times of its processing time. The interesting scheduling objectives are : minimize the total delay time of parts (O_1) and minimize the makespan (O_2). A multiobjective index (MI) on these criteria is defined by analytic hierarchy process (Saaty 1980) :

$$MI = 0.75\ O_1 + 0.25\ O_2 \tag{3}$$

In order to select an operation in state WTG and define the indirect transition from WTG to PRC, we utilize a linear combined priority model. This model considers the following elements: modified due date (F_1), earliest due date (F_2), remaining work (F_3), processing time (F_4) and latest completion time (F_5)

To apply the local search algorithm to build a linear combined priority, a initial linear combined priority model $P0$ is constructed according to the building principles of MOLCP (Multi-Objective Linear Combined Priority) model (see Wu 1995, 1987), and presented as follow:

$$P0 = 0.3476\ F_1 + 0.\ 1263\ F_2 + 0.0786\ F_3 + 0.0542\ F_4 + 0.0564\ F_5 \tag{4}$$

In the follows, we utilize the local search algorithm in two way. Firstly, we build a priority model with single objective (tardiness), that is, to get a schedule with better performance in tardiness; and secondly, we build a priority with multiobjective, that is, to improve the multi-objective index MI of scheduling. The minimum search step for both cases is 0.0001. To get a reasonable result, 20 independent simulations are run and the mean is taken as a final result.

The Table 1 shows the tardiness of the initial model P0, the best simple scheduling rule MDD and the algorithm, we call it $P0^*$. It is noted the algorithm gets a significant improvement. It reduces 14,1% in the tardiness of $P0$ and 10,71% of MDD. It gets also the better results in makespan and multiobjective index.

The Table 2 shows the multiobjective index of the initial priority model $P0$, the algorithm (known as $P0^{**}$) and the best simple scheduling rule for this multiobjective index, that is, EDD. It is observed that the algorithm improves significantly the multiobjective index. It reduces 18,26% in MI of the initial priority $P0$ and 13,2% of the rule EDD. The reduction of MI causes a increase of tardiness and reduction of makespan.

The Figures 5 and 6 show the values of tardiness and multiobjective index we found during the searching processing. The computation cost to get a better solution depends on the points we searched with algorithm. Each point implicates a resolution of the scheduling problem. The Table 3 shows the values of these parameters of the models $P0$, $P0^*$ e $P0^{**}$.

Table 1 Tardiness

priority	Tardi-ness	Make-span	Index MI	Reduction (%)
P0	2823	4217	2317	14,1%
*P0**	2425	4216	2286	-
MDD	2716	4238	2522	10,71%

Table 2 Index MI

priority	Tardi-ness	Make-span	Index MI	Reduction (%)
P0	2823	4217	2317	18,26%
*P0**	2998	4174	1894	-
EDD	2730	4204	2182	13,2%

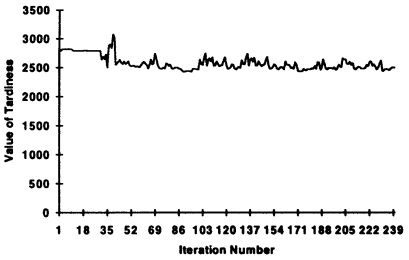

Figure 5 Tardiness

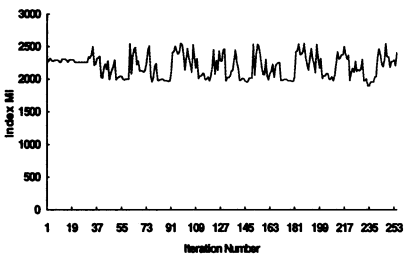

Figure 6 Index MI

Table 3 Variation of parameters

	α_1	α_2	α_3	α_4	α_5
P0	0,3476	0,1263	0,0786	0,0542	0,0564
*P0**	0,4609	0,1063	0,0289	0,0542	0,0128
*P0***	0,2446	0,1763	0,1291	0,0542	0,0589

7 REFERENCES

Montazeri, M. and Van Wassenhove, L. N. (1990) Analysis of scheduling rules for an FMS. *International Journal of Production Research*, **28**, 785-802.

Panwalkar, S. S. and Iskander, W. (1977) A survey of scheduling rules. *Operations Research*, **25**, 45-61.

Saaty, T. L. (1980) *The Analytic Hierarchy process*. McGraw-Hill.

Stecke, K. E. and Solberg, J. J. (1981) Loading and control policies for a flexible manufacturing system. *International Journal of Production Research*, **19**, 481-490.

Wu, X.-B. (1995) *Programação da Produção e do Transporte em Sistemas Flexíveis de Manufatura*. Ph.D. Dissertation, UNICAMP-Campinas-SP-Brasil.

Wu, X.-B. (1993) Analytic process of state and simulation. *Proceedings of XIX Latin-American Conference on Information*, Buenos Aires, Argentina, Aug. 2-6, 1993.

Wu, X.-B. (1987) Multiobjective job shop scheduling with finite capacity. in *Modern Production Management System*, (ed. A. Kusiak), Elsvier Science Publishers B. V.

Wu, X.-B. and Armentano, V. A. (1994) Analytic process of state and its application in FMS scheduling. in *Production Management Methods*, (eds. C. Walter, F. J. Kliemann and J.P.M. de Oliveira), Elsvier Science Publishers B. V.

8 BIOGRAPHY

Mr. Wu got his bachelor degree in Computer Engineering, Master and Doctor degree in Industrial Automation. Now, he is working for the Faculty of Engineering in University of Passo Fundo. His research interests are mainly in production scheduling, modeling and simulation methodologies in a manufacturing environment.