RECURRENT SCHEDULING.

Mag.scient., dr.oecon. Kim Andersen
Professor in Computer Based Control Models
Department of Production, University of Aalborg
Fibigerstraede 16, DK-9220 Aalborg Ø, Denmark.
Tel.: +45 9635 8969 • Fax: +45 9815 3030 • E-mail: i9ka@iprod.auc.dk

Abstract: Making a schedule normally is a recurrent problem as the conditions in the actual surroundings continually are subject to changes. The planned process times may be exceeded which causes a delay in the already planned schedule, there may be some changes in the available capacities of the involve resources, there may be some modifications in the present jobs, or there may be an addition of a new job which either have to be included in the schedule or for which in connection with making an offer on this new job we have to find the earliest possible due-date. Using an analytical method you normally find just one feasible solution, if any at all, feasible with respect to the resource constraints in question. However in some cases it will be impossible to find such a feasible solution even though feasible solutions exist. Whereas using a "trial and error"-like simulation procedure you usually may find many feasible solutions, if possible at all, among which, you can find the best one subject to a given objective function. The actual objective function may for instance be the makespan for all the jobs, the accumulate penalty for possible tardiness, inventory costs, change-over costs or possible due-date for a new job.

1. INTRODUCTION

A job-shop, consisting of a set, A, of N different activities

(1.1) $A = \{a_1, a_2, \ldots, a_N\}$

where the activities may come from Q different project

(1.2) $P = \{p_1, p_2, \ldots, p_Q\}$

each consisting of a certain subset of the activities, such that for i = 1, 2, ... , Q

(1.3) $p_i \subset \{A \mid p_i \cap p_j = \varnothing \text{ for } i \neq j\}$

to be performed on a set, R, of M different resources

(1.4) $R = \{r_1, r_2, \ldots, r_M\}$

may in many cases be considered as a *directed open network*, where the nodes in the network represent the activities and the arcs represent the precedence relationship among the activities.

A *closed network* is a network with just one start node as well as just one end node. Otherwise the network is an *open network*.

A network is called a *connected network*, if there is a path from every activity to every other activity in the network. In this case, there is just one project in P in (1.2). Otherwise the network is called an *unconnected network*, and P in (1.2) consists of more than one project.

An activity, y, is said to *precede* an activity, x, if y have to be finished before x can start, and we will use the notation

(1.5) $y < x$

In a similar way the activity x is said to *succeed* the activity y. The activity y is called a *predecessor* for activity x, and activity x is called a *successor* for activity y.

An activity, y, if said to *directly-precede* an activity, x, if $y < x$, and there is no activity, z, such that $y < z < x$. In this case we will use the notation

(1.6) $y <= x$

The activity y is called a *direct predecessor* for activity x, and activity x is called a *direct successor* for activity y.

The only arcs strictly necessary, are the arcs between activities, where one of the activities directly precede the other. However even though this is not the case, it is not illegal to include arcs between two activities, as long as a precedence relation between the activities in question exists. The final results will be the same, except the computations possibly will be some more time consuming.

Each activity must be given a unique number. It will be convenient, if the numbers are assigned, such that all activities has a number greater than the numbers for its possible predecessors and less than the numbers for its successors.

Example 1.1

In figure 1.1 is shown a project with 4 activities as an open network.

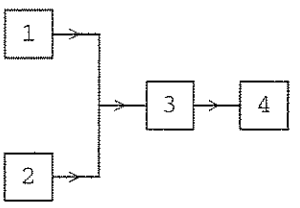

figure 1.1

Activity no. 4 has 3 predecessors, activity no. 1, 2 and 3, but just 1 direct predecessor, activity no. 3. In a similar way activity no. 1 has 2 successors, activity no. 3 and 4, but just 1 direct successor, activity no. 3.

2. SCHEDULING

In connection with the scheduling there may be two different kind of *boundary conditions* for the activities to take into consideration: some possible *arrival time/earliest start time constraints* respectively some possible *due date constraints*. Furthermore we normally have some *resource restrictions* concerning the resources in (1.6). These restrictions tell how many activities a resource can handle at the same time.

Altogether we may have the following restrictions

(2.1) **RESTRICTIONS**

1) Arrival times/earliest start times.
2) Due dates.
3) Resource restrictions.

(2.2) **REMARK NO. 1.1**

In many cases it is impossible simultaneously on the same project to execute more than one activity. If that is the case, it may be appropriate to regard the project as a resource, and in that way assign a resource number to the project.

We will say, a schedule is *feasible* if all the restrictions in (2.1) are fulfilled.

Normally there will exist more than one feasible schedule. Therefore it will be useful, if it is possible to find an *optimal schedule*, optimal with respect to a given *objective function*.

As usable objective functions can for instance be suggested

(2.3) **OBJECTIVE FUNCTIONS.**

1) The total inventory investment.
2) The total change-over costs.
3) Least possible makespan.
4) Penalties for lateness with respect to possible due-dates.
5) Earliest completion time for a certain project, e.g. for a new project.

In many cases a practicable way of making a schedule is by use of an iterative procedure, which for instance can be a *stepwise scheduling*, where after step number k we have scheduled k of the activities

(2.4)

$SCHEDULED_k$ = { Set of scheduled activities }

such that

(2.5) $SCHEDULED_k \subset A$

The *non scheduled* activities before step k + 1 can be classify in 2 different groups, the activities which are *possible* to activate

(2.6) $POSSIBLE_{k+1} \subset A \setminus SCHEDULED_k$

and by that the *non possible* activities.

For the next activity to schedule in step number k + 1 we then have

(2.7) $NEXT_{k+1} \in POSSIBLE_{k+1}$

The three different classes in which the activities can be classified are shown in figure 2.1.

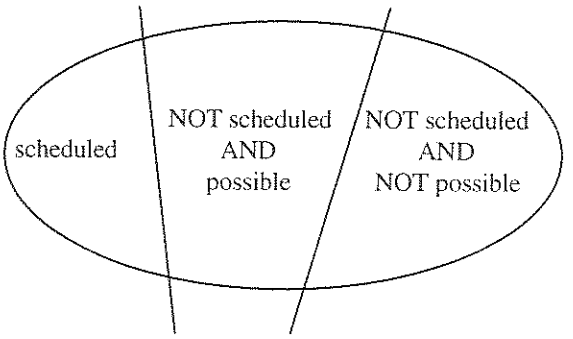

figure 2.1

The main problem may now be how to make the choice of the next activity to schedule among the possible activities. Dependent on the used scheduling procedure, the criterion, for which of the possible activities we will choose, may for instance be

(2.8) **CRITERION FOR SELECTION.**

1) Earliest due-date for the project in question.
2) Least remaining processing time for the project in question.
3) Least *slack* defined as the remaining time to due-date minus the remaining planned processing time for the project in question.
4) Earliest completion time for the activity in question.
5) Latest due-date for the project in question.
6) Least *reverse slack* defined as the remaining time from the earliest possible start minus the remaining planned processing time for the project in question.
7) Latest possible start time for the activity in question.
8) Randomly.

If the chosen criterion not for sure determine which activity to choose as the next one in the scheduling, but

gives more than one possibility, we maybe can use one of the other mentioned criteria as a secondary one.

2.1 Time Interface.

In connection with each activity we need to know the processing time: *time*. Furthermore we have to know which resources shall take part in the process. The quantity *time* is the total time needed to carry out the whole activity, but maybe it will not be necessary for all the required resources to take part in the process in the whole period of time. For activity y we can define for resource no. k the *start-idle-time*, $its_k(y)$, as the time from the start of the activity until resource no. k needs to take part. In a similar way we can define the *end-idle-time*, $ite_k(y)$, as the time from the end of use of the resource no. k until the very end of the whole activity, see the example in figure 2.2.

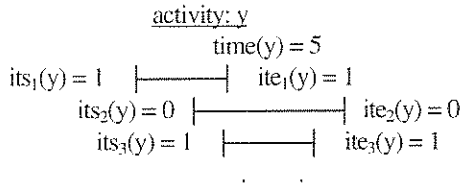

figure 2.2

In figure 2.2 the *end-time* for activity y, et(y), is computed by

(2.9) $et(y) = st(y) + time(y)$

where st(y) is the *start-time* and *time(y)* is the processing time for activity y.

In the scheduling it is possible to take advantage of these idle-times. Let us for instance look at another activity x given by figure 2.3

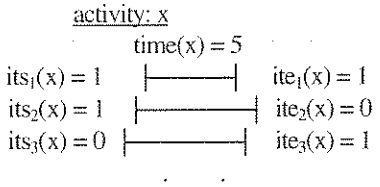

figure 2.3

Assuming activity y is the only predecessor for activity x, such that y <= x, we have to find the start-time for activity x given the start-time for activity y, see figure 2.4

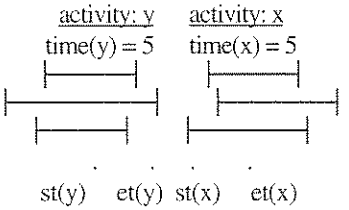

fig. 2.4

If activity x have to start as early as possible without any regard to some possible resource constraints, the final result of the computation, is shown in figure 2.5.

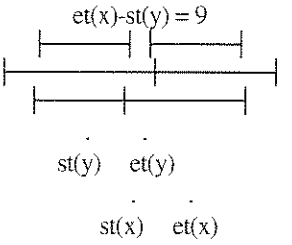

fig. 2.5

Notice that the sum of the processing times is 10, but the time from the start of activity y to the end of activity x is just 9.

3. STEPWISE PROCEDURES.

Making a schedule by a stepwise procedure can for example be done on the basis of one of the following algorithms

(3.1) **STEPWISE PROCEDURES**

1) Forward Algorithm.
2) Backward Algorithm.
3) Forward Algorithm by Random Choice.

3.1 Forward Algorithm.

In making a schedule by a forward algorithm every activity in (1.1) is tried to be planned with a starting time **as early as possible** according to some possible restrictions as mentioned above in (2.1).

For step number k + 1 the possible activities in (2.6) are the activities, for which **all the predecessors are scheduled**.

In the iterative procedure we will very often be in a situation, where in the next step in the scheduling, we have to choose among several possibilities. As criterion, for which of the possible activities we will choose, may for instance be one of the *criterion for selection* as mentioned in (2.8): 1), 2), 3), 4) or 8).

(3.2) **REMARK NO. 3.1**

It may be impossible to find a feasible solution by use of a forward algorithm, in spite of the fact there exist such a feasible solution.

The most important advantages in using the forward algorithm are:

(3.3) **ADVANTAGES.**

1) If possible at all, if the degree of utilization of the resources is not "too big" and if the slacks as defined in (2.8): 3) are not "too close" to zero, it seems to be

rather probable finding a feasible schedule.

2) Once we have found a feasible schedule, on the same assumption as in 1) above, it is very often possible in a reasonable easy way to make an update of the schedule, in the case where some few changes in the restrictions in (2.1) will occur.
3) Once we have found a feasible schedule, in many cases it is possible to make an update of the schedule, in the case where some changes in the volume of order, i. e. the projects in (1.2), will occur.

The most important disadvantages in using the forward algorithm are:

(3.4) **DISADVANTAGES.**

1) A risk for some unnecessary inventory investment.
2) The schedule will normally not be optimal with respect to the chosen objective function in (2.3)
3) If either the utilization of the resources is "too big" or if the slacks as defined in (2.8): 3) are "too close" to zero, it may be impossible to find a feasible schedule, in spite the fact there exist such a feasible schedule.

3.2 Backward Algorithm.

In making a schedule by a backward algorithm every activity in (1.1) is tried to be planned with a starting time **as late as possible** according to some possible restrictions as mentioned above in (2.1).

For step number k + 1 the possible activities in (2.6) are the activities for which **all the successors are scheduled.**

Like in the forward algorithm, in the iterative procedure we will very often be in a situation, where in the next step in the scheduling, we have to choose among several possibilities. In the backward algorithm, as criterion, for which of the possible activities we will choose, may for instance be one of *the criterion for selection* as mentioned in (2.8): 5), 6), 7) or 8).

(3.5) **REMARK NO. 3.2**

It may be impossible to find a feasible solution by use of a backward algorithm, in spite of the fact there exist such a feasible solution.

The most important advantages in using the backward algorithm are:

(3.6) **ADVANTAGES.**

1) If possible at all, if the degree of utilization of the resources is not "too big" and if the reverse slacks as defined in (2.8): 6) are not "too close" to zero, it seems to be rather probable finding a feasible schedule.
2) The inventory investment will in many cases be smaller than it will be by using the forward algorithm.

The most important disadvantages in using the backward algorithm are:

(3.7) **DISADVANTAGES.**

1) If we have a feasible schedule it is often difficult in a reasonable easy way to make an update of the schedule in the case where some changes in the restrictions in (2.1) or in the volume of orders will occur.
2) The schedule will normally not be optimal with respect to the chosen objective function in (2.3)
3) If either the utilization of the resources is "too big" or if the reverse slacks as defined in (2.8): 6) are "too close" to zero, it may be impossible to find a feasible schedule, in spite the fact there exist such a feasible schedule.

3.3 Forward Algorithm by Random Choice.

In the forward algorithm for step number k + 1 the set of the possible activities, $POSSIBLE_{k+1}$, consists in the activities, for which all the predecessors are scheduled. Normally this set will consist of more than one activity. Instead of using a fixed deterministic rule, we can select the next possible activity, $NEXT_{k+1}$, by a random choice. This choice can for instance be done by giving all activities the same probability, and that will say using (2.8): 8). Instead we can use one of the criterion for selection in (2.8): 1), 2), 3) or 4). In these cases the probability of choosing a certain activity may depend of the value of the criterion in question in such a way, the smaller value the bigger probability. This can be done in many ways. One method is shown in example 3.1.

Example 3.1

Given 3 different activities with slack values

activity no.	1	2	3
slack	6	7	9

First of all we subtract the smallest value, in this case equal to 6.

activity no.	1	2	3
new value	0	1	3

Then we add the biggest value, in this case equal to 3.

activity no.	1	2	3
new value	3	4	6

Notice, in this way the smallest value is now half as big as the biggest value.
For the reciprocal values we get

activity no.	1	2	3
new value	.3333	.2500	.1667

In order to normalize these new values in such a way that the sum equals to 1, we divide the values with the current value of the sum, in this case 0.75.

activity no.	1	2	3
new value	.4444	.3333	.2222

Using these new values as probabilities in the random choice we will obtain that the probability of choosing activity no. 1 with the smallest value of the slack is twice the probability of choosing activity no. 3 with the largest value of the slack.

Notice the random choice of a certain activity can be done in many other ways with some other probabilities for the different events and possibly less time consuming at runtime.

3.3.1 Scheduling Procedure.

The iterative procedure in the scheduling now consist of in the next step to choose an activity randomly according to the probabilities for the possible activities in question. The schedule have to fulfill the restrictions 1) and 3) in (2.1): *Arrival times/earliest start times* and *resource restrictions*. If the final schedule do not fulfill the restriction 2) in (2.1): the *due dates*, the schedule will be canceled. Otherwise the schedule is a feasible schedule, and we can inspect if the value of the chosen objective function in (2.3) is better than the similar values of the previously obtained feasible schedules. This procedure will continue either until we have examined a in advance fixed number of schedules, or until the value of the chosen objective function *meets our expectation*.

Making a schedule normally is a recurrent problem as the conditions in the surroundings continually are subject to changes. There may for instance be two different main reasons for making a schedule

(3.8) **SCHEDULING REASONS.**

1) Some of the planned process times may be exceeded, there may be some changes in the available capacities of the resources, there may be some modifications in the projects or there may be an addition of a new project which have to be included in the schedule.
2) There may be an addition of a new project for which in connection with making an offer on this new project we have to find the earliest possible due-date.

In connection with scheduling reason 1) in (3.8), in the random choice of the next activity, and by that the computation of the probabilities, we can apply from (2.8) as *criterion for selection* as shown in (3.9)

(3.9) **CRITERION FOR SELECTION.**

1) Earliest due-date for the project in question.
2) Least remaining processing time for the project in question.
3) Least slack defined as the remaining time to due-date minus the remaining planned processing time for the project in question.
4) Earliest completion time for the activity in question.
8) Randomly.

In connection with scheduling reason 2) in (3.8), we do not know the due-date for the new project, therefore we can not compute the slack. In this case we can apply from (2.8) as *criterion for selection* as shown in (3.10)

(3.10) **CRITERION FOR SELECTION.**

1) Earliest due-date for the project in question.
2) Least remaining processing time for the project in question.
4) Earliest completion time for the activity in question.
8) Randomly.

In addition to the advantages in using the forward algorithm as mentioned in (3.3), the most important advantages in using *the forward algorithm by random choice* are:

(3.11) **ADVANTAGES.**

1) If possible at all we normally will be sure to find a feasible solution.
2) We normally will be sure to find an optimal schedule, or at least a schedule very close to an optimal one, with respect to the chosen objective function in (2.3).

The most important disadvantages in using *the forward algorithm by random choice* are:

(3.12) **DISADVANTAGES.**

1) A risk for some unnecessary inventory investment.
2) More time consuming.

4. COMPUTATION.

For every project in (1.2) we first of all need to know

(4.1) **PROJECT DATA.**
1) **noa**: number of activities

Apart from this information we need to know maybe not for all but for some of the activities belonging to the project in question:

(4.2) **PROJECT DATA.**
2) **est**: earliest start time
3) **dd**: due date.

Furthermore for each activity we need to know:

(4.3) **PROJECT DATA.**
4) **an**: activity number
5) **nop**: number of predecessors
6) **lpd**: list of predecessors
7) **nos**: number of successors
8) **lsc**: list of successors
9) **pt**: process time
10) **nor**: number of resources
11) **lr**: list of resources

For each resource in (4.3): 11) we need:

(4.4) **PROJECT DATA.**
12) **rn**: resource number
13) **its**: start-idle-time
14) **ite**: end-idle-time

Additionally for all of the resource we need to know:

(4.5) **RESOURCE DATA.**
15) How many activities the different resources can handle at the same time.
16) Possible time intervals in which the different resources are pre-occupied.

Finally for each of the project in the volume of orders we have to have a distinct number and information of the project type:

(4.6) **RESOURCE DATA.**
17) **pn**: project number
18) **typ**: project type

Example 4.1

A factory manufactures two different types of products: *type A* and *type B*.

Products of type A consist of 4 different activities, where the precedence relations between the activities are shown in figure 1.1. The data according to (4.3): 4), 9), 10), 12), 13) and 14) are shown in table 4.1.

an	pt	nor	rn	its	ite
1	3	2	0	1	0
			3	0	1
2	5	2	0	0	0
			1	1	2
3	4	2	0	0	0
			4	1	0
4	4	3	0	1	1
			2	0	1
			3	0	0

table 4.1

Referring to **REMARK 1.1** the resource number for the project in question is fixed to 0.

Product of type B consists of 7 different activities, where the precedence relations between the activities are shown in figure 4.1.

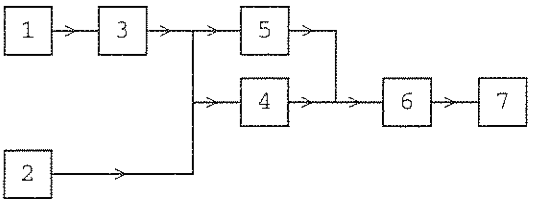

figure 4.1

The data according to (4.3): 4), 9), 10), 12), 13) and 14) are shown in table 4.2.

an	pt	nor	rn	its	ite
1	4	2	0	0	0
			1	1	1
2	5	2	0	1	0
			1	0	0
3	3	2	0	0	0
			2	0	1
4	4	2	0	0	1
			3	1	0
5	3	2	0	0	0
			4	0	0
6	4	3	0	0	0
			1	1	1
			5	0	0
5	4	2	0	0	0
			6	0	0

table 4.2

We will assume, we have three different orders on the product of type A and three different orders on the product of type B, each order with its own *earliest start time* and *due date*.

The data according to (4.2): 2) and 3) for the three project of type A are shown in table 4.3.

pn	an	est	dd
2	1	6	
	2	6	
	4		36
4	1	9	
	2	9	
	4		38
6	1	9	
	2	9	
	4		44

table 4.3

The data according to (4.2): 2) and 3) for the three project of type B are shown in table 4.4

pn	an	est	dd
1	1	4	
	2	5	
	7		40
3	1	8	
	2	8	
	7		40
5	1	8	
	2	8	
	7		50

table 4.4

The information concerning the data in (4.5): 15) and 16) can be deduced from the results in figure 4.2 and figure 4.3, where ▓ means, the resource is pre-occupied. From the figures it appears that resource number 1 and resource number 3 may handle two activities at the same time.

As mentioned in (3.8), for instance there may be two different reasons for making a schedule. In the first place we will deal with the case (3.8): 2)

ADDITION OF A NEW PROJECT.

In this connection we have to find a feasible schedule with the earliest possible due-date for the last project, the project with the largest project number, in this case **pn = 6**. That will say, in (2.3) *objective function* we will use 5): *Earliest completion time for the last project.*

The final result of the simulation is shown in figure 4.2. The lines starting with p contain the information for the different project, and the lines starting with represent the information for the different resources.

figure 4.2

The earliest possible due-date for the last project seems to be at the end of time unit 26 and that is to say immediately before time unit 27. In this case the completion time for the latest project is at the end of time unit 50.

In the second place we will deal with the case (3.8): 1)

NEW SCHEDULE.

In this connection we have to find a feasible schedule with the earliest possible due-date for the latest project. That will say, in (2.3) *objective function* we will use 3): *Least possible makespan.*

The final result of the simulation is shown in figure 4.3.

figure 4.3

The earliest possible due-date for latest project seems to be at the end of time unit 49. In this case the completion date for project number 6 is at the end of time unit 41, which gives a feasible schedule as the due-date for project number 6 has been fixed to 44.

5. CONCLUSION.

The simulation in Example 4.1 was carried out by three different of the *criterion for selection* in (2.8)

(5.1) **CRITERION FOR SELECTION.**

1) Earliest due-date for the project in question.
3) Least *slack* defined as the remaining time to due-date minus the remaining planned processing time for the project in question.
8) Randomly.

Judge by the efficiency on runtime, i. e how rapidly the best schedule was generated, best with respect to the chosen *objective function* in (2.3), the order of precedence seems to be (with the most efficient first)

(5.2) **ORDER OF EFFICIENCY.**

1) Least slack
2) Earliest due-date
3) Randomly.

However in the case with *addition of a new project* we probably do not know the due-date for the last project. Accordingly we can not find the slack and by that, we can not use 1) *Least slack* as *criterion for selection.*

6. SELECTED REFERENCES.

[1] ANDERSEN, KIM: *Scheduling and Resource Planning in Turbo-Pascal.* Department of Production, Aalborg University, 1994. (1994F-02)

[2] ANDERSEN, KIM: *Scheduling by Simulation. The First Congress on Intelligent Manufacturing Processes & Systems.* University of Puerto Rico, 1995. (1994F-08)

[3] ANDERSEN, KIM: *Scheduling Procedures.* Department of Production, Aalborg University, 1995. (1995F-10)

[4] ANDERSEN, KIM: *Scheduling by use of the n-traveling Salesmen Problem.* IFORS'96, Vancouver, Canada, 1996. (1996F-11)

[5] CONWAY, RICHARD W. & MAXWELL, WILLIAM L. & MILLER, LOUIS W.: Theory of Scheduling. Addison-Wesley Publishing Company. 1967.

[6] JOHNSON, LYNWOOD A. & MONTGOMERY, DOUGLAS C.: *Operations Research in Production Planning, Scheduling, and Inventory Control.* John Wiley & Sons, Inc., 1974.

[7] MODER, JOSEPH J. & PHILLIPS, CECIL R.: Project Management with CPM and PERT. Reinhold Publishing Corporation. 1964.

[8] MUTH, JOHN F. & THOMPSON, GERALD L.: Industrial Scheduling. Prentice-Hall, Inc., 1963.