1. INTRODUCTION

The design process can be considered as a serial execution of the product development phases shown in Figure 1. Each phase can be decomposed into a set of interrelated design activities. The main purpose of this decomposition is to gain control over the total duration of the design process and utilization of resources during planning and execution, so that total cost is minimized. The design activities are performed with the resources available and according to the precedence constraints and some predefined scheduling rules (Belhe and Kusiak 1997).

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identify customer requirements</td>
<td>Preliminary design</td>
<td>System design</td>
<td>Detail building</td>
<td>Testing and evaluation</td>
</tr>
</tbody>
</table>

Figure 1. Five phase design process

With the advent of concurrent engineering (CE), an emphasis is given on the integration of design with manufacturing and other disciplines. Due to the concurrent consideration of many
factors, the design process in a CE environment requires input from various disciplines. In an organization, people with similar skills often form functional groups (departments). The product development activities are performed in teams representing different disciplines. In a CE environment, experts from different functional groups need to contribute towards the design goal. As the design process progresses, the design team may change its form. Since the main emphasis of the design organization is on the concurrent consideration of many factors, a design activity can not proceed if the resource requirements are not satisfied. A project manager is responsible for scheduling and monitoring the activities to be performed by the team. The tools for management of design activities have to be more flexible than the existing project management techniques (e.g., see Londono et al. 1992).

Once the resource requirements of the design activity are determined, it is necessary to confirm that they are available before the activities are undertaken. Since the total number of resources in the organization is limited, resource conflicts may occur among design activities of the same design project or different projects and it may not be possible to undertake all the design activities simultaneously. Then, the decision has to be made about which activities are to be scheduled. After completion of some design activities the number of eligible activities may change and hence it may be necessary to restructure the design teams. A number of members may leave and/or members from some functional group may join the design team.

Scheduling design activities should be an essential part of the concurrent engineering process. A number of problems stem from the fact that the nature of design activities is quite different from manufacturing activities. The predictability and standardization found in manufacturing is not present to the same degree in development of new products. In design projects, uncertainty and diversity seem to be the predominant features.

2. PROBLEM DESCRIPTION

The goal of this chapter is to model and solve the problem of scheduling design activities constrained by the precedence constraints as well as discrete resources. The case with dynamic resource constraints is of particular interest. Solving this problem, characteristic of industrial situations, involves the determination of a schedule subject not only to limited resources but also subject to unforeseen changes. For example, the increase or decrease in the level of one or more resources or management’s decision to purposely delay one or more of the design projects.

Therefore any schedule developed at time \( t \) may have to be extensively modified after some \( t_1 \) time units. Such a situation precludes the benefits from computing a schedule over a long planning horizon. The development of a long horizon schedule would not be practical and this effort would have to be repeated every time that a change occurs.

The approach considered in this chapter is to develop at a given time instant a schedule for a relatively short time interval. Once the schedule in this interval is close to its completion, the procedure can be repeated for the next interval. The interval length can be determined, for example, based on the longest processing time of the activities that are selected for that time interval. The design activities need multiple types of resources. These resources are experts from different functional groups and therefore it is assumed that these resources are not interchangeable. The objective is to schedule design activities at time \( t \) so that total reward is
maximized without violating any of the existing constraints. Therefore the resource constrained scheduling problem at some specific time $t$ can be represented as a multidimensional knapsack model (Nemhauser and Wolsey, 1988). Once this problem is solved, one can then move to a new instant of time and solves another knapsack model. If these time instances are properly selected, and the results are properly interpreted, a complete schedule can be obtained over time.

Design activities have to compete for the limited available resources. Therefore the schedule of design activities obtained should be feasible also with respect to the multiple resource constraints.

The resource-constrained project scheduling problem (RCPSP) has been studied extensively in the project management literature. For a recent survey of project scheduling techniques, see Icmeli et al. (1993).

2.1 Iterative Nature of the Design Process

In addition to the dynamic resource constraints, cycles occur quite frequently in a design process. For example, in the design of a power circuit module, the layout of the circuit may need to be changed based on the results of the tests performed. This results in a cycle $j - k - j$, as shown in Figure 2.

![Figure 2. A cycle in the design process](image)

Due to the cycles, it is difficult to determine a priori the path followed by the design project through the network of design activities. For example, for the process in Figure 2 the following paths are possible: $(i - j - k - n)$ or $(i - j - k - j - k - n)$ or $(i - j - k - j - k - j - k - n)$, and so on. It is also difficult to estimate the duration of remote downstream design activities.

Pritsker (1979) analyzed the probabilistic networks with feedback loops. This approach can not be used for the design activities, because the stochastic project networks assume a fixed probability with which a certain activity is executed. A Q-GERT network approach (Neumann, 1990) assumes branching with a certain probability. In the case of a design activity network, after a number of iterations have been performed, the probability of repeating the same sequence of activities reduces.

In the approach presented in this chapter, the set of activities eligible for scheduling (called here schedulable activities) at time $t$ is determined, which is useful in dealing with the uncertainty of a design path. For example, after the completion of activity $k$ in Figure 2, it is known to the scheduler whether the next schedulable activity is going to be activity $j$ or activity $n$. 
3. MATHEMATICAL PROGRAMMING MODEL

Since the resources are limited, it may not be possible to perform at time $t$ all activities in the set of schedulable activities. The problem of selecting activities to be scheduled is arising. As the resource requirements of individual design activities are known, a solution to the problem of scheduling design activities at time $t$ is in the form of a selected subset of schedulable activities. In this section, the problem of scheduling design activities at time $t$ is formulated as a multidimensional knapsack problem.

Define:

- $n$: the total number of design activities eligible for scheduling at time $t$
- $m$: number of types of resources available
- $p_i$: the reward obtained by undertaking activity $i$
- $r_{ij}$: the number of resources of type $j$ required to perform activity $i$
- $R_j$: the total number of resources of type $j$ available
- $x_i$: \[ \begin{cases} 1 & \text{if activity } i \text{ is scheduled} \\ 0 & \text{otherwise} \end{cases} \]
- $t_i$: the duration of activity $i$
- $k_i$: due date of activity $i$

The design activities can be selected based on some criteria. Reward indices are defined for selection of design activities, based on the objectives selected. In this chapter, the reward index used is described next.

Let the non-time-based importance of activity $i$ be indicated by the weight $w_i$. The weight $w_i$ may represent management’s preferences towards some design activities.

The slack for activity $i$ at time $t$ is

$$ s_i(t) = k_i - t_i - t $$

Let the urgency of undertaking activity $i$ be expressed by index $V_i(t)$, the function of the slack $s_i(t)$. The function $f(s_i(t))$ is defined in such a way that, it takes a large value if the activity can not be completed before the due date and it should take a relatively smaller value otherwise. For example,

$$ V_i(t) = f(s_i(t)) = \begin{cases} K s_i(t) & \text{if } s_i(t) \leq 0 \\ M / s_i(t) & \text{otherwise} \end{cases} \tag{1} $$

where $K$ and $M$ are constants.

The reward index is represented as a combination of cost expressed as a function of slack and weight of the design activity $w_i$. Then the reward index obtained by undertaking (not delaying) the activity $i$ is given by $p_i = w_i V_i(t)$.

For example, assume that two activities with weights 20 and 10, respectively are to be scheduled at time $t = 5$. The values of $K$ and $M$ are 50 and 10, respectively. Suppose that, the first activity has a duration of 3 units and it is due at $t = 10$. Then the reward index of the first activity is 5. If the second activity has duration of 7 units and it is due at $t = 9$, then the reward index for the second activity is 150.

The problem of selecting design activities for scheduling is formulated as follows:
Max $Z = \sum_{i=1}^{n} p_i x_i$ \hspace{1cm} (2)

s.t. $\sum_{i=1}^{n} r_{ij} x_i \leq R_j \hspace{1cm} j = 1, 2, ..., m$ \hspace{1cm} (3)

$x_i = 0 \text{ or } 1 \hspace{1cm} i = 1, 2, ..., n$ \hspace{1cm} (4)

The objective function is the total of reward indices of all design activities to be performed. Constraint (2) indicates the limited availability of all types of resources. It has been shown that the problem ((2) - (4)) is NP-complete (Karp 1972).


Two efficient solution approach for solving the problem ((2) - (4)) are presented in this section. Before presenting the solution approaches, the problem ((2) - (4)) is converted to a canonical form.

For each design activity $i$, define

$$a_{ij} = \frac{r_{ij}}{R_j} \hspace{1cm} j = 1, 2, ..., m$$

Then problem ((2) - (4)) is equivalent to the following problem:

Max $Z = \sum_{i=1}^{n} p_i x_i$ \hspace{1cm} (5)

s.t. $\sum_{i=1}^{n} a_{ij} x_i \leq 1 \hspace{1cm} j = 1, 2, ..., m$ \hspace{1cm} (6)

$x_i = 0 \text{ or } 1 \hspace{1cm} i = 1, 2, ..., n$ \hspace{1cm} (7)

In order to solve the problem ((5) - (7)), a push-type heuristic approach is presented in the next section.

4. PUSH SCHEDULING MODE

4.1 Basic Solution Approach

Toyoda (1975) proposed the primal effective gradient method. In the basic approach discussed here, instead of computing gradients, a schedule index is developed for each design activity. The activities to be performed in order to maximize the total reward are determined by obtaining the set of activities that have to be delayed. Initially, all eligible activities are included in the set of activities to be scheduled. The resource requirements for all these activities may be satisfied with the available resources. Then the solution found is not only feasible but also
optimal. If the solution obtained is not feasible then the algorithm proceeds to search for a feasible solution.

Denote:

- $E$ set of design activities eligible for scheduling at time $t$
- $D$ set of set of eligible design activities to be delayed (subset of $E$)
- $U$ set of eligible design activities to be scheduled (subset of $E$)

The resource requirement of all the activities in set $U$ needs to be satisfied. It means that, only the less beneficial activities are to be delayed (if necessary) so that the total reward is maximized.

The status of each resource is maintained in the vector $B = [b_1, ..., b_j, ..., b_m]$, where,

$$b_j = 1 - \sum_{i \in U} a_{ij}$$

The vector $B = [b_j]$ is computed based on the design activities in set $U$. A feasible solution to problem ((5) - (7)) implies that the vector $B$ corresponding to this solution is non-negative. In order to determine a solution to this problem, consider that all eligible design activities are to be scheduled. If the vector $B$ is non-negative then it means that all resource constraints are satisfied and hence all eligible activities can be scheduled simultaneously. If the vector $B$ is negative then it means that some of the resource constraints are violated and some activities need to be delayed based on some criteria. The delayed activities are those which minimize the decrease in the total reward. For this purpose, a schedule index $d_i$ is used to select the design activities to be delayed. The schedule index $d_i$ for activity $i$ is defined as reward per unit fraction of the total resources required by activity $i$, i.e.,

$$d_i = \frac{p_i}{\sum_{j=1}^{m} a_{ij} b_j}$$

The design activity $i$ with the minimum value of the schedule index $d_i$ should be delayed first. The activity $i$ is delayed (deleted from $U$) provided that the vector $B$ is not non-negative. Note that in rare cases when the denominator in (9) would become 0, it would have to be replaced with an arbitrary small value. This protection should be incorporated in the algorithm presented next.

The main idea behind this procedure is to use the schedule index $d_i$ obtained from (9) to decide which activities are to be delayed. In the algorithm presented below, all eligible activities are included in the set of activities to be scheduled and the feasibility of this solution is checked. If it is found that the current solution violates the resource constraints, then the activity with the smallest schedule index is delayed first. This process of delaying activities is discontinued as soon as a feasible solution is obtained.

**Algorithm**

1. Set $U = E$ and $D = \phi$.
2. Compute vector $B$, from equation (8). Based on equation (9), compute schedule indices $d_i$ for all activities in set $U$. Sort design activities in ascending order of their schedule indices.
3. If $B$ is non-negative, then go to step 4, otherwise go to step 2.
4. Select activity $k$ with minimum schedule index $d_i$ in the set $U$.
   - Set: $U = U \setminus \{k\}$ and $D = D \cup \{k\}$.
Step 3. Update vector $B$ and index $d_i$. If $B$ is non-negative go to step 4, otherwise go to Step 2.

A large number of knapsack models may need to be solved in order to determine the entire schedule. The quality of solution of the series of knapsack problems solved to determine the schedule of the design project may have a great impact on the quality of the schedule generated. Therefore, it might be necessary to improve the quality of solutions of the multidimensional knapsack problem ((5) - (7)), without largely increasing the complexity of the solution procedure (see Belhe and Kusiak 1997).

### 4.2 Case Study

In this section, a case study is presented. Consider design of two electronic products $P_1$ and $P_2$. The design process is divided into five phases $A$, $B$, $C$, $D$ and $E$ (see Figure 1).

Each of the two products $P_1$ and $P_2$ is at a different stage of its completion. The core design teams for products $P_1$ and $P_2$ perform design activities and they seek human resources from functional groups according to the resource requirements of these activities. Consider time instant $t$ and design activities for $P_1$ and $P_2$ eligible at time $t$. At time $t$, product $P_1$ is in phase $B$ and product $P_2$ is in phase $E$. The parts of design activity networks for phase $B$ and phase $E$ (corresponding to $P_1$ and $P_2$) are shown in Figure 3 and Figure 4 respectively. The schedulable activities 1 through 5 for $P_1$ and $P_2$ are indicated with shaded blocks.

At some time instant $t$ for product $P_1$ the following activities become eligible for scheduling:
1. Building critical circuits and testing
2. Prepare test specifications
3. Prepare allocation of costs

Similarly, for product $P_2$, the following design activities become schedulable:
4. Finalize test plans
5. Prepare customer required test data

Therefore, the set of schedulable activities consists of 5 activities. To perform the above mentioned five activities the core design teams for $P_1$ and $P_2$ need resources from the following three functional groups:
1. Reliability and maintainability ($R_1$)
2. Product support ($R_2$)
3. Testing ($R_3$)

The weight $w_i$ of each of the five activities and the values of slacks at the current time instant are listed in Table 1. The values of $V(t)$ are calculated using equation (1) (with $K = 5$, $M = 10$) and then the reward indices are obtained (see Table 1).

Figure 3. A partial design activity network for product P1 in phase B

Figure 4. A partial design process for product P2 at Phase E

Table 1. Weights and Reward Indices of Design Activities

<table>
<thead>
<tr>
<th>Activity</th>
<th>$w_i$</th>
<th>$s(t)$</th>
<th>$V_i(t)$</th>
<th>Reward index ($p_i$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>-1</td>
<td>5</td>
<td>15</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>2</td>
<td>5</td>
<td>25</td>
</tr>
<tr>
<td>3</td>
<td>10</td>
<td>5</td>
<td>2</td>
<td>20</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>-1.6</td>
<td>8</td>
<td>40</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>1</td>
<td>10</td>
<td>20</td>
</tr>
</tbody>
</table>

The set of eligible activities is:

$$E = \{1, 2, 3, 4, 5\}$$

The resource requirements of the eligible activities are indicated in Table 2. Three resources R1, R2, and R3 are available in quantities 8, 9, and 7, respectively.
Iteration 1

Step 1. \( U = E = \{1, 2, 3, 4, 5\} \) and \( D = \emptyset \).

The resource status vector \( B \) is calculated, \( B = [-0.38, -0.56, -0.43] \).

Since the resource status vector is negative, it is necessary to delay some activities presently included in \( U \) in order to determine a feasible solution to the problem.

The schedule index \( d_i \) for all activities in set \( U \) are calculated in Table 2.

<table>
<thead>
<tr>
<th>Activity</th>
<th>Resource Requirements of Activities</th>
<th>( a_{i1} )</th>
<th>( a_{i2} )</th>
<th>( a_{i3} )</th>
<th>( d_i )</th>
<th>Rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1 2 1</td>
<td>1/8</td>
<td>2/9</td>
<td>1/7</td>
<td>64.78</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>2 3 1</td>
<td>2/8</td>
<td>3/9</td>
<td>1/7</td>
<td>73.49</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>3 3 3</td>
<td>3/8</td>
<td>3/9</td>
<td>3/7</td>
<td>39.26</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>4 5 3</td>
<td>4/8</td>
<td>5/9</td>
<td>3/7</td>
<td>58.84</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>1 1 2</td>
<td>1/8</td>
<td>1/9</td>
<td>2/7</td>
<td>86.56</td>
<td>5</td>
</tr>
</tbody>
</table>

Step 2. The design activities are sorted in ascending order of their schedule indices. The activity with the minimum schedule index is activity 3. Then, \( U = \{1, 2, 4, 5\} \) and \( D = \{3\} \).

Step 3. \( B = [0.00, -0.22, 0.00] \). The schedule index \( d_i \) for all activities in set \( U \) are shown in Table 3.

Since vector \( B \) is still negative, another iteration is required.

<table>
<thead>
<tr>
<th>Activity</th>
<th>Resource Requirements of Activities</th>
<th>( a_{i1} )</th>
<th>( a_{i2} )</th>
<th>( a_{i3} )</th>
<th>( d_i )</th>
<th>Rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1 2 1</td>
<td>1/8</td>
<td>2/9</td>
<td>1/7</td>
<td>306.8</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2 3 1</td>
<td>2/8</td>
<td>3/9</td>
<td>1/7</td>
<td>340.9</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>4 5 3</td>
<td>4/8</td>
<td>5/9</td>
<td>3/7</td>
<td>327.3</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>1 1 2</td>
<td>1/8</td>
<td>1/9</td>
<td>2/7</td>
<td>818.2</td>
<td>4</td>
</tr>
</tbody>
</table>

Iteration 2
Step 2. The activity with the minimum schedule index in set $U$ is activity 1. Then, \[ U = \{2, 4, 5\} \text{ and } D = \{1, 3\}. \]

Step 3. $B = [0.50, 0.33, 0.43]$. Since $B$ is non-negative, the set $U$ is accepted as an initial feasible solution to the problem considered.

The solution is:
\[ U = \{2, 4, 5\} \quad \text{and} \quad D = \{1, 3\} \]

The value of the objective function is 85.

The iterations performed are summarized in Table 4.

<table>
<thead>
<tr>
<th>Iteration</th>
<th>Activity</th>
<th>Resource Status Vector</th>
<th>Decision</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>None</td>
<td>[-0.38, -0.56, -0.43]</td>
<td>Continue</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>[0.00, -0.22, 0.00]</td>
<td>Delay activity 3</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>[0.50, 0.33, 0.43]</td>
<td>Delay activity 1 (the last activity to be delayed)</td>
</tr>
</tbody>
</table>

The following activities are scheduled:
2. Prepare test specification
4. Finalize test plans
5. Prepare customer required test data

The following activities are delayed:
1. Building critical circuits and testing
3. Prepare allocation of costs

The computational results with the above procedure are promising and they are reported in Belhe and Kusiak (1997).

The LINDO input of the formulation in ((5) - (7)) for the data in Table 1 is shown next.

\[
\begin{align*}
\text{MAX} & \quad 15X_1 + 25X_2 + 20X_3 + 40X_4 + 20X_5 \\
\text{SUBJECT TO} & \quad 0.13X_1 + 0.25X_2 + 0.38X_3 + 0.5X_4 + 0.13X_5 \leq 1 \\
& \quad 0.22X_1 + 0.33X_2 + 0.33X_3 + 0.56X_4 + 0.11X_5 \leq 1 \\
& \quad 0.14X_1 + 0.14X_2 + 0.43X_3 + 0.43X_4 + 0.29X_5 \leq 1 \\
\text{END} \\
\text{INTEGER} & \quad 5
\end{align*}
\]

This formulation produced the same solution as the heuristic algorithm.
5. PULL SCHEDULING MODE

The flow of information among design activities can be considered in two different ways:
1) Each design activity generates the intended information and passes on to the succeeding activities in the network. The activities are performed according to a predetermined schedule. In a way, the information is ‘pushed’ down to the succeeding design activities irrespective of its importance that may vary over a time period.
2) Another strategy is to schedule design activities dynamically according to the importance of the information generated by these activities. While scheduling the design activities, weights of these activities are changed according to the importance of the generated information with respect to the succeeding activities and a new schedule is obtained. The schedule tends to pull the information from the preceding activities.

In this section, the second approach is adopted. Design activities are scheduled with the objective of minimizing the total weighted lateness of these activities. The level of detail in which information is generated by each design activity may vary between design projects. Since the weights and expected duration of design activities in the network change over time with respect to the demands for information from the succeeding design activities, we propose to solve this problem by decomposing it into multiple problems of minimizing the total weighted lateness over an interval of time.

5.1 Analogy of Design Process to Pull System

A proper flow of information is required for timely execution of design activities. In a pull system interpretation of the design process, the importance of the information required by the succeeding activities governs its flow throughout the design process, instead of the design information being released according to some predetermined schedule. The design process resources are used to generate the information only when the information is required. The flow of the design information is adjusted by changing the weights of design activities appropriately. The pull strategy provides a set of rules that control the issuing of information to the design stages. This means that the information is transferred according to the demand. An excess generation of information is considered as waste.

The information can be requested from the preceding activities in a specified format. The principle of production kanbans is useful in this sense. The number of requests made by one design activity to another is determined dynamically and it usually varies for different projects. An individual strategy can be used at a design activity to process (serve) the requests. Similar to the kanban system, the request flag is attached to the information sent back to the succeeding design activity and in this way the receiving design activity can keep track of the incoming information and detect discrepancies of design information, if any.

Whenever it appears that the design activity will soon catch up with the information requests, the scheduling system can reduce the resources used to perform that particular activity. It is believed that, in most cases the scheduling system will be able to maintain a comfortable level of information demand in the entire design process. For this reason, it is assumed that there is always demand at each design activity for at least some type of information and therefore it is possible to assign a weight to each activity relative to the weights of other activities.
5.2 Scheduling Problem

The design process is divided into phases and an activity model represents each of these design phases. The completion time of each of the design phases is regarded as a milestone in the schedule of the entire design project. The due dates for phases in the design process are assigned by the global schedule. The due date for each design phase derives the latest finish time of end activities in the design activity network of the corresponding phase.

The following notation is used in the approach discussed in this section:
- \( R \): vector indicating availability (status) of resources at time \( t \)
- \( r_i \): vector of resource requirements for activity \( i \)
- \( S \): set of schedulable activities at time \( t \)
- \( U \): set of design activities in process at time \( t \)
- \( D \): due date for the current design phase
- \( w_i \): weight of design activity \( i \)
- \( LS(i) \): latest start time of activity \( i \)
- \( LF(i) \): latest finish time of activity \( i \)
- \( d_i \): estimated duration of activity \( i \)
- \( c_i \): completion time of activity \( i \)
- \( s_i \): start time of activity \( i \)

Each design project is unique in nature. Therefore, it is not possible to collect some data to obtain distribution of the random value of the duration, \( d_i \). In the absence of data, simulation practitioners use heuristic procedures for choosing a distribution. We assume that, the quantities \( a \) and \( b \), the subjective estimates of the minimum value of the estimated duration of activity \( d_i \) and the maximum value of \( d_i \) can be obtained. Similarly, the subjective estimate of the most likely value \( g \) can also be obtained. The duration, \( d_i \) is then considered to have triangular distribution on the interval \([a, b]\), with mode \( g \). The expected value of \( d_i \) is determined using, \( d_i = (a + b + g) / 3 \). The expected value of \( d_i \) is used as deterministic value of activity duration in further analysis.

The schedule of design activities need to be determined under the following constraints:

**Temporal Constraints:** The four possible relations that constrain the start and finish times of two activities are specified by lead/lags as follows: the start-to-start (SS), the finish-to-finish (FF), the start-to-finish (SF), and the finish-to-start (FS) relations. In addition to these relations, four more relations are used that may hold between the beginning of the design phase, end of the design phase, and the activity start and finish times (see Elmaghraby and Kamburowski 1992). These relations are begin-to-start (BS), begin-to-finish (BF), start-to-end (SE), and finish-to-end (FE). The latter relations permit specification of a start lead/lag and the finish lead/lag for any design activity relative to the design phase beginning and end, respectively. The beginning time of any design phase can be assumed to be zero.
Figure 5. Design activities

For example, consider the design activities in Figure 5. The temporal constraints need to be considered in addition to the precedence constraints. Assume the constraints:

1) \( s_3 \geq c_2 + 3 \)
2) \( s_5 \geq s_3 + 5 \)
3) \( c_5 \geq c_4 + 6 \)
4) end of phase \( \geq c_7 + 3 \)

The first constraint is of type finish-to-start (FS) and it says that activity 3 can begin only after at least 3 time periods after completion of activity 2. Second constraint is of type start-to-start (SS), according to which activity 5 can begin only after at least 5 time periods after the start time of activity 3. According to constraint 4, finish-to-finish (FF) type constraint, activity 4 and activity 5 should be scheduled in such a way that the completion of activity 5 (as per expected duration of activities 4 and 5) occurs after at least 6 time periods after completion of activity 4. The latest finish time of activity 7 can be calculated based on the constraint 4 (finish-to-end constraint) which says that activity 7 should be completed at least 3 time periods prior to the due date for the design phase shown in Figure 3.

**Resource Constraints:** Consider a set of activities for which the input information required is available, i.e., all preceding design activities have been performed and these activities are eligible for scheduling according to the temporal constraints. Each of these activities needs multiple resources and total available resources are limited. The availability of each type of resource may vary over the time horizon of the design project. However, it is assumed that the total resource availability as well as resource requirement of design activities remains constant during execution of the design activities.

### 5.3 Minimization of the Estimated Weighted Lateness of Design Activities

The objective of the solution presented next is to minimize the total weighted lateness, subject to temporal constraints, precedence constraints, and multiple resource constraints. This problem is dynamic in its nature, in the sense that the weights of design activities, expected duration of design activities and the total resource availability may change over time. Therefore any schedule developed at time \( t \) may have to be extensively modified at time \( t + t_1 \). Such a situation precludes the usefulness for the computation of a schedule over a limited planning horizon. The computational effort involved in the development of an overall schedule would be wasted and the effort itself would have to be repeated every time that a change occurs.
Therefore, we decide to look at the problem at any given time instant, and to develop a schedule over a relatively short interval. At the completion of the first activity after time $t$ the problem is solved again to schedule another set of activities.

Consider set $S$ of activities eligible for scheduling at time $t$. The set $S$ is determined based on the temporal constraints and precedence constraints. The status of resources is indicated by a vector, $R$. We also look at the total number of resources required to perform the activities in set $S$. If the resource requirements of all activities in set $S$ can be satisfied by the resources available then all activities in set $S$ are scheduled and we move to the next scheduling instant. If the total number of resources required by activities in set $S$ is greater than vector $R$, then it is not feasible to schedule all the activities in set $S$. In order to obtain a feasible solution, one or more activities of set $S$ must be delayed. Therefore, all feasible solutions to the problem are subsets of set $S$. In order to schedule as many activities as possible and not to violate the resource constraints, we need to find all subsets of set $S$ such that adding one more activity in the subset will violate the resource constraint. Even though the problem of finding all such subsets is a complex combinatorial problem, for a practical problem of moderate size, the search tree can be reduced to a great extent by removing activities from set $S$ and checking the resource constraints instead of adding activities to an empty set.

Set $S$ is the root of the tree at level zero. At the next level, i.e., level 1, all subsets with one activity less than set $S$ are formed and the resource constraints are evaluated at each of these nodes. If a node results into an infeasible solution then that node is branched into subsets of activities at the next lower level with one activity less than their parent node. This search process continues until all the terminal nodes are feasible subsets of set $S$ such that,

$$\sum_{i=1}^{m} r_i \leq R \leq \sum_{i=1}^{m+1} r_i \quad \text{for some } m$$

(8)

The decision to select one of these terminal nodes is based on the priorities (weights $w_i$) assigned to the eligible design activities, the estimated design activity time ($d_i$) and the latest start time ($LS(i)$) and latest finish time ($LF(i)$) determined using the duration estimates for activities and the due date of the corresponding design phase. For each of the feasible subsets, the next time instant for scheduling is obtained using:

$$t^* = \min_{i \in S, j \in U} \{c_i, c_j\}$$

(9)

The weighted lateness of the activities in each subset is calculated for the corresponding nodes and the node with the minimum value of the weighted lateness is selected. Suppose that a feasible subset, $S'$, with the minimum value of the weighted lateness is selected. All activities in $S'$ are scheduled. As a number of design activities are undertaken the resource status vector is updated. Completion of an activity means that
some information request is met. The information generated is used by the succeeding activities. As a result, some more design activities may become eligible for scheduling. The set $S$ of eligible activities is updated. The scheduling problem is solved again at the new instant, $t^*$. The main advantage of this approach is that, it is dynamic in the sense that the system makes decision at every completion of an activity. For a design activity network with hundreds of activities, the problem solved at any time instant $t$ is considerably small. In addition, the computational effort of re-computing the entire schedule at each time instant is saved. The push mode scheduling algorithm is presented next.

**Algorithm**

Step 0. At the completion of an activity $i$ at time $t$, update the resource status vector $R$ and weights of activities in set $S$. Determine the latest finish time, latest start time and duration for each of these activities at time $t$.

Step 1. Check whether it is feasible to schedule all activities in set $S$ according to the resource constraints. If yes, schedule all activities in set $S$ and move to the next time instant. Otherwise, construct a search tree and obtain all feasible subsets of set $S$ such that (8) is satisfied.

Step 2. Determine the value of $t^*$ for each subset corresponding to each terminal node using (9) and evaluate value of the objective function using the following function $Z$ at all terminal nodes of the search tree:

$$Z = \sum_{i \notin S^k} \max(t^* - LS(i), 0)$$

Step 3. Schedule activities in subset $S^*_k$ for which the value of $Z$ is minimum. If more than one set has the same value of the objective function then break the tie by selecting the set with the maximum number of activities. Set new value of $t$ to $t^*$ corresponding to the subset selected $S^*_k$. Go to Step 0.

In the push approach, the activities are scheduled in a static mode. The availability of resources, weights of activities, and their duration are determined a priori and it is assumed that none of them changes throughout the project duration. Under these constraints, a schedule of activities is obtained to achieve a certain objective. The estimates of design activities duration are based on the amount of design information required as output from these activities. The nature of the design process is dynamic and each design project is unique in nature. As the design process progresses, more information becomes available and the importance of information and the level of detail required from the remaining activities may change. This is simply due to the changing demand of downstream activities. Although the push approach is computationally less demanding than the pull system approach described in this paper, it is highly likely that the schedule obtained using the push approach may be less efficient.

In the pull approach, the schedule of activities is obtained over a short time interval by using the above algorithm. Frequent updates and schedule computations are performed. Since the updates are based on the current requirements of the downstream activities, the pull schedule is of better quality. The approach presented in this paper avoids numerous computations of the entire schedule of activities.

5.4 The General Framework for Scheduling Design Activities
In general, there may be a number of design projects undertaken at the same time. The same activity might need to be performed for different design projects. For example, thermal analysis might need to be performed for different electronic devices in the design process, provided that all the preceding activities have been performed. The weight of the same design activity for different design projects may vary. A design activity common to a number of projects can be considered, for example, as a workstation processing requests for the design information placed in a queue.

The previously discussed algorithm can be generalized for multiple design projects. According to the algorithm, in the case of a single project a set of schedulable activities is considered for this project. Now, consider activity i eligible for scheduling in a number of projects, subject to the precedence and temporal constraints. Therefore, in general it is required to select a project for each schedulable activity. Given a number of projects for which the same activity is eligible, dispatching rules as in the case of single machine scheduling, can be used to prioritize a certain design project with respect to the others corresponding to a particular activity. Arranging the design projects in a sequence at each schedulable activity can be done to minimize a certain objective function. For example, suppose that there are n design projects for which activity i is schedulable. As stated in Baker (1984), the maximum lateness can be minimized by sequencing the projects such that:

\[ LF(i)_1 \leq LF(i)_2 \leq \ldots \leq LF(i)_n \]

where, \( LF(i)_j \) is the latest finish time of activity i corresponding to design project j.

The set of schedulable activities thus formed consists of the activities corresponding to the same or different design projects. The choice of weights for the design activities in this set depends on the importance of the information generated by these activities with respect to the succeeding activities as well as the importance of the design project.

### 5.5 Illustrative Example

An example illustrating application of the pull is presented. The design process has been divided into five different phases (see Figure 1). The model of phase C is shown in Figure 6. The temporal constraints for the activities in phase C are listed in Table 5. The latest finish time for each design activity in the network is calculated considering the expected time duration of design activity and due date for design phase C and the temporal relationships between activities. The latest finish time of each design activity is obtained by setting beginning time of phase C to zero and the due date is adjusted accordingly.
Figure 6. The activities of Phase C

Table 5. The relationship among design activities in Phase C

<table>
<thead>
<tr>
<th>Activity Pair</th>
<th>Relation Type</th>
<th>Relation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Begin phase C, 1</td>
<td>BS</td>
<td>$3 \leq s_1$</td>
</tr>
<tr>
<td>Begin phase C, 2</td>
<td>BS</td>
<td>$0 \leq s_2$</td>
</tr>
<tr>
<td>6, 8</td>
<td>SF</td>
<td>$s_6 + 3 \leq c_8$</td>
</tr>
<tr>
<td>6, 8</td>
<td>FS</td>
<td>$c_6 \leq s_8 + 2$</td>
</tr>
<tr>
<td>9, 13</td>
<td>FS</td>
<td>$c_9 + 2 \leq s_{13}$</td>
</tr>
<tr>
<td>13, 11</td>
<td>SF</td>
<td>$s_{13} + 3 \leq c_{11}$</td>
</tr>
<tr>
<td>11, End of phase C</td>
<td>FE</td>
<td>$c_{11} + 3 \leq D$</td>
</tr>
<tr>
<td>12, End of phase C</td>
<td>FE</td>
<td>$D \leq c_{12} + 1$</td>
</tr>
</tbody>
</table>

Determine the schedule of design activities at $t = 10$.

Step 0. At $t = 10$, the set of activities eligible for scheduling is obtained, $S = \{4, 7, 8, 9, 10\}$. Phase C requires 3 types of resources. The availability of these resources at the time instant given is updated to (10, 8, 8). The end activities of the design Phase C require information, either directly or indirectly, from all the remaining activities in Phase C. Based on the importance of input information required by the succeeding activities at time $t = 10$, weight values are updated considering the information is available at the current time instant. The latest finish times,
latest start times, resource requirements, weights of activities and expected duration of activities at time $t$ are shown in Table 6.

Step 1. Due to the resource requirements of the eligible activities and the limited resource availability it is determined that all activities in set $S$ can not be scheduled simultaneously. Therefore it is required that one or more activities need to be delayed so that the total weighted lateness is minimized. All feasible subsets of set $S$ are determined by delaying the minimum number of possible activities using the search tree rooted at node representing set $S$ (see Figure 7).

Table 6. Activities with their resource requirements, weights, and estimated duration

<table>
<thead>
<tr>
<th>Activity</th>
<th>Resource Requirements</th>
<th>Weights</th>
<th>Estimated Duration</th>
<th>Latest Finish Time</th>
<th>Latest Start Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>(2, 3, 3)</td>
<td>10</td>
<td>2</td>
<td>11</td>
<td>9</td>
</tr>
<tr>
<td>7</td>
<td>(1, 2, 2)</td>
<td>20</td>
<td>7</td>
<td>17</td>
<td>10</td>
</tr>
<tr>
<td>8</td>
<td>(3, 2, 2)</td>
<td>10</td>
<td>3</td>
<td>14</td>
<td>11</td>
</tr>
<tr>
<td>9</td>
<td>(1, 0, 1)</td>
<td>30</td>
<td>6</td>
<td>19</td>
<td>13</td>
</tr>
<tr>
<td>10</td>
<td>(4, 1, 2)</td>
<td>10</td>
<td>3</td>
<td>14</td>
<td>11</td>
</tr>
</tbody>
</table>

Figure 7. Search tree to determine all feasible subsets of the set $S$

Step 2. The schedule of activities corresponding to each node is evaluated using the objective function, $Z$. For example, consider the feasible schedule represented by node $g$ in the search tree in Figure 7. The partial schedule depends upon the resource requirements. The result of scheduling activities 7, 8 and 10 is shown in Figure 8. The processing of design activities 7, 8 and 10 begins at $t = 10$. As a result, according to the expected duration of activities in set $S$, the activities 8 and 10 will be completed at time $t = 13$. Since the only activities being processed are
activities 7, 8 and 10, the first completion after time $t = 10$ is at $t = 13$. The value of the objective function is 40, i.e., $(13 - 9)$ times weight of activity 4. In this way, the values of the objective function, $Z$, at all feasible nodes of the search tree have been calculated (see Table 7).

Table 7. Nodes of the search tree

<table>
<thead>
<tr>
<th>Node</th>
<th>Subset of Activities</th>
<th>Total Resources Required</th>
<th>Objective Function Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>{4, 7, 8, 9, 10}</td>
<td>(11, 8, 10)</td>
<td>infeasible</td>
</tr>
<tr>
<td>b</td>
<td>{7, 8, 9, 10}</td>
<td>(9, 5, 7)</td>
<td>40</td>
</tr>
<tr>
<td>c</td>
<td>{4, 8, 9, 10}</td>
<td>(10, 6, 5)</td>
<td>40</td>
</tr>
<tr>
<td>d</td>
<td>{4, 7, 9, 10}</td>
<td>(8, 6, 8)</td>
<td>10</td>
</tr>
<tr>
<td>e</td>
<td>{4, 7, 8, 10}</td>
<td>(10, 8, 9)</td>
<td>infeasible</td>
</tr>
<tr>
<td>f</td>
<td>{4, 7, 8, 9}</td>
<td>(7, 7, 8)</td>
<td>20</td>
</tr>
<tr>
<td>g</td>
<td>{7, 8, 10}</td>
<td>(8, 5, 6)</td>
<td>40</td>
</tr>
<tr>
<td>h</td>
<td>{4, 8, 10}</td>
<td>(9, 6, 7)</td>
<td>40</td>
</tr>
<tr>
<td>i</td>
<td>{4, 7, 10}</td>
<td>(7, 6, 7)</td>
<td>10</td>
</tr>
<tr>
<td>j</td>
<td>{4, 7, 8}</td>
<td>(6, 7, 7)</td>
<td>10</td>
</tr>
</tbody>
</table>

Step 3. The minimum value of the objective function was found at nodes d, i and j. The subset corresponding to node d has 4 activities and hence the activities 4, 7, 9, and 10 are scheduled. Then, the time instant is set to $t = 12$ and we proceed to Step 0.
6. SUMMARY

The problem of scheduling design activities with precedence constraints and dynamic resource constraints was discussed in this chapter. Due to its complexity, the problem was decomposed into a series of multidimensional knapsack problems. In order to schedule a large number of design activities belonging to multiple projects, a number of large size multidimensional knapsack problems were to be solved. Hence, new solution procedures were developed to solve the subproblem and it was shown that a beam search procedure can largely enhance the quality of the solution.

A design process can be represented as a network of design activities. A number of design projects may be undertaken simultaneously. This chapter deals with the problem of scheduling design activities of multiple design projects competing for the limited available resources. The problem of determining a schedule subject to precedence and resource constraints is difficult to solve. It becomes even more complex when unforeseen changes are considered, for example, in the level of resources. Therefore, the scheduling problem is decomposed into a series of multidimensional (multiresource) knapsack problems. Due to high computational complexity of the multidimensional knapsack problem, two solution procedures are proposed.

The design process can be considered as a series of design phases. Each design phase can be further divided into number of design activities. In the network of design activities, the design activities generate information that is used by the succeeding activities. Since importance of the input information available from different activities changes with respect to the requirement of the
design activities, a pull system approach for the management of design activities is proposed. Due to the concurrent consideration of many factors in the design process, it is required that the design activities are performed by teams of experts. In addition to the multiple resource requirement, the uncertainty in the weights and duration of design activities makes the problem of scheduling design activities complex. A heuristic algorithm was presented to solve the problem considered in this chapter.

A model for implementing a pull system for design activities was presented. The problem of scheduling design activities is a multiple resource constrained scheduling problem. The due dates of individual design phases were treated as milestones to obtain the schedule for each design phase. Consideration was given to the fact that in practice the importance of design activities and duration of activities may change over time. Due to such unpredictable nature of design process and variable resource levels, a dynamic scheduling approach was selected. A new algorithm was discussed and illustrated with an example.

REFERENCES
    Mellon University, Pittsburgh, PA.

QUESTIONS
1. Why do we schedule design activities?
2. When scheduling is not needed?
3. What is a reward index?
4. What is an eligible scheduling activity?
5. What is a scheduling index?
6. What are the advantages and disadvantages of mathematical programming formulation of the scheduling problem?
7. How the push schedule differs from the pull schedule?

PROBLEMS
1. Consider the three partial design networks in Figure A1:

![Figure A1. Partial design networks for three products](image)

The following data is provided:

- Duration of each activity is the same and equals $t_i = 4.5$ hours, $i = 1, \ldots, 17$.
- The resource requirement of each activity is the same and equals $r_{ij} = 1$, $i = 1, \ldots, 17$;
- $j = 1, \ldots, 4$.
- Four types of resources are available in the quantity, $R = [3, 4, 5, 6]$.
The due dates for each activity equals: current time + activity number [hours], e.g.,
if the current time is 12 noon, the due date of activity 6 is 6 p.m.
The constant \( K = L = 1 \)
The weight \( w_i \) equal to the corresponding activity number, e.g., for activity 7, the weight is \( w_7 = 7 \)

At 10:35 am, activities 1, 2, 8, 12 and 13 are about to be completed.

(a) List the set of activities eligible for scheduling.
(b) Determine the set of activities to be scheduled, the set of activities to be delayed (if any) by formulating and solving an integer programming model presented in this chapter.
(c) Determine the set of activities to be scheduled with the scheduling algorithm presented in this chapter. Present all iterations of the algorithm and compare this scheduling result with the result obtained in (b).

2. Consider the design networks for two products shown in Figure A2.

![Figure A2. Design networks for two products](image)

The following data is provided:

(a) Duration of each activity is the same and equals \( t_i = .5 \) hour, \( i = 1, \ldots, 13 \).
(b) The resource requirement of each activity is the same and equals \( r_{ij} = .5, i = 1, \ldots, 13, j = 1, \ldots, 4 \).
(c) Four types of resources are available in the quantity, \( R = [2, 2, 3, 3] \).
(d) The due dates for the activities (in minutes) are as follows: \( k_i = 35, 65, 35, 60, 85, 90, 35, 65, 35, 35, 40, 60, 70 \), for \( i = 1, \ldots, 13 \).
(e) The constant \( K = L = 1 \).
(f) The weights \( w_i \) are 1, 2, 1, 3, 5, 1, 1, 7, 2, 3, 4, 2, 2, for \( i = 1, \ldots, 13 \).
(g) In addition, for product 1 only one of the two activities 1 and 3 is performed.

Perform the following:

(a) Formulate a mathematical model of the activity-scheduling problem.
(b) Modify the heuristic algorithm and solve the problem. Present all iterations of the heuristic and compare this scheduling result with the result obtained in (a).