**Algorithm for Determining Controlling Path Considering Resource Continuity**

By Mohammad A. Ammar\(^1\) and Emad Elbeltagi\(^2\)

**Abstract:** Scheduling of construction projects that have multiple units, wherein activities repeat from one unit to another, always represent a major challenge to project managers. These projects require schedules that ensure the uninterrupted usage of resources from an activity in one unit to the similar activity in the next unit and maintaining logic constraints at the same time. The scheduling method presented in this paper considers both logic and resource continuity constraints. The method utilizes the critical path method network of a single unit. Start-to-start and finish-to-finish relationships are used. Constant activity production rate is assumed. The proposed approach determines the controlling path (logically and resource critical units) in a simplified way. To automate the proposed algorithm, a macroprogram has been written on commercial scheduling software. Details of the model development and implementation are described, and an example application is presented to validate the proposed approach. The advantages, limitations, and future extensions of the proposed approach are then discussed.

**Introduction**

Repetitive construction projects represent a large portion of the construction industry. Construction projects that contain several identical or similar units are usually referred to as repetitive or linear projects. Linearity may be due to the uniform repetition of a set of activities throughout the project, or due to the physical layout of the project. Activities that repeat from unit to unit create a very important need for a construction schedule that ensures the uninterrupted flow of resources from one unit to the next. The major goals of planning and controlling linear projects, according to Reda (1990) and Arditi and Albulak (1986), are (1) maintain a target production rate for the crews employed in each activity; (2) maintain continuity of work for each crew from one unit to the other; (3) allow for time buffers between activities in the same unit; (4) allow for stage buffers between activities on different units; (5) balancing resources throughout the project; and (6) finishing the project at the minimum possible cost given a target project duration.

To maintain work continuity, repetitive units must be scheduled in such a way as to enable timely movement of crews from one unit to the next, avoiding crew idle time. Ensuring work continuity, during scheduling, provides for an efficient resource utilization strategy (El-Rayes and Moselhi 1998) that leads to (1) maximization of the benefits from the learning curve effect for each crew; (2) minimization of idle time of each crew, and (3) minimization of the off-on movement of crews on a project once work has begun (Ashley 1980).

Traditional scheduling techniques (i.e., bar chart and networks) applied to repetitive projects have been criticized widely in the literature for their inability to maintain work continuity (Selinger 1980; Reda 1990; Russell and Wong 1993). In addition, such techniques initially assume unlimited availability of resources in the development of a project schedule and through resource allocation require revision of the project schedule to comply with resource availability (El-Rayes and Moselhi 1998). Detailed criticism of traditional techniques applied to linear projects is given by Hegazy et al. (1993).

Unlike traditional techniques, resource-driven scheduling methods account directly for work continuity as well as resource availability to ensure effective resource utilization. Line of Balance is the most familiar resource-driven technique, which is used widely to schedule linear projects. It was applied to industrial manufacturing and production control with the objective of evaluating a production rate of finished products in a production line (Johnston 1981). Although variations of the original method have been proposed to suit the two categories of linear projects, all are graphically similar and provide compatible graphical representation (Arditi and Albulak 1986). Variations include Line of Balance (Lumsden 1968), Vertical Production Method (O’Brien 1975), Linear Scheduling Method (Johnston 1981), Time Space Scheduling (Stradal and Cacha 1982), and Time Changes Charts (Mawdesley et al. 1989).

One of the many apparent features of network-based techniques is the ability to specify critical path(s). This path(s) identifies activities that, if their duration changes, the project completion time changes. For linear scheduling to be accepted as a valuable tool, it must also be able to determine a set of controlling (critical) activities. Scheduling techniques applicable to linear projects must be able to provide a synonymous set of critical activities as those calculated by critical path method (CPM). This ability would provide an analytical or engineering foundation on which a full range of functionality such as float identification, resource and cost allocation, and schedule updating could be built (Harmelink and Rowings 1998). Determining critical path in traditional CPM, or controlling activity path in a linear schedule, is a crucial aspect. It helps in controlling and updating the original schedule. Any subsequent resource management (resource leveling) of a linear schedule requires as input the critical segments (Mattila and Abraham 1998).

Harmelink and Rowings (1998) introduced a method—the Linear Scheduling Model—that identifies the controlling activity path through a linear schedule based on the time and distance relationships of activities. The controlling activity path is similar to the critical path of CPM. Harris and Ioannou (1998) use a similar approach—the Repetitive Scheduling Method—to identify controlling activities of a linear schedule. However, these methods are mainly graphically based techniques, which limit their practical use. In this paper, a proposed method is introduced to schedule both repetitive and linear projects. The method utilizes the CPM network of a single unit. Start-to-start (SS) and finish-to-finish (FF) relationships are used. Constant activity produc-
tion rate is assumed. Resource continuity will also be considered. The controlling activity path(s) is determined. Variable activity production rates and/or variable activity duration will be treated in a subsequent research work. Details of procedure development are described along with its implementation on a commercial software program. An example application is then presented to validate the model and discuss future extensions.

LINEAR SCHEDULING REPRESENTATION

In the graphical representation proposed in the most previously mentioned linear scheduling techniques, repetitive activities are plotted as boxes (or lines) with constant or varying slopes (where slope represents production rate), with the axes being units versus time. The activity production rate is the inverse of the activity duration. Fig. 1 shows the common forms of linear scheduling charts, where activities are represented as boxes, whose width is the activity duration. In this case, the left side of the box represents the start times of the various units, while the right side represents the finish times. The intersection of a horizontal line, passing through a unit, indicates the duration required to finish that unit as shown in Fig. 1. In physically linear projects, especially roadway projects, the axes are reverted. In such case, the x-axis is used to denote stations or locations, while the y-axis represents time (Harmelink and Rowings 1998; Mattila and Abraham 1998).

Harris and Ioannou (1998) introduced another efficient representation for linear projects. They used the x-axis to denote time and the y-axis for repetitive units, as the traditional representation. The only difference is that activities are represented by single lines instead of boxes. Each unit is represented by two points, the first denotes activity start, while the second denotes activity finish, as shown in Fig. 2. The horizontal difference between the two points is the activity duration for that unit. The slope of a repetitive activity line represents the activity production rate. It is clear that this visualization can easily handle variable activity production rates along different units. This representation will be adopted in the present study.

LOGIC AND RESOURCE CRITICALITY

In CPM networks, a critical activity is the one that has no float. CPM calculations satisfy logical dependency between network activities. Resource availability requirement may be fine tuned subsequently, but resource continuity cannot be maintained. However, in linear projects, resource continuity must be considered. This requirement forces some noncritical (by CPM definition) activity to become critical. Some of the activities on the controlling path may be critical in the CPM sense, and some may be not. If the activity is critical, a delay in the completion of the activity delays the project. If the activity is noncritical, a delay in the activity completion does not affect project completion, but introduces discontinuities in resource utilization.

Resource continuity, therefore, shifts some activities to become critical even though they are, by CPM definition of criticality, noncritical. Harris and Ioannou (1998) referred to such activities as resource critical activities. As shown in Fig. 3, activities A1, B1–B5, C5, D2–D4, and E1–E5 are critical because of logical dependency. On the other hand, activities C1–C4 and D1 are critical only because they are scheduled to provide resource continuity. If any of them are delayed, then part of the controlling path, and consequently the project, will be delayed. Activities A2–A5 and D5 are not critical. A delay in any of these activities will not delay the project, but will cause an interruption of resource usage from unit to unit.

DEVELOPED LINEAR SCHEDULING METHOD

The development made in this paper benefits from that given by Harris and Ioannou (1998) to determine the controlling path graphically. The proposed method is used to schedule both linear and repetitive projects and to determine the controlling path. The activities production rates are assumed to be constant, and consequently duration is constant along the different repetitive units of each activity. If \( d_i \) and \( r_i \) denote unit duration and production rate of activity \( i \), respectively, then

\[
    r_i = \frac{1}{d_i} \quad (1)
\]

Most activities use several types of resources to be performed. The present method assumes that only the most significant resource will be considered (single resource) and initially requires the following data:
The developments presented in this paper were carried-out by applying the following three main steps.

**Step 1—Specifying Relationship Type**

One of the most important aspects of this procedure is the ability to specify the relationship type among different activities to maintain the resource continuity usage. To specify such relationships between two consecutive activities, the production rate of each activity is compared with that of its successors. The activity under consideration will be referred to as current activity. If \( r_c \) and \( r_s \) denote production rates of current and succeeding activities, respectively, the possible cases that may be encountered are as follows:

- **Case 1.** \( r_s < r_c \).—This implies that the start of the first unit of the succeeding activity is controlled by the finish of the first unit of the current activity. Then an SS relationship is to be specified. The lag associated with an SS relationship (Lag\(_{SS}\)) equals the unit duration of the current activity, as follows:

\[
\text{Lag}_{SS} = d_c \tag{2}
\]

where \( d_c \) = unit duration of the current activity. Referring to Fig. 3 and considering, for example, activities A and B, where B is a succeeding activity to A. In this case, \( d_c \) (activity A) = 1 and \( d_s \) (activity B) = 2.5, and consequently \( r_s \) (activity A) = 1 and \( r_c \) (activity B) = 0.4 (1/2.5). Therefore, an SS relationship exists between activities A and B and the corresponding \( \text{Lag}_{SS AB} = d_c = 1 \).

- **Case 2.** \( r_s > r_c \).—In this case, the start of the last unit of the succeeding activity is controlled by the finish of the last unit of the current activity. Then an FF relationship exists. The lag associated with an FF relationship (Lag\(_{FF}\)) equals the unit duration of the exceeding activity, or

\[
\text{Lag}_{FF} = d_s \tag{3}
\]

where \( d_s \) = unit duration of succeeding activity. For example, the relation between activities C and D in Fig. 3 depicts an FF relationship. In this case, \( r_c \) (activity C) = 0.67 (1/1.5) and \( r_s \) (activity D) = 2 (1/0.5), and the corresponding \( \text{Lag}_{FF CD} = d_s = 0.5 \).

- **Case 3.** \( r_s = r_c \).—In this case, both SS and FF relationships can be specified, with lags as in Cases 1 and 2.

Step 1 must be carried-out for activities in the network. After specifying the relationship type between consecutive activities, the duration of an activity \( i (D_i) \) is calculated as the sum of unit duration of all repetitive units, i.e.

\[
D_i = n \times d_i \tag{4}
\]

**Step 2—Network Scheduling**

Having the initial data specified and the relationship type among activities determined with their associated lags, network calculations similar to that of CPM are done. Forward path calculations are done to determine the early times of each activity, while the backward path determines the late times. Also, the critical activities are specified. The critical activities will be checked in Step 3 to identify those logic and resource critical units.

**Step 3—Determining Controlling Path**

After the timings of all activities are determined in Step 2, the critical units (logic and/or resource) can be specified based on the following general rules:

**Rules for Defining Critical Units**

<table>
<thead>
<tr>
<th>Rule number</th>
<th>Case</th>
<th>Logic</th>
<th>Resource</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( r_s &gt; r_c )</td>
<td>No predecessors</td>
<td>All</td>
<td>No predecessors</td>
</tr>
<tr>
<td>2</td>
<td>( r_s \leq r_c )</td>
<td>All</td>
<td>No successors</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>( r_s = r_c )</td>
<td>All</td>
<td>Unit 1 of predecessor is critical</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>( r_s &gt; r_c )</td>
<td>Unit n of predecessor is critical</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>( r_s \leq r_c )</td>
<td>All</td>
<td>Unit n of predecessor is critical</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>( r_s &gt; r_c )</td>
<td>( n )</td>
<td>Unit n of predecessor is critical</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>( r_s &lt; r_c )</td>
<td>All</td>
<td>Unit n of predecessor is critical</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>( r_s = r_c )</td>
<td>All</td>
<td>Unit n of predecessor is critical</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>( r_s &gt; r_c )</td>
<td>( n )</td>
<td>Unit n of predecessor is critical</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>( r_s = r_c )</td>
<td>( 1 )</td>
<td>Unit n of predecessor is critical</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>( r_s = r_c )</td>
<td>( (n+1) )</td>
<td>Unit n of predecessor is critical</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>( r_s = r_c )</td>
<td>( (n+2) )</td>
<td>Unit n of predecessor is critical</td>
<td></td>
</tr>
</tbody>
</table>

Note: \( r_c \) = production rate for current activity; \( r_s \) = production rate of its successor; \( n \) = total number of units.
on the activities production rates. The production rate of each current activity is compared with that of its preceding and succeeding activities. The different alternatives, with the critical units, are summarized in Table 1. By applying the rules given in Table 1, the controlling path can be easily determined. These rules will be demonstrated in details by the example problem discussed in the next section.

The basic steps of the developed algorithm are depicted by the flowchart given in Fig. 4. Step-by-step application of the procedure is given in the next section.

EXAMPLE APPLICATION

The procedure described in the previous section is demonstrated by an example application. This example consists of 12 activities, and each activity contains five repetitive units. The data for the example application are presented in Table 2. The example application will be solved manually by applying the procedure described earlier in the following steps:

- Determine activities' production rates $r_i$. For example, the production rate for activity $A$ using (1), $r_A = 1/12$. The activities' production rates are shown underneath the activities in Fig. 5.
- Calculate the activities' durations by multiplying activity unit duration by the number of units using (4). The activities' durations are also shown in Fig. 5.
- Draw the precedence network and specify the relationship type between different consecutive activities according to the analysis given in Step 2. Relationship type and the corresponding lag are shown in Fig. 5 along with the link between different activities.
- Perform the forward and backward paths calculations to determine the activities' timing and mark the critical activities.
- Apply the rules given in Table 1 to determine the controlling path (logical and resource critical units). To demonstrate this, three activities will be considered: $A$, $G$, and $L$. Activity $A$ has no predecessors and its production rate, $r_A = 1/12$, is less than that of its successor activity $B$, $r_B = 1/10$. Accordingly, rule 2 given in Table 1 applies, and therefore all units of activity $A$ are logically critical.

On the other hand, activity $G$ has more than one predecessor—$C$, $D$, $E$, and $F$. Because activity $F$ is the critical one, it will be considered only in the comparison with activity $G$. Activity $G$ has a production rate, $r_G = 1/5$, greater than that of its preceding activity $F (1/11)$ and equal to that of its succeeding activity $H (1/5)$. Accordingly, rule 8 in Table 1 applies, which results in units 2, 3, 4, and 5 to be logically critical and unit 1 to be resource critical.

Now, consider activity $L$ which has no successors and two critical preceding activities—$I$ and $J$. In this case, the rate of

---

**TABLE 2.** Example Application Data

<table>
<thead>
<tr>
<th>Activity</th>
<th>Duration (days)</th>
<th>Predecessors</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>4</td>
<td>B</td>
</tr>
<tr>
<td>D</td>
<td>8</td>
<td>B</td>
</tr>
<tr>
<td>E</td>
<td>6</td>
<td>B</td>
</tr>
<tr>
<td>F</td>
<td>11</td>
<td>B</td>
</tr>
<tr>
<td>G</td>
<td>5</td>
<td>C, D, E, F</td>
</tr>
<tr>
<td>H</td>
<td>5</td>
<td>G</td>
</tr>
<tr>
<td>I</td>
<td>4</td>
<td>C, H</td>
</tr>
<tr>
<td>J</td>
<td>8</td>
<td>H</td>
</tr>
<tr>
<td>K</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>L</td>
<td>10</td>
<td>I, J</td>
</tr>
</tbody>
</table>

---

**FIG. 5.** Network Analysis of Example Application

---

**TABLE 3.** Solution of Example Application

<table>
<thead>
<tr>
<th>Activity</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>LC</td>
<td>LC</td>
<td>LC</td>
<td>LC</td>
<td>LC</td>
</tr>
<tr>
<td>B</td>
<td>RC</td>
<td>LC</td>
<td>LC</td>
<td>LC</td>
<td>-----</td>
</tr>
<tr>
<td>C</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td>D</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td>E</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td>F</td>
<td>LC</td>
<td>LC</td>
<td>LC</td>
<td>LC</td>
<td>LC</td>
</tr>
<tr>
<td>G</td>
<td>RC</td>
<td>LC</td>
<td>LC</td>
<td>LC</td>
<td>LC</td>
</tr>
<tr>
<td>H</td>
<td>LC</td>
<td>LC</td>
<td>LC</td>
<td>LC</td>
<td>LC</td>
</tr>
<tr>
<td>I</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td>J</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td>K</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td>L</td>
<td>LC</td>
<td>LC</td>
<td>LC</td>
<td>LC</td>
<td>LC</td>
</tr>
</tbody>
</table>

Note: LC—logically critical unit; RC—resource critical unit.
activity L is compared with the production rate of activity I (1/4), which has a production rate greater than that of activity J (1/8). Then, rule 3 in Table 1 applies, and hence all units of activity L are logically critical.

Proceed in the same manner and in accordance with the rules given in Table 1, logical and resource critical units for all activities can be easily determined, and then the controlling path can be specified. The complete solution of this example application is given in Table 3.

As a verification of the manual solution, the timings of the activities determined in Fig. 5 are plotted to determine graphically the controlling path (logical and resource critical units) as shown in Fig. 6. It can be concluded that the logical and resource critical units are the same as given in Table 3, using the developed procedure.

PROCEDURE AUTOMATION

Implementing the proposed procedure for determining the activities' controlling path on commercial scheduling software simplifies the process and provides project managers with an automated tool to improve the results of their familiar software. In this study, Microsoft Project software (Microsoft Project 98) is selected for the implementation of the proposed scheduling procedure because of its ease of use and programmability features. Using Visual Basic for Applications macro-language of Microsoft Project, the procedure was coded and then used to solve the example application presented in the previous section.

The developed program is named Critical Path Linear Scheduling Method (CPLSM), which has a user-friendly interface. To facilitate the use of CPLSM, two command buttons were arranged in one toolbar that appears in the main menu of Microsoft Project software (Fig. 7). To demonstrate the ca-
pabilities of CPLSM, the example application at hand was input to Microsoft Project software. The user first inputs activities and their predecessors only as shown in Fig. 7. Then, proceeds by pressing the “CPM/LSM Schedule” button shown in Fig. 7, a window, shown in Fig. 8, will appear for input unit duration and number of units for each activity and store them in the database. Having entered all of the activities’ initial data, CPLSM starts the scheduling process and calculates the start and finish dates for each activity and the project completion date, as shown in Fig. 9. During this process, the developed procedure automatically calculates the activities’ duration and the correct relationship type among activities and their associated lags.

Once the scheduling process was completed, the “Show Critical Units” button on the toolbar (Fig. 7) is activated and enables the user to view all logical and resource critical units. Pressing the Show Critical Units button, a window, shown in Fig. 10, appears and the user can easily view the status of different units of any activity using the right or left arrows of the scroll bar. For example, the status of activity 1 (Fig. 10) shows that unit 1 is resource critical and units 2 to 4 are logically critical.

COMMENTS AND FUTURE EXTENSIONS

The method presented in this paper has been demonstrated to work effectively on the example application. Various experiments were also conducted on various other problems and CPLSM performed well. Based on its performance, the main features of CPLSM that make it an efficient tool for scheduling repetitive project are that

- It satisfies an important feature of scheduling repetitive projects by maintaining the resource continuity and logical relationships.
- It schedules repetitive projects in an easy nongraphical way.
- It needs a small amount of data to perform the required analysis. Only activities, their predecessors, unit duration, and number of repetitive units are required as input data.
- It defines automatically the relationship type among activities (SS and/or FF) and calculates the lag among them.
- It determines the logically critical and resource critical units in each activity.
- It has been implemented on a commercial scheduling software that is customary to many practitioners in construction management.
- It provides the user with a detailed report of the critical units in each activity.

Despite its important benefits, there are a number of possible extensions currently being pursued by the writers. These include

- Changing the number of repetitive units from one activity to another.
- Allowing for variable production rate within the same activity.
- Determining the schedule start and finish dates for each unit and drawing a bar chart for all units (each unit will be treated as a single activity).
- Use the controlling activity path as a base to level resources for repetitive projects.

SUMMARY AND CONCLUSIONS

This paper presented an effective algorithm for scheduling repetitive and linear projects. The major benefits of the proposed approach are the ability to perform network scheduling considering logic constraints while satisfying resource continuity and to define the controlling path (logically and resource critical units). A structured manual approach was presented and used to solve an example application to demonstrate the effectiveness of the proposed model. A computer program was written using Visual Basic for Applications Macro Programming Language to automate the proposed approach and to provide a good interface and reporting capabilities. The procedure was implemented on a commercial scheduling software to uti-
lize its familiar interface and programmability features. The advantages and future extensions of the system are also outlined.

REFERENCES


