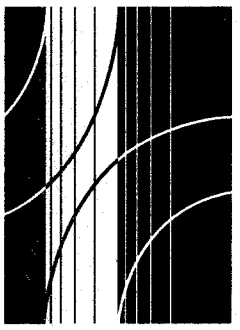


Joint Transportation Research Program

JTRP

FHWA/IN/JTRP-2000/21

Final Report

DEVELOPMENT AND APPLICATION OF LINEAR SCHEDULING TECHNIQUES TO HIGHWAY CONSTRUCTION PROJECTS

David J. Harmelink
Rene Antonio Yamin

October 2000

Indiana Department of Transportation

Purdue University

FINAL REPORT

FHWA/IN/JTRP-2000/21

DEVELOPMENT AND APPLICATION OF LINEAR SCHEDULING TECHNIQUES TO HIGHWAY CONSTRUCTION PROJECTS

by

David J. Harmelink, PhD
Assistant Professor
and
René Antonio Yamín, MSCE
Graduate Research Assistant

Division of Construction Engineering and Management
School of Civil Engineering
Purdue University

Joint Transportation Research Program

Project No. C-36-67YY
File No. 9-10-50
SPR-2330

In Cooperation with the
Indiana Department of Transportation and the
U.S. Department of Transportation
Federal Highway Administration

School of Civil Engineering
Purdue University

October 2000

TECHNICAL REPORT STANDARD TITLE PAGE

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
FHWA/IN/JTRP-2000/21		

4. Title and Subtitle	5. Report Date
Development and Application of Linear Scheduling Techniques to Highway Construction Projects	October 2000
	6. Performing Organization Code

7. Author(s)	8. Performing Organization Report No.
David J. Harmelink and Rene Antonio Yamin	FHWA/IN/JTRP-2000/21

9. Performing Organization Name and Address	10. Work Unit No.
Joint Transportation Research Program 1284 Civil Engineering Building Purdue University West Lafayette, Indiana 47907-1284	
	11. Contract or Grant No. SPR-2330

12. Sponsoring Agency Name and Address	13. Type of Report and Period Covered
Indiana Department of Transportation State Office Building 100 North Senate Avenue Indianapolis, IN 46204	Final Report
	14. Sponsoring Agency Code

15. Supplementary Notes

Prepared in cooperation with the Indiana Department of Transportation and Federal Highway Administration.

16. Abstract

Crucial to the successful outcome of major highway construction and reconstruction projects today is the ability to accurately plan, predict and control the construction process. Evermore sensitive to budget control, schedule control, resource allocation and impacts on the motoring public, highway constructors today require sophisticated project management tools to achieve project goals. One of the tools recently being used across the country is project scheduling. The use of scheduling techniques on highway projects has grown out of the successful application of these methods to building construction for the past 20 to 30 years. Unfortunately, the effective application of traditional scheduling techniques to highway construction has been limited because major highway construction project activities are fundamentally different than those typically found on a building project.

Major work activities on typical highway construction/rehabilitation projects are linear activities. Unfortunately, currently accepted scheduling techniques (Critical Path Method and bar charts) are unable to accurately model projects consisting primarily of linear work. Contractors and transportation officials are increasingly frustrated with CPM's inability to provide relevant planning and project management information. A recently rediscovered technique called Linear Scheduling coupled with advances in computer technology and software has the potential to provide significant advancement to highway construction project scheduling and management. Further research is necessary however, to advance the underdeveloped Linear Scheduling technique to the point of actual implementation.

A method of producing linear schedules for use in planning and managing suitable highway construction projects, is provided. The ultimate product of this research is a Linear Scheduling Tool (PULSS – Purdue University Linear Scheduling Software) comprised of methods, procedures and software tools that allow for implementation of the Linear Scheduling Method. Furthermore, this software is able to:

- Allow schedulers to visually plan highway construction projects
- Calculate the controlling activity path of such schedules
- Be able to print reports of the status of the schedule

17. Key Words	18. Distribution Statement
Highway construction, Project management, Project Control, Linear projects, Linear Scheduling, Software application.	No restrictions. This document is available to the public through the National Technical Information Service, Springfield, VA 22161

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price
Unclassified	Unclassified	78	

Form DOT F 1700.7 (8-69)

TABLE OF CONTENTS

TABLE OF FIGURES

Implementation Report

Crucial to the successful outcome of major highway construction and reconstruction projects today is the ability to accurately plan, predict and control the construction process. Evermore sensitive to budget control, schedule control, resource allocation and impacts on the motoring public, highway constructors today require sophisticated project management tools to achieve project goals. One of the tools recently being used across the country is project scheduling. The use of scheduling techniques on highway projects has grown out of the successful application of these methods to building construction for the past 20 to 30 years. Unfortunately, the effective application of traditional scheduling techniques to highway construction has been limited because major highway construction project activities are fundamentally different than those typically found on a building project.

The predominant technique used in building construction today is the Critical Path Method (CPM). This technique has evolved over the past several decades into highly sophisticated and computerized applications, which only recently have been applied to major highway construction projects. Linear scheduling techniques have been in existence long before CPM but have not received the same amount of attention and development effort. Within the last ten years however, as the need to effectively repair and rebuild major transportation infrastructure, linear scheduling techniques are again being developed.

Major work activities on typical highway construction/rehabilitation projects are linear activities. Unfortunately, currently accepted scheduling techniques (Critical Path Method and bar charts) are unable to accurately model projects consisting primarily of linear work. Contractors and transportation officials are increasingly frustrated with CPM's inability to provide relevant planning and project management information. A recently rediscovered technique called

Linear Scheduling coupled with advances in computer technology and software has the potential to provide significant advancement to highway construction project scheduling and management. Further research is necessary however, to advance the underdeveloped Linear Scheduling technique to the point of actual implementation.

The overall objective of this work is to provide INDOT and contractors with a method of producing linear schedules for use in planning and managing suitable highway construction projects. The ultimate product of this research being a Linear Scheduling Tool comprised of methods, procedures and software tools that allow for implementation of the Linear Scheduling Method. Furthermore, this software is expected to:

Allow schedulers to visually plan highway construction projects
Calculate the controlling activity path of such schedules
Be able to print reports of the status of the schedule

This research provides INDOT and contractors with PULSS (Purdue University Linear Scheduling Software). PULSS is a fully functional prototype software that allows the visual planning of highway construction based on the linear scheduling method.

Contribution of the study include the following:

Complete code for each of the subroutines and programs contained in PULSS.
Installation instructions of the software
Practical applications of the prototype in two INDOT projects in Indiana
Positive feedback from contractors regarding the utilization of the method.

In addition to PULSS, this research provides INDOT with insights into contractors opinion regarding the utilization of linear scheduling for project management and control. With PULSS, INDOT is able to further develop and integrate the software with its current project management tools improving further the existing methods for cost analysis and control.

However, for full implementation of PULSS additional "modules" have to be developed, and hence, further research is suggested. One of the proposed modules is the Risk assessment module, which would allow project planners to estimate the risk of project delay based on existing productivity rates. This additional capability will provide the linear scheduling method with statistical analysis tools comparable to those of CPM.

I. Introduction

Crucial to the successful outcome of major highway construction and reconstruction projects today is the ability to accurately plan, predict and control the construction process. Evermore sensitive to budget control, schedule control, resource allocation and impacts on the motoring public, highway constructors today require sophisticated project management tools to achieve project goals. One of the tools recently being used across the country is project scheduling. The use of scheduling techniques on highway projects has grown out of the successful application of these methods to building construction for the past 20 to 30 years. Unfortunately, the effective application of traditional scheduling techniques to highway construction has been limited because major highway construction project activities are fundamentally different than those typically found on a building project. Where most of the activities on a building project are discrete, sequentially-related activities, activities on a highway project are linear in nature and related spatially. Major activities proceed along the path of the project separated by some reasonable distance. Delays occur when there are conflicts in the space occupied by adjacent activities, not necessarily by the duration of the activities.

The predominant technique used in building construction today is the Critical Path Method (CPM). This technique has evolved over the past several decades into highly sophisticated and computerized applications, which only recently have been applied to major highway construction projects. Linear scheduling techniques have been in existence long before CPM but have not received the same amount of attention and development effort. Within the last ten years however, as the need to effectively repair and rebuild major transportation infrastructure, linear scheduling techniques are again being developed.

The Iowa Department of Transportation (IDOT) has commissioned three research projects over the last six years on the application of linear scheduling

techniques to highway construction projects. Other DOT's have also initiated similar projects trying to assess the implementation potential of LS, as well as ways to computerize the method (Florida, Pennsylvania and Virgina). In addition, currently at Purdue a method of leveling resources in a linear schedule and the prediction of productivity rates for non-stationary linear activities using simulation have been developed.

Within the Indiana Department of Transportation today, the use of advanced scheduling techniques is relatively low. The current specification only requires contractors to provide a Gannt (bar) chart. A survey performed for IDOT in 1993 indicated that nearly all states have scheduling specifications. These specifications typically require the Contractor to *furnish* the Engineer with a "*Progress Schedule*", typically using CPM, showing the order of the work and the time required for the completion. This schedule would be used to establish *major construction operations* (or *salient features* or *controlling items*) and to *check progress*. As the use of CPM schedules become more widespread however, there appears to be a growing concern, as indicated in the IDOT projects, as to the ability of the critical path method to accurately model linear type activities. The only means by which a linear activity can be modeled by CPM is to artificially break the activity up into a sequential series of shorter activities. Unfortunately, every time an activity is divided the logical relationships needed to model the activity increase as well. Within a very short time the number of activities and relationships becomes so complicated and cumbersome to manage that the value-added by the schedule is significantly impacted. On the other hand, if the activities are not subdivided the natural variations in productivity of linear activities over significant distances cannot be interpreted by the CPM schedule.

Certainly there must be an alternative to using CPM to model linear projects. Linear scheduling provides a means of modeling linear activities as a continuous set of points on a line rather than discrete events. Although severely underdeveloped when compared to CPM, the ability to model linear activities as

a continuous set of points provides significant advantages over CPM. For example, the productivity can vary in any way necessary to accurately model the expected productivity of an activity. CPM requires that the productivity of any discrete activity is constant. Imagine a PCC paving operation moving away from a concrete batch plant using a fixed number of trucks. At some distance from the batch plant, the number of trucks will affect the rate of the paver. As the paver continues to move away from the batch plant the production rate will continue to diminish. This effect, the non-linear rate, and the affect it has on subsequent linear activities is impossible to model using CPM.

I.1 Problem Statement

Major work activities on typical highway construction/rehabilitation projects are linear activities. Unfortunately, currently accepted scheduling techniques (Critical Path Method and bar charts) are unable to accurately model projects consisting primarily of linear work. Contractors and transportation officials are increasingly frustrated with CPM's inability to provide relevant planning and project management information. A recently rediscovered technique called Linear Scheduling coupled with advances in computer technology and software has the potential to provide significant advancement to highway construction project scheduling and management. Further research is necessary however, to advance the underdeveloped Linear Scheduling technique to the point of actual implementation.

I.2 Objective of the Study

The overall objective of this work is to provide INDOT and contractors with a method of producing linear schedules for use in planning and managing suitable highway construction projects. The ultimate product of this research being a Linear Scheduling Tool comprised of methods, procedures and software tools that allow for implementation of the Linear Scheduling Method. Furthermore, this software is expected to:

Allow schedulers to visually plan highway construction projects

Calculate the controlling activity path of such schedules

Be able to print reports of the status of the schedule

As part of the study the researchers will test the prototype software with different Indiana highway contractors. From this collaboration researchers will be able to identify the shortcomings of the prototype as well as practical enhancement needed for a fully working product.

II. Prior research and DOT work

Although the existence and use of linear schedules (also called string diagrams by the English) by French engineers can be traced as far as the 1800's (Ibry), and European and Arabic countries have been using it for different purposes (mostly train scheduling and construction management), linear schedules have not been very popular among US engineers.

In recent years however, renewed interest by different department of transportations (DOT) in the US have rediscovered Linear Schedules (LS) and have initiated through multiple research a process of exploration for computerization and application potential of the LSM. Through these research projects DOT's want to determine if LS tools are at a level of development so that they can be used to assist in project planning and control, and hence they could be required as part of the specs for Highway construction. Following a brief summary of the most resent and relevant studies is shown:

II.1 Iowa DOT (1993, 1994 & 1995):

The Iowa DOT required Critical Path Methods (CPM) schedules on some larger or more schedule sensitive projects; however, their expectation for enhanced project control and improved communication of project objectives was not fully met by the CPM. As a result, the Iowa DOT commissioned Prof. Rowings of Iowa state center for engineering to perform two different research projects on scheduling for highway construction in the state. The first research project had the following objectives:

To evaluate the existing state-of-the-art scheduling techniques used by other states,

To develop a new or improved methods that will enable Iowa DOT to determine a reasonable contract duration,

To develop a method to monitor the project progress more accurately, and

To develop procedures to objectively evaluate the time impacts of changes and extra work.

As a result of this project, the researchers identified the Linear Scheduling Method as an appropriate alternative scheduling method to the CPM for highway construction projects. As part of the study researchers evaluated the LSM in a diverse group of project, concluding that the LSM held great potential for project management applications in the highway construction area, particularly in those projects were the most important activities were linear and had variable production rates.

As a follow-up study a second research project was started. This second project was directed to evaluate in greater detail the real potential of LSM as a project management technique to be used in the state projects. For this purpose three projects were identified and researchers were involved early in the planning stage throughout the completion of the project. It was anticipated that by applying LSM to several projects from early in the planning phase through the construction of the project, that the technique could be refined to meet the needs of the users in the following ways:

The benefits LSM may present for Iowa DOT personnel in the areas of understanding the contractors "plan of attack" and also being able to monitor actual work progress would be indentified.

A determination of how construction managers would use LSM to manage their projects would be made. The necessary information, and form of the information, required by managers to function effectively would be identified.

A consistent set of symbols to describe the various types of activities involved in a typical highway construction project would be developed.

A comparison of how CPM is used by Contractors and IDOT currently with how LSM may be used would be made.

As part of the study researchers produced as-built schedules of the projects and gather impressions by both contractors and IDOT inspectors on the performance and usability of the LSM.

All objectives of the study were accomplished and it was confirmed that the visual capabilities of the LSM provided both Contractors and IDOT personnel a clear insight about linear activities that could not be achieved by using CPM.

Researchers concluded that the LSM had great potential as a project management tool for both contractors and IDOT personnel. However, it was also commented that the elaboration of LS was cumbersome and that a computerized tool for creating these schedules was needed for it to be really valuable and widely accepted by Contractors. Such software tool should bring to LS a degree of functionality as rich and as comprehensive as that found in micro computer based CPM software on the market at the time.

For this implementation algorithms needed to be developed that could provide the information necessary to successfully meet highway construction management needs. Some of the tasks that the application will need to perform are:

Determine and identify the critical path based on activity logic and location.

Reconcile the various project calendars.

Provide a library of resource that can be used to determine production rates and equipment requirements to meet planned production rates.

Allow for input of as-built information graphically or through a spreadsheet format which imitates the inspector's progress reports.

Track progress against the baseline schedule and display graphically and in text-based reports the schedule variance of completed and in-progress activities, and the anticipated activity dates of activities not completed.

Finally, the research team recommended IDOT to extend the research to develop a LS application with the abovementioned capabilities.

II.2 Florida DOT (1999):

Herbsman and Glagola (1999) performed a study for the Florida DOT, in which they found through a survey to different DOTs that out of thirty seven (37) respondents, sixty-five percent (65%) were not familiar with the LSM; twenty five (27%) were somewhat familiar and only three (3) DOT's (constituting 8%) were very familiar with this scheduling method. In addition, researchers found that although the LS is considered by contractors more intuitive and easy to use, they have not been successfully adopted in the highway construction industry for basically the following reasons:

Government agencies (as owners) fail to recognize the benefits of linear scheduling and adopt its use.

Computerization of the method

Limitations in functionality when compared with network based scheduling tools (Resource allocation, resource leveling, critical path calculations, float, and completion time confidence interval estimation)

As part of the study researchers developed a software application called Florida Linear Scheduling Program (FLSP), which allows schedulers to create LS using a set of screens done in VBA for Access. The prototype allows user to introduce general information about the project (start/end dates, name of project, working days). Other information is then introduced in a sequence of screens that prevent users from scheduling physically conflicting activities. For scheduling activities of the project, users are given three different data input options:

Start and Finish dates are known

Productivity Rate is known

S-Curve

For every input option, different resource selection procedures are offered. The software can then present usage histograms for a particular resource, as

well as, production rate charts. In addition, users can access a database for standard production rates.

The output capabilities of the FLSP are:
Linear schedules graph
Resource histogram
S-curves

Although the FLSP handles resource usage and provides users with an S-curve to monitor the financial progress of the project, the program has the following identified shortcomings:

It does not fully exploits the visual planning capabilities of Linear Schedules. Instead, the user has to perform an iterative process introducing activity information using dates and stations without been able to see its physical layout and how it is compared to other activities.

The program does not calculate the controlling activity path (CAP); hence estimating which activities are controlling is not possible, failing to incorporate one of the most important and needed features of the LSM.

Representation of the project is done placing the stations in the Y axis and time in the X axis. This representation resembles the Line of Balance representation used in manufacturing scenarios, and does not provide users the plant view of the project which provides with visual clues for improved planning and control

As conclusions of the study, researchers ratify the linear scheduling technique as a superior technique for planning and controlling highway construction projects, as well as, a valuable tool for prove or disprove claims.

II.3 PennDOT

PennDOT recently funded a study called "Comparison Of Critical Path Method (CPM) Of Scheduling and the Linear Scheduling Method (LSM)". According to Prof. H. Randolph Thomas, "the objective of this work order is evaluate the suitability of the LSM for use on large capital projects and compare

the results from the LSM to the CPM schedule developed contractually for several sections of I99. The criteria for comparison is to determine if the LSM monitors schedule progress better than the CPM schedule and if the graphical visualization of the LSM schedule shows physical constraints in a way that will permit a shortening of the overall project schedule. The work plan in general will involve tracking the bi-weekly contract CPM schedules on two sections of I99. A parallel LSM schedule will be developed for comparison. The input data for the two schedules will be obtained from the bi-weekly progress meetings."

As part of this research, Penn State sought cooperation from Purdue University researchers to create the initial LS schedules and its updates. These will be continuously compared against the CPM schedules.

II.4 Other Research

In addition to the research directly funded by different DOT's in the US, there are several academics working towards solving some of the issues faced by the LSM to achieve wide acceptance. These advances have taken the LS from a graphical tool to a more analytical and CPM-like type of scheduling tool (Harmelink 1995, Mattila 1997 and Shu-Shun 1998), as well as, development of software applications of LS (Harmelink 1995, Harmelink & Yamin 2000 with the PULSS, Herbsman and Glagola (1999) with the Florida Linear Scheduling Program (FLSP)) that offer features comparable to those of the CPM.

III. CONTROLLING ACTIVITY PATH – LSM ALGORITHM

One of the most significant advancement made for towards a wide acceptance and use of linear schedules was the Linear Scheduling Model (LSM) developed by Harmelink (1995). The model identifies and analyzes seven types of activities that can occur on a linear schedule identifying a set of those activities that are controlling activities. Other, or non-controlling, activities have float similar to that defined in CPM, but in the LSM it is called Rate Float. The LSM also provides a means of tracking progress to allow statusing and updating linear schedules to predict future completion dates.

The software application PULSS automates the LSM algorithm allowing users to visually plan highway construction projects, and calculate the controlling activity path (CAP) of such schedule. Following is a brief summary of the LSM components and steps needed to calculate the CAP in LS.

III.1 Activity Types

To understand the procedure for determining the controlling activity path in a linear schedule it is necessary to become familiar with the different types of activities that can appear on a linear schedule. Previous research has suggested that there are three types of activities that can appear in a linear schedule: linear, block, and bar (Vorester et al. 1992). The LSM retains the three basic activity types but refines linear and block activities into specific subtypes. Linear activities are divided into the following four specific subtypes:

Continuous full-span linear (CFL)
Intermittent full-span linear (IFL)
Continuous partial-span linear (CPL)
Intermittent partial-span linear (IPL)

The block-type activity is divided into the following two types:
Full-span block (FB)

Partial-span block (PB)

These subtypes relate to whether or not the activity spans the entire location of the project and whether or not the activity is in continuous or intermittent operation. Fig 3.1. shows all of the activity types as they would appear on a linear schedule. The procedure to determine the controlling activity path in a linear schedule involves three steps:

Determine the activity sequence list

Perform the upward pass

Perform the downward pass

III.2 Activity Sequence List

The activity sequence list (ASL) identifies all of the possible logical sequences through the activities on a linear schedule. The controlling activity path is defined as the continuous path of longest duration through the project and defines the sequence of activities that must be completed as planned to finish the project within the overall planned duration. Generally, the activity sequence with the longest duration (or the least free time) contains all of the activities on the controlling activity path. There are some exceptions to this, and will not be discussed in detail in this paper.

Although not a necessary step in actually determining the controlling activity path, the activity sequence list is fundamental to understanding the process of determining the controlling activity path in a linear schedule and also provides a basis for the development and implementation of the LSM algorithm in CAD environments.

The activity sequence list must describe the order in which activities will occur at any location on the project. Fig 3.2 shows a simplified example of a typical highway construction project linear schedule. This project has three CFL activities: A, C, and G. Also assume that there is a "0" time CFL activity at the

beginning (start) and at the end (end) of the project. Intermediate activities, any activities that are not CFL activities, will always lie between two CFL activities. For example, activity B, a PB, and activity C, two bars (B), lie between CFL activities A and D. The activity sequence list must describe the activity sequence through these activities, regardless of location. Fig. 3 shows the logical sequence of activities for this project. Notice, that for the CFL activities start, A, D, G and end, there cannot be multiple logical paths because these activities span the entire project. However at intermediate locations between CFL activities, there may be multiple paths. The possible activity sequences can be determined by examining the order of activities at any possible location (vertical line) between the CFL activities. The vertical dashed lines on Fig. 3.2 show the possible activity sequences through this sample linear schedule. The five possible paths are represented schematically in Fig. 3.3 and the activities involved in each path are listed. As in CPM scheduling, the goal is to find the longer continuous path through this sequence of activities. This path will define the controlling activity path and determine when and where activities are controlling.

III.3 Upward Pass

The goal of the upward pass is to determine which activities or portions of activities could potentially be controlling. The process starts with the beginning of the project and progresses upward, identifying the path with the least free time between each pair of continuous full-span activities. In each case the activity for which the potential controlling segment is being determined is designated the origin activity. The origin activity will always be a CFL activity, and the earliest point in time on this activity is designated as the origin. The next CFL activity in the sequence list will be the target activity. The potential controlling segment of the origin activity can be determined by examining the relationship between these two activities. All activity types other than CFL activities are examined with respect to the upper- and lower-bounding CFL activities.

The following three specific elements must be determined to describe the relationship between these two activities:

Least-time (LT) interval. This is the shortest time interval between any two consecutive activities. Consecutive activities are activities that can be connected in time (vertically) without crossing another activity. The LT interval will always occur at a vertex of one of the activities in question. Vertices occur at the end points or anywhere the slope changes on a linear activity, at the corners of box activities, and at the endpoints of bar activities.

Coincident duration. This is an interval in the time during which the two activities connected by the LT interval are both in progress.

Least-distance (LD) interval. This is the shortest distance between the two activities that lies within the coincident duration interval and intersects the LT interval.

Figure 3.4 shows an example of a pair of CFL activities for which the LT, coincident duration, and LD have been identified.

In this step, the origin activity is always viewed as if it were the first activity in the activity sequence. Therefore, the segment from the origin up to where the target activity begins will always be a potential controlling segment. Somewhere during the time after the target activity begins and the origin activity ends, there is a line along which the controlling activity path must occur. It is logical to assume that this path will occur where activities are closest to each other. The LD interval describes the location at which this closest point occurs. Once the LD interval has been determined, the point of intersection with the origin activity is called the critical vertex as shown in Figure 3.5. The segment of the origin activity between the origin and the critical vertex is a potential controlling segment for this activity, and the LD interval becomes a potential controlling link between the origin and target activities. The determination of which portion of the potential controlling segment is actually controlling is determined in the downward pass. The target activity in this step of the upward pass becomes the origin activity for the next step and the process repeats until all the potential controlling activity segments have been determined. Figure 3.6 identifies the potential controlling segments in

a linear schedule comprised of only CFL activities. The actual controlling activity path may contain all or part of the potential controlling path but it will not contain any other activity segments.

III.4 Downward Pass

The purpose of the downward pass is to determine which portions of the potential controlling segments are actually on the controlling activity path. This step can be compared with the backward pass used in CPM scheduling, which identifies activities that do not have float. In the case of linear activities on a linear schedule, the backward pass identifies segments of activities for which the production rate cannot decrease without extending the duration of the project. This also means that segments of activities not on the controlling activity path have rate float. An activity that has rate float can progress at a slower rate than planned without affecting the duration of the project.

To perform the downward pass, start with the end of the last activity on the project. In this example, that point would be the end of activity F, as shown by point 1 in Figure 3.7. Next, follow activity F back in time, or downward on the linear schedule, until the potential controlling link with activity E is reached, as shown by point 2 in Figure 3.7. The segment identified by these two points determines the portion of activity F that is a controlling segment and the potential controlling link is now a controlling link. A controlling link is a point in time where the controlling activity path changes from one activity to another. Move horizontally to activity E along the critical link beginning at point 2 on Figure 3.7, and repeat the process performed on activity F to find the controlling segment on activity E. If, while moving back in time along an activity, the beginning of the activity is reached before a potential controlling link with a preceding activity is reached, a new critical link is established at the beginning of the activity. This is illustrated by the new critical link established with activity A at point 3 on Figure 3.7. Repeat the process of identifying critical segments and critical links by

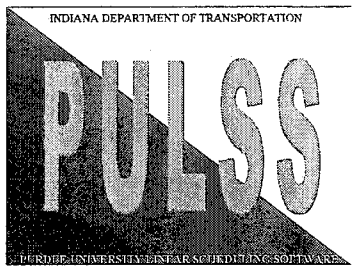
moving downward through the linear schedule until the start of the project is reached.

Once the downward pass is complete, the controlling activity path through the linear schedule has been identified. Notice, on Figure 3.7, that each controlling segment has endpoints identified. These endpoints describe when and where (at what time and location) the activity will become a controlling activity and when and where it will cease to be a controlling activity.

IV. Purdue University Linear Scheduling Software (PULSS)

IV.1 PULSS Description

PULSS (v1.0) is a prototype software that serves as a proof of concept for the computerization of the LSM algorithm that determines the controlling activity path in a linear schedule. It was developed within a Computer Aided Design (CAD) environment to take advantage of the visual capabilities of Linear Schedule and offer the user a friendly and simple way to visually plan linear schedules of highway construction projects.



In general, CAD applications provide the graphical environment in which engineering and architectural models can be created. These models are drawn as graphical entities (lines, points, circles, etc) that are abstractions of physical objects. These entities can be drawn in a two-dimensional (X, Y) or a three-dimensional (X, Y, Z) coordinate system space. Looking at dimensions from a broader perspective, any of the XYZ axes can also be used to represent attributes other than distance, such as time.

Linear schedules represent construction activities in a two-dimensional coordinate system of time and space. By converting one the XYZ dimensional axes to time coordinates, CAD can be used to model linear construction activities. Computerized spreadsheets convert dates in Gregorian calendar format (mm/dd/yyyy) into integer numbers by counting the elapsed days since some point in time, typically January 1, 1900. The "elapsed day" calendar is referred to as a Julian calendar. For example, a normal date in Gregorian calendar such as October 1, 1998, is represented in a Julian calendar as an integer number 36,017, the number of days elapsed since 01/01/1900. This integer date is then used as the Y coordinate of all points in the drawing

To implement the LSM algorithm in CAD, the Y coordinate is considered as the time dimension (date) and the X coordinate the spatial dimension (location). Time and space are only two of the attributes of an activity in a linear schedule. CAD can represent other attributes as well. Table 1 shows the relationship between a construction activity's attributes and the CAD environment.

Activity Characteristic	CAD equivalence
Name of Activity	LAYER name of object
Start/End Location	X coordinate in drawing
IV.2 Start/End Date	Y coordinate in drawing
Duration	Delta Y coordinate
Productivity	Slope of line between Start and end point
Resources	Metadata (extended data)

(i) Table 4.1. Attribute Relationships

Name of activity: A fundamental property of entities in CAD is the layer on which they exist. Entities on these layers can be manipulated in various ways such as making them visible or invisible, changing their color, or a number of other attributes. In LSM, the entities that represent a construction activity exist on a unique layer with the activity name.

Start/End location: X coordinates from entities representing activities are equivalent to location of the activity in the project.

Start/End Date: Y coordinates from entities representing activities are equivalent to dates represented by Julian calendar dates (integer numbers)

Duration: Activity durations are determined by the difference in Y coordinates of the start and end points of activities.

Productivity: Measured in units of space per units of time (feet/day) is represented by the slope of linear activities.

Resources: Metadata, also called entity extensions, are attributes that can be associated with particular entities in a drawing. In the proposed framework, this metadata field can be used to include the number and type of resources that the activity consumes for its completion.

Modeling construction activities in the manner described above provides the basis for the implementation of the LSM algorithm that calculates the Controlling Activity Path in a linear schedule. CAD environments provide comprehensive sets of functions to manipulate entities. These functions along with the development environment included in CAD packages can be exploited to implement the LSM algorithm briefly described on chapter III of this report. Depending on the particular CAD package, routines can be written in a variety of programming languages, such as Visual C, Visual Basic and different flavors of LISP. These development tools and functions allow the manipulation of graphical entities, selection of objects according to different layers or positions in the drawing space and distance calculations between different objects to name a few.

IV.3 How to use PULSS

PULSS allow users to visually create linear schedules and most importantly, automates the calculation of the controlling activity path. The whole process from creating a LS to calculating the CAP is done in four steps:

Step 1 - General Project Information: The user has to introduce general information about the project (start date, start station, end station, working days, etc). This is done by clicking the layout button () from the LSM menu bar. The user is prompted with the layout screen (Figure 4.1). After this information is introduced the layout for the linear schedule is created as shown in Figure 4.2.

Step 2 – Create activities: In this step the user introduces information for each activity to be scheduled. This process is started by clicking on the Create Activity button () on the LSM menu bar. The Create activity screen is shown and information about the activity can be introduced through the dialog box shown in Figure 4.3. Users can select the type, length and direction of the activity (linear, full-span from right to left). Activity duration can be introduced in three different ways:

Providing the total duration of the activity

Providing start and end dates for the activity

Providing the productivity rate of the activity. If the activity has multiple productivity rates, this can be accommodated and the user is prompted with the screen for this purpose (Figure 4.4).

Multiple production rate: Multiple productivity rates can be indicated by defining the different start/end dates and stations, or by providing productivity rates directly.

Step 3 – Activity Sequence List Calculation: After all the activities are drawn into the schedule layout, the Activity Sequence List has to be calculated. For PULSS v1.0 this process is not automatically done and the user has to click on the Calculate ASL button (). PULSS will indicate when the process has been performed (Figure 4.5). Once the ASL is calculated users can proceed to perform the CAP calculation.

Step 4 – Controlling Activity Path Calculation: Once the ASL is calculated the user can click on the Calculate CAP button (), and PULSS will draw the CAP for the schedule in color red using a different layer named "controlling" (Figure 4.6).

IV.3 Programming details

Operating System, Environment and languages

Operating system: Windows – Most common platform and approved by INDOT

CAD program: AUTOCAD r14 with ARX and VBA support. Although INDOT has selected MicroStation as their preferred CAD package, by the time this research project was initiated AUTOCAD was being extensively used by INDOT. In addition, AUTOCAD is widely adopted by contractors and engineers, and ultimately PULSS is designed to be used by contractors and not so much by INDOT.

Programming Language: For the full implementation of the LSM algorithm, different programming languages were used:

LISP: List processing language is a very efficient language for AutoCAD programming and automation of tasks. In addition, there are several LISP editing tools that aid in the programming and debugging of the application. For PULSS development the LISP editor used was VISUAL Lisp (VLISP), which provides editing, compiling for LISP and Object ARX code.

Dialog C Language: Some of the initial development of user interfaces was developed in DCL language which is a compatible, yet primitive, user interface design language.

ObjectARX: Is an API language for AutoCAD. Its syntax is similar to C++ but it is specifically made for AutoCAD. Its name comes from AutoCAD Runtime extension.

Visual Basic Applicaion (VBA): Different macros were developed in Visual basic application for AutoCAD Release 14 has full support and functionality of such language offering increased versatility and easier user interface design. As a disadvantage, protection of the code for the macro is weak.

IV.4 Flow Diagrams and Code

As part of the programming effort for the PULSS, a set of flow diagrams of the LSM algorithm were elaborated so that future improvements and modifications could be easily achieved. In addition, all coding was done in a

structured format with detailed description at the beginning of each subroutine or macro. All flow diagrams and codes are included in Annex 1.

IV.5 Files and Installation procedure

The installation process for PULSS v1.0 is still a rather manual one, certain files have to be directly copied onto different directories of ACADR14, and in order to create a "desktop icon" some manual alteration of the ACADR14 icon have to be done.

For PULSS installation the following files most be copied as shown in the following table:

Directory	File
ACADR14/	ACADR14.lsp
	ACAD.rx
	ACAD.dvb
	PULSS.arx
	ASL.bmp
	ACT.bmp
	CAP.bmp
	LAYOUT.bmp
	PULSSlogo.bmp
ACADR14/SUPPORT/	ACAD.mnu
	LAYOUT.lsp
	LAYOUT.dcl
ACADR14/TEMPLATE/	LSMPROTO.dwt

IV.6 Known limitations and further improvements

PULSS cannot calculate CAP is there is a time period where there is no activity being executed in the project. It is recommended that if such time of no-activity is represented by a BAR activity.

PULSS does not differentiate between intermittent and continuous activities. All activities are considered continuous for the CAP calculation.

V. Alfa tests of PULSS

As part of any software development project, prototypes of the program have to be used by user that could be considered "typical users" of the future application. For this project several contractors were contacted but finally only two accepted to dedicate some time to provide us with information about their projects. The results of this "alfa-test" is shown as follows:

V.1 I-465 with Berns construction

The first company contacted, Berns construction of Indianapolis, provided valuable information for testing PULSS. Several sites visits and scheduling meetings were held both at the offices and job-site office. From these meetings researchers could understand how schedulers plan the work and what are their needs. Some test schedules were elaborated and reviewed. From these meetings contractors offered the following valuable comments:

The linear schedule should include (allow) the possibility of considering the width of the lanes being paved. Space restrictions directly affect productivity and the way the job is scheduled. By including this "third" dimension scheduling jobs will be more realistic and will provide contractors with a greater value added over other conventional methods.

A report in spreadsheet format is necessary in order to execute the schedule.

Include multiple productivity rates in schedules.

Prototype is not very user-friendly.

In addition to their direct comments, researchers learned that the company uses plan view of the highway to be built (showing only certain landmarks and the number of lanes). Scheduling was done mainly by superintendents, and

some coordination was obtained from the scheduling engineering. The method used is simple and easy to communicate. No controlling path is calculated.

The prototype was tested several times by scheduling PHASE II of the I-465 project just before it started on June 26, 1999. The linear scheduled was obtained by representing the existing schedule in the linear scheduling format shown in Figure 5.1. Once the scheduled was introduced the controlling path was calculated and shown to the contractors. However, at this point some problems with the prototype were encountered impairing researchers to continue working in the project. These issues were considered of extreme importance and no further experiments were deemed appropriate until the "bugs" were identified and solved.

From examining multiple times the code and logic of the software application, the following issues were identified:

When there are non-working periods in the schedule that were not initially specified in the layout of the linear schedule, the prototype fails to calculate the controlling path.

In certain cases the activity sequence list of the project is different that the assumed total least time group of activities. Hence, a new algorithm has to be developed in order to include those cases were our initial assumption does not hold.

V.2 Walsh Construction – US 231 (South River Road project in West Lafayette)

Walsh Construction was approached due to their involvement in the construction of the US 231 (South River Road project) in West Lafayette. This project, although small, offered the possibility of a quicker updating and monitoring.

Several meetings were held with the project manager, out of which the schedule shown in Figure 5.2 was done.

From this project, researchers learned:

The schedule was done every six weeks and revised every two.

The prototype needs an easier way to create plan views of the project.

Uncertainty for activity durations has to be incorporated in the schedule.

VI. Conclusions and Recommendations

VI.1 Conclusions

All of the objectives proposed by this study were achieved. Through the development of PULSS v1.0, INDOT and highway contractors have a linear scheduling software that allows to visually plan projects, calculate the controlling activities and present standard reports in Excel format. PULSS also proves that the computerization of the LSM is feasible, eliminating one of the more frequent arguments against the LS. We are confident that this step will allow the further refinement of the software tool taking it one step further in its evolution towards a mainstream application used by all highway contractors and DOT's..

In addition, and based on the feedback received from contractors that were contacted to use PULSS, we learned the following:

Linear schedules are easer to understand, review and change than CPM schedules. To scientifically prove this, further usability and human-computer interaction studies could be performed.

There is still some work to do to incorporate in LS a feature that will allow planners to avoid physical conflicts when working in a multiple lane highway. This experience was particularly important for Berns Construction when repaving the I-465.

The best environment for development of the tool might not be ACAD as initially assumed, since the use of this program is required and costly for contractors. In addition, the lower the learning curve and acquisition costs the faster the adoption.

INDOT Inspectors were not familiar with Linear Schedules, much less with the LSM technique.

VI.2 Further Research

Further research is recommended on the following areas that were indicated by contractors as needing improvement:

Visualization. Issues with linear scheduling and scheduling work that is performed in the same stations in different lanes. This would allow contractors to improve space, equipment and time utilization.

Human-computers interaction. Several of the interfaces of PULSS can be improved to be more user friendly. Other area needing improvement is the on-screen presentation of the linear schedules generated. These representations are sometimes difficult to see completely. The larger the project the more difficult the visualization in the screen.

It is recommended to continue the development of the application since results and feedback from contractors is encouraging. However, full collaboration from Inspectors is required, since they will also be important users of the product and consumers of the information generated with it.

Uncertainty and delay risk estimation. The LSM only deals with deterministic scenarios and does not allow the planner to perform statistical analysis on the schedule. This would be equivalent to have PERT for CPM. As LSM evolves and its features are improved, the next logical step is the inclusion of such risk/statistical analysis capabilities. Contractors find this feature particularly useful for performing "what-if" analysis and accurately estimate the project duration.

Information sharing with other INDOT software should be explored in depth since most of the information used to build LS is also used for billing and quality assurance.

INDOT should conduct workshop for its contractors and inspectors were linear schedules are discussed, as well as, their differences and benefits over CPM.

APPENDIX A

FIGURES

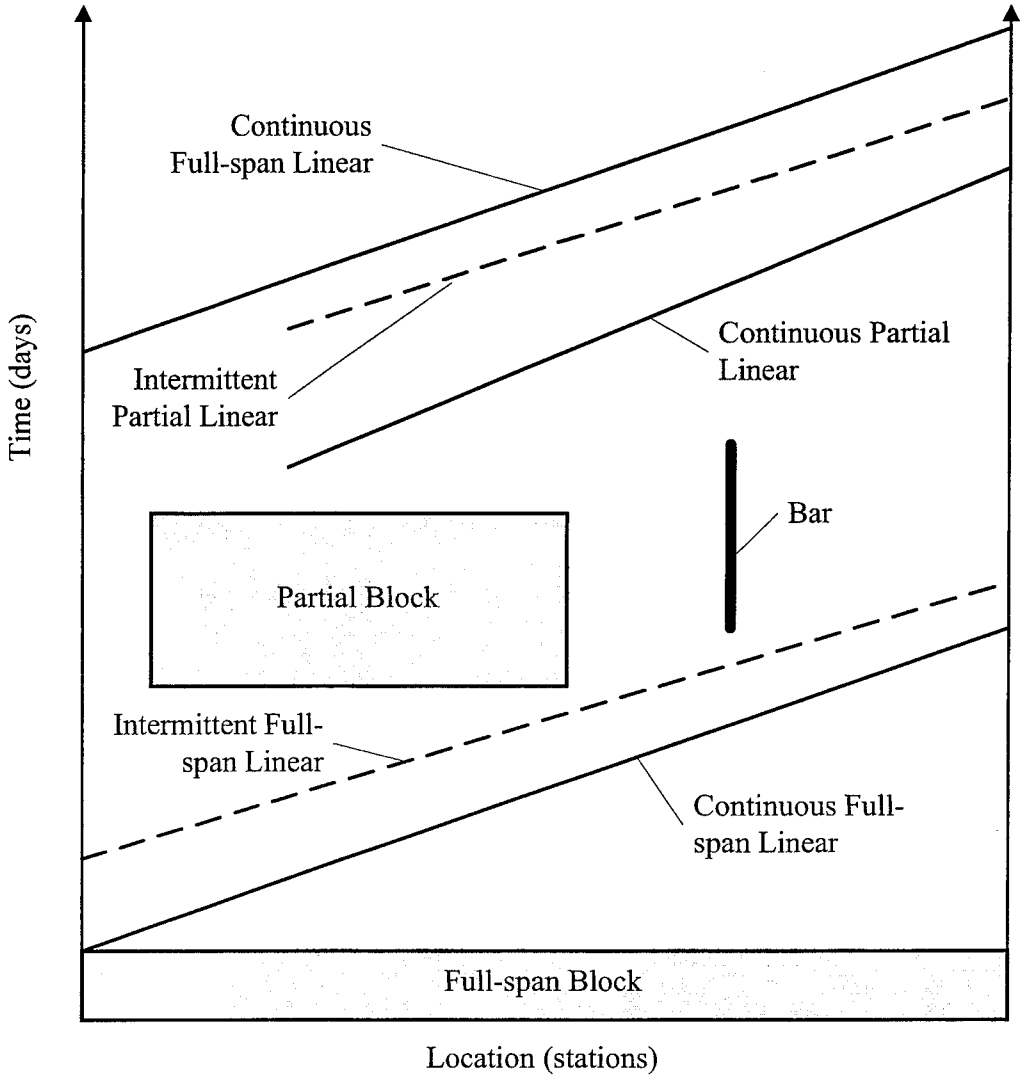


Figure 3.1 – Activity Types of Linear Schedules

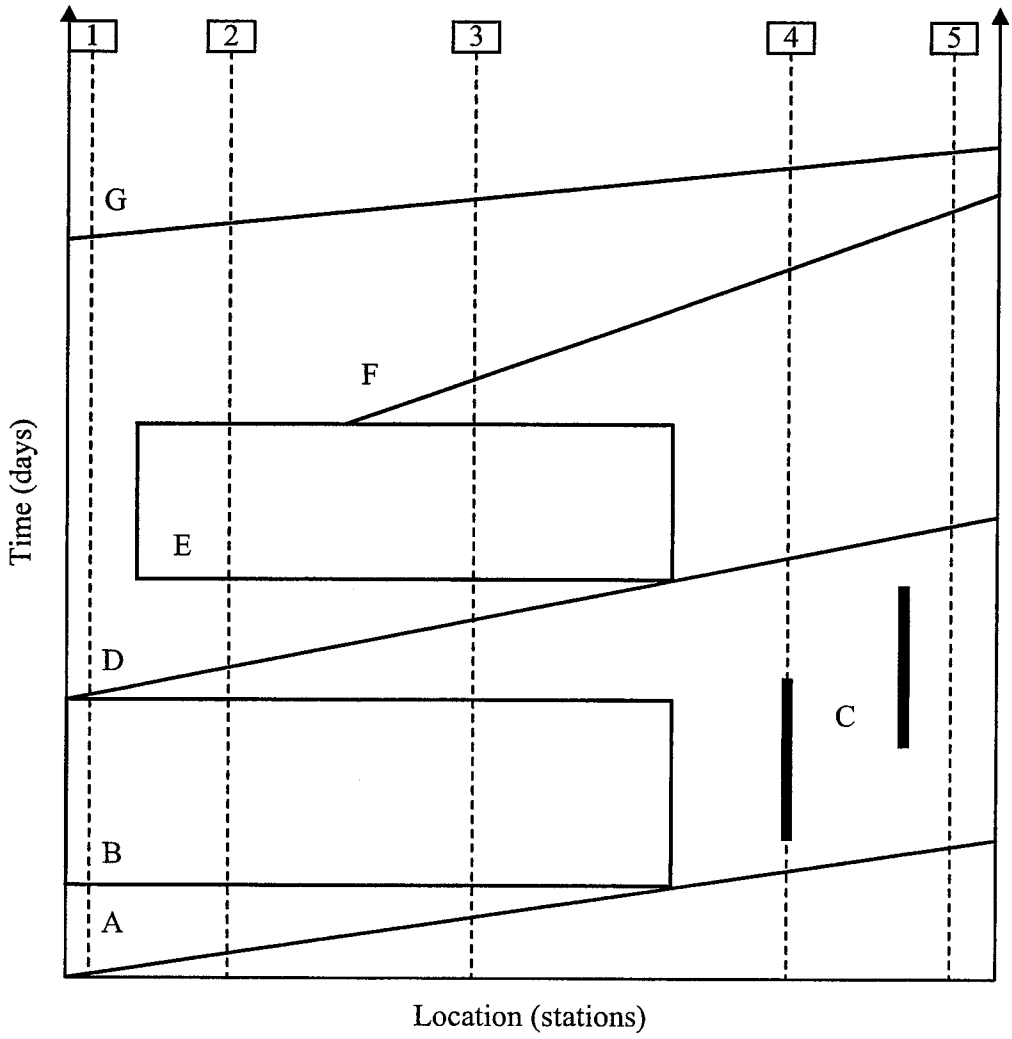
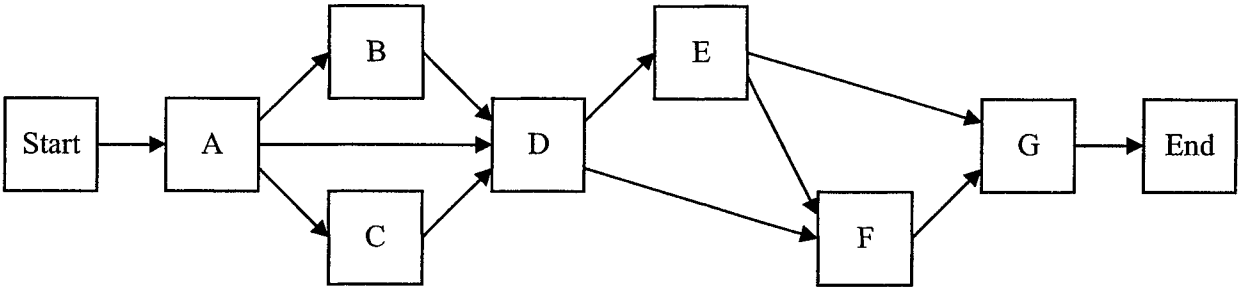


Figure 3.2 – Linear Schedule



Sequence 1 - Start-A-B-D-G-End
Sequence 2 - Start-A-B-D-E-G-End
Sequence 3 - Start-A-B-D-E-F-G-End
Sequence 4 - Start-A-C-D-F-G
Sequence 5 - Start-A-D-F-G

Figure 3.3 – Possible Activity Sequences

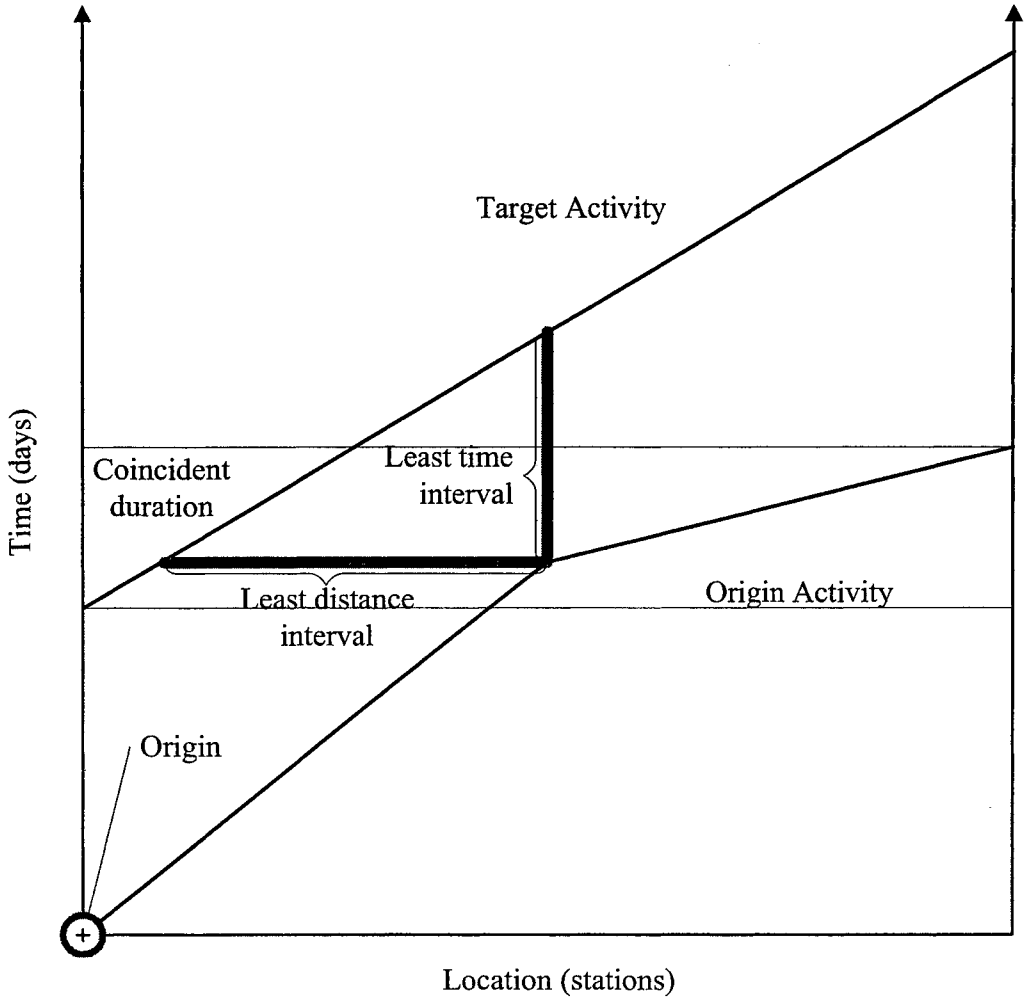


Figure 3.4 – Least Time and Least Distance

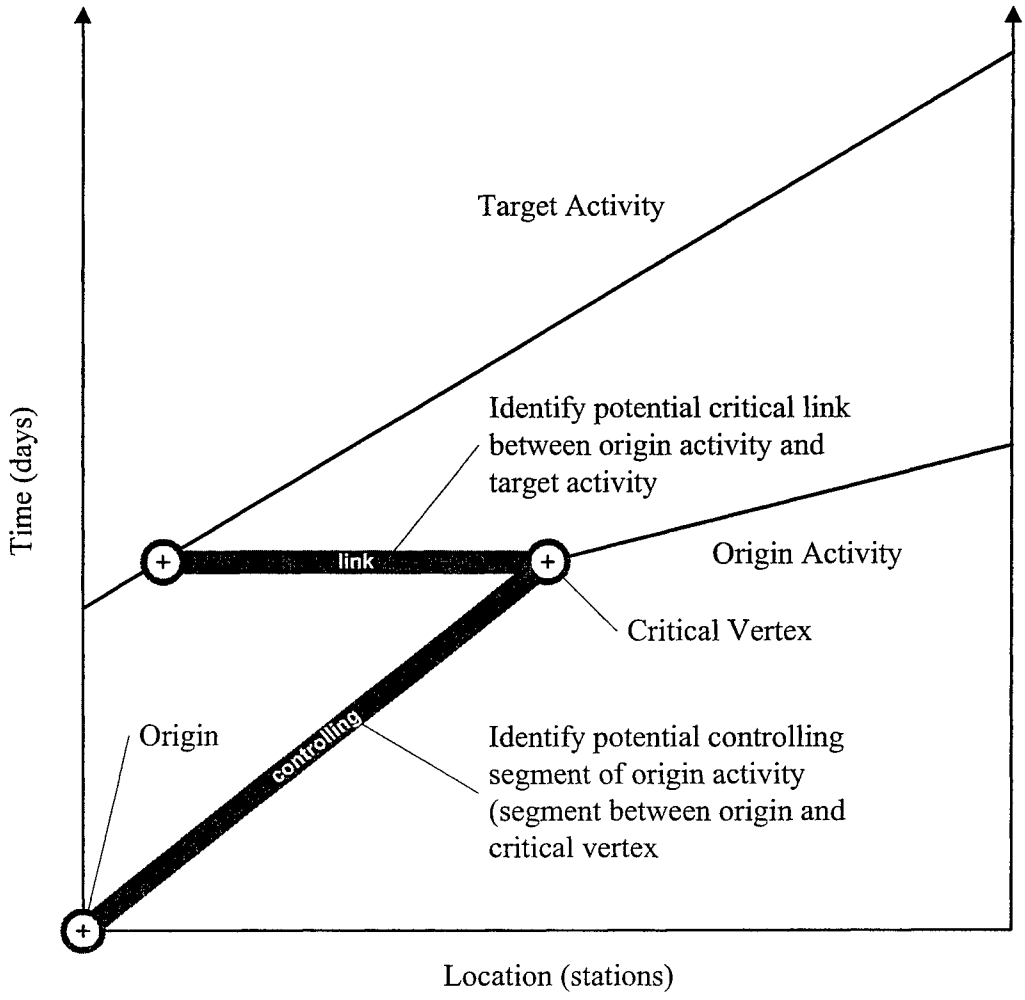


Figure 3.5 – Potential Controlling Segments

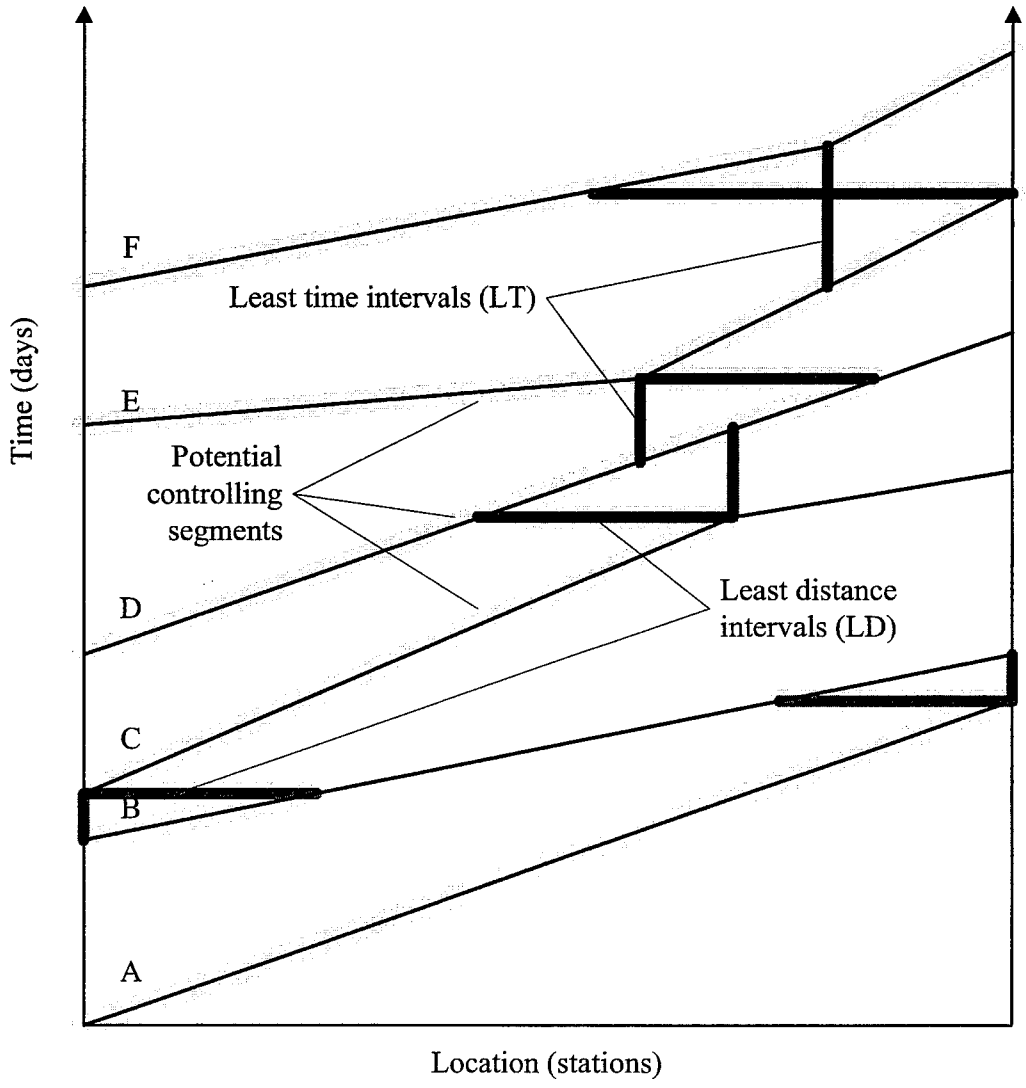


Figure 3.6 – Upward Pass

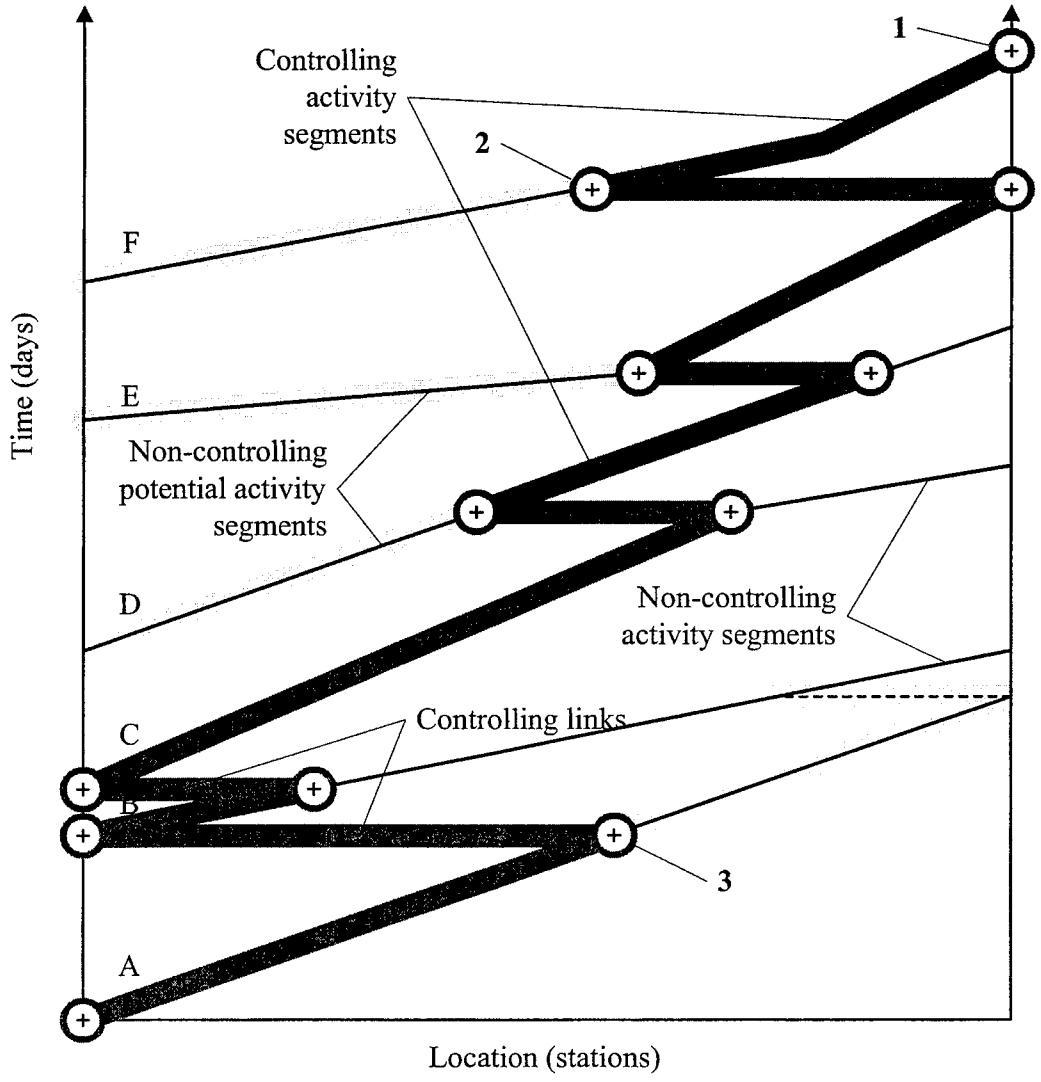


Figure 3.7- Downward Pass and Controlling Activity Path

Figure 4.1 – Linear Schedule Layout Screen

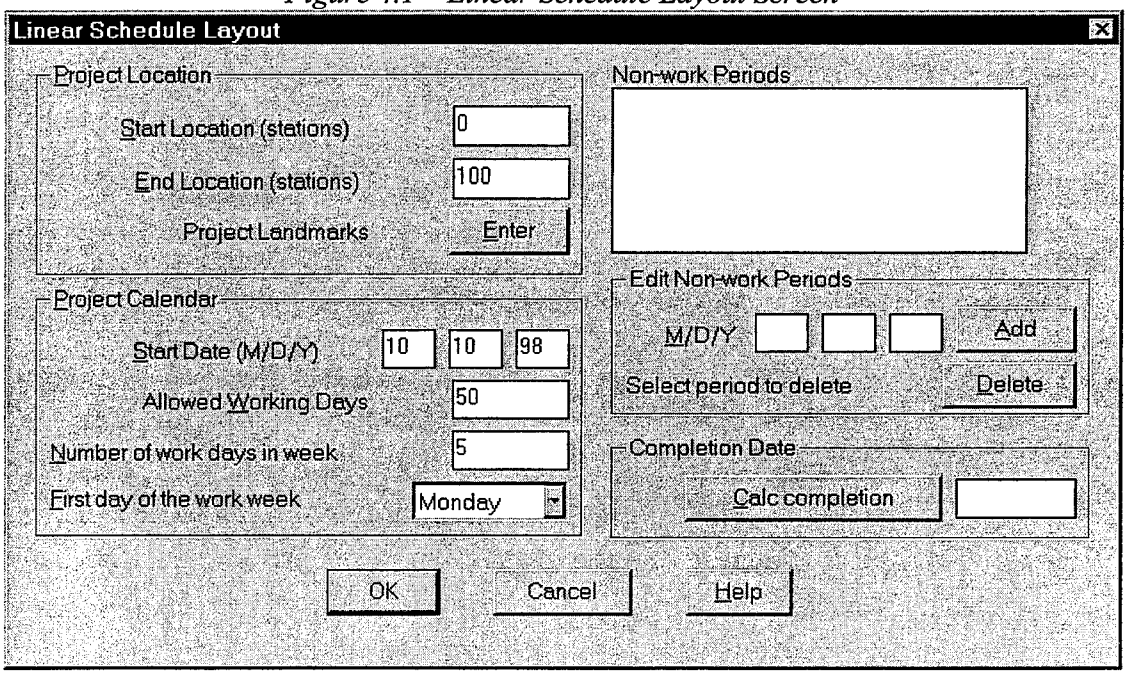


Figure 4.2 – Linear Schedule Layout Frame

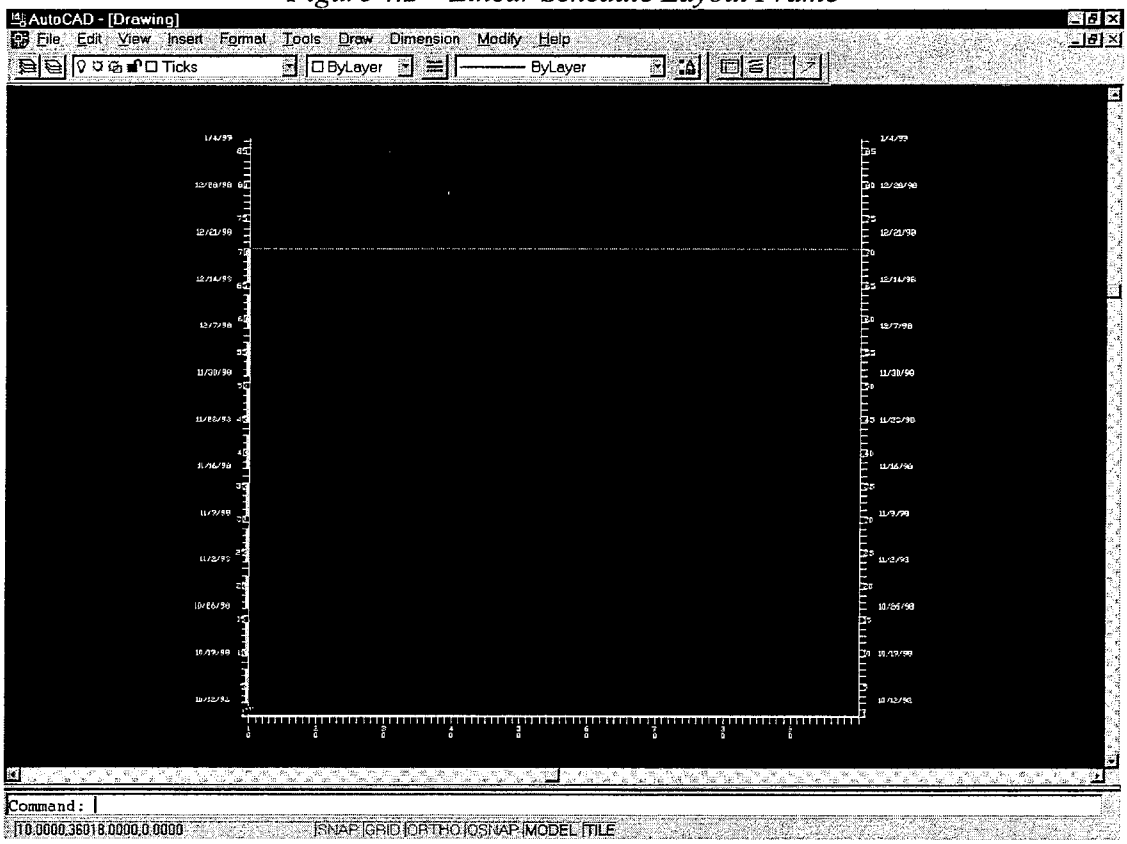


Figure 4.3 – Create Activity Dialog Box

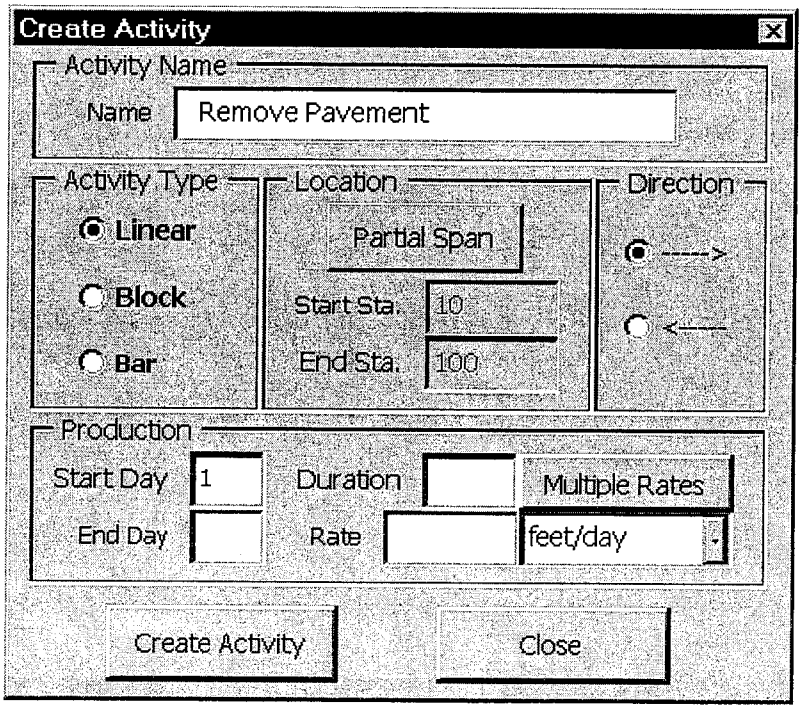


Figure 4.4 – Multiple Production Rates dialog box

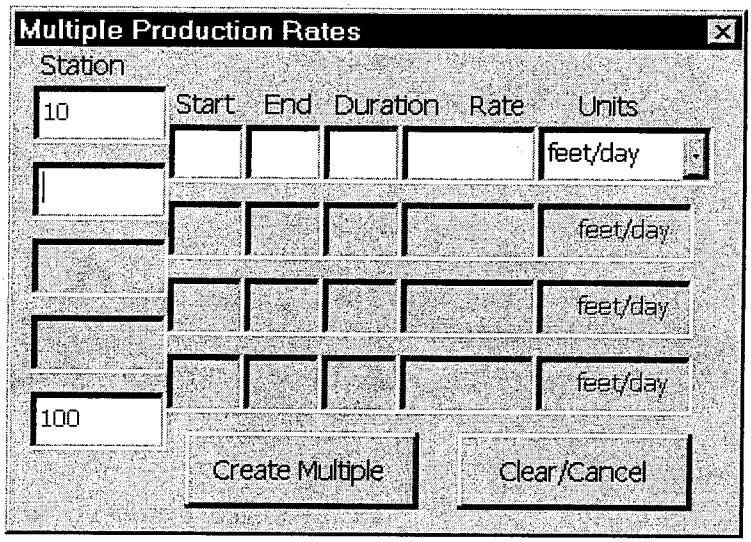


Figure 4.5 – ASL Information dialog box

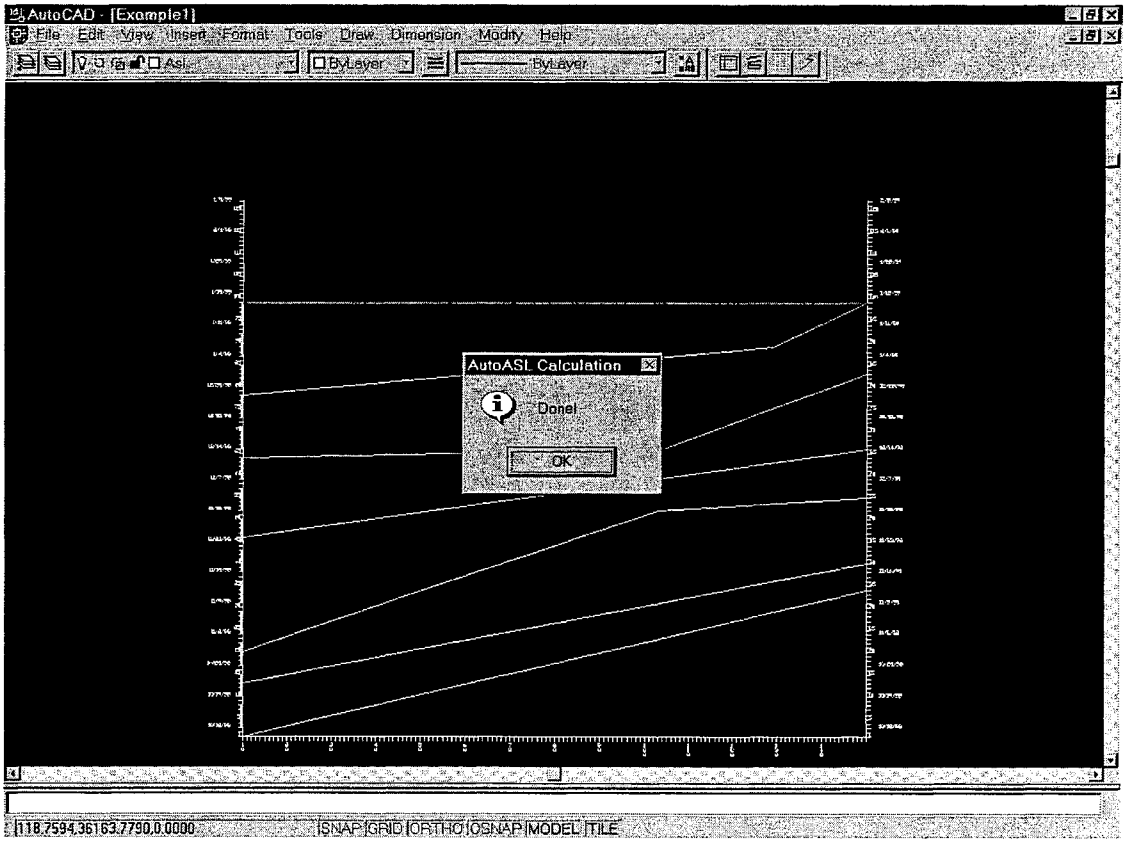
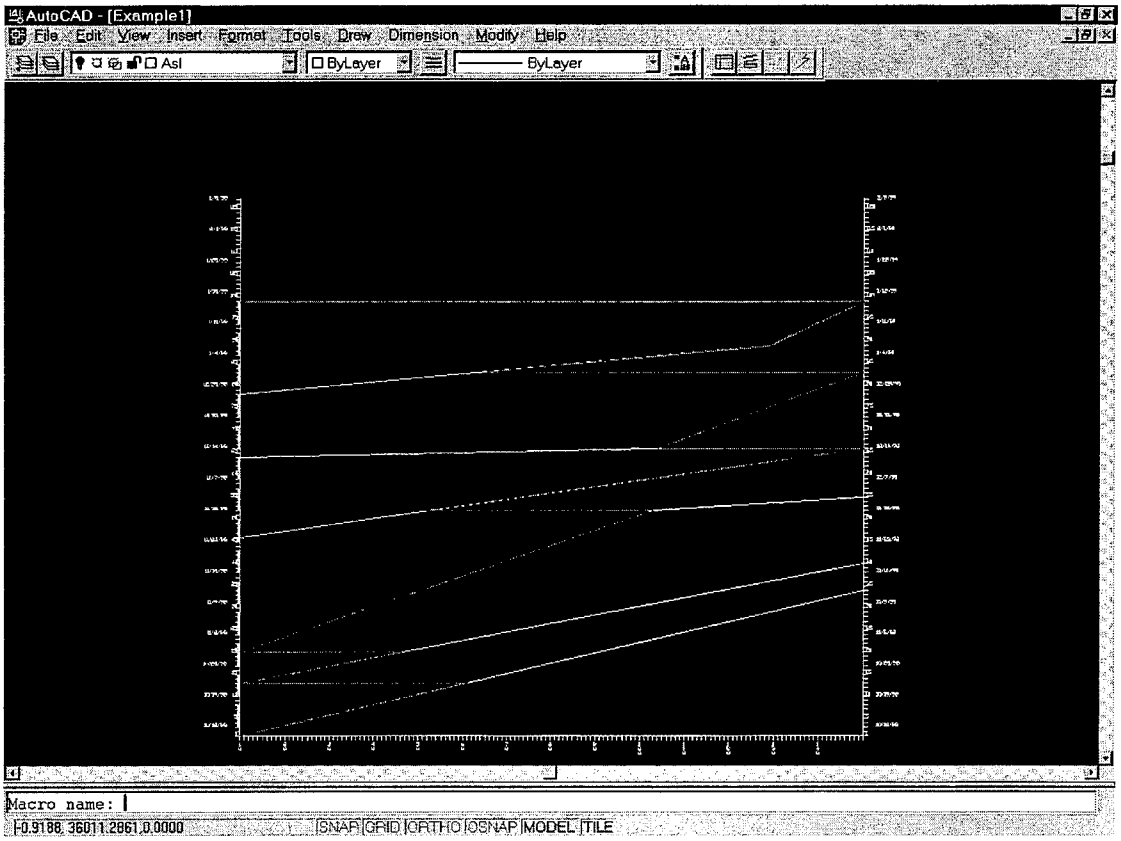


Figure 4.6 – Controlling Activity Path



APPENDIX B

Flow Diagrams and PULSS code

Figure B.1 – FINDLD Flow Diagram

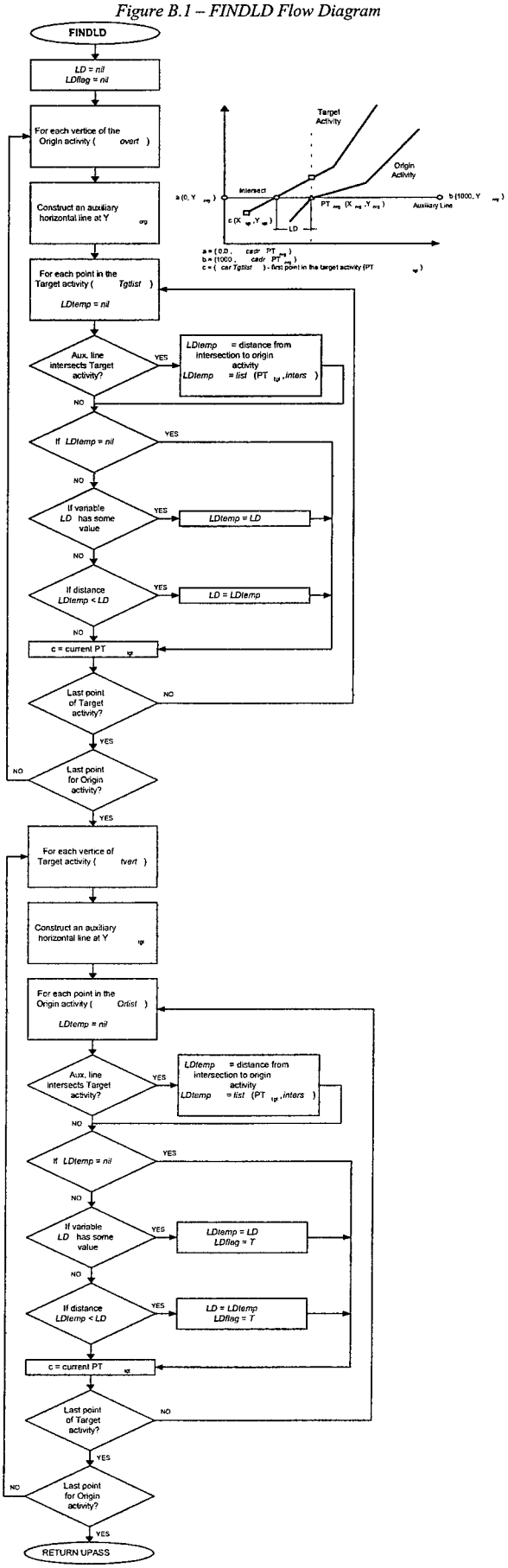


Figure B.2 – UPASS Flow Diagram

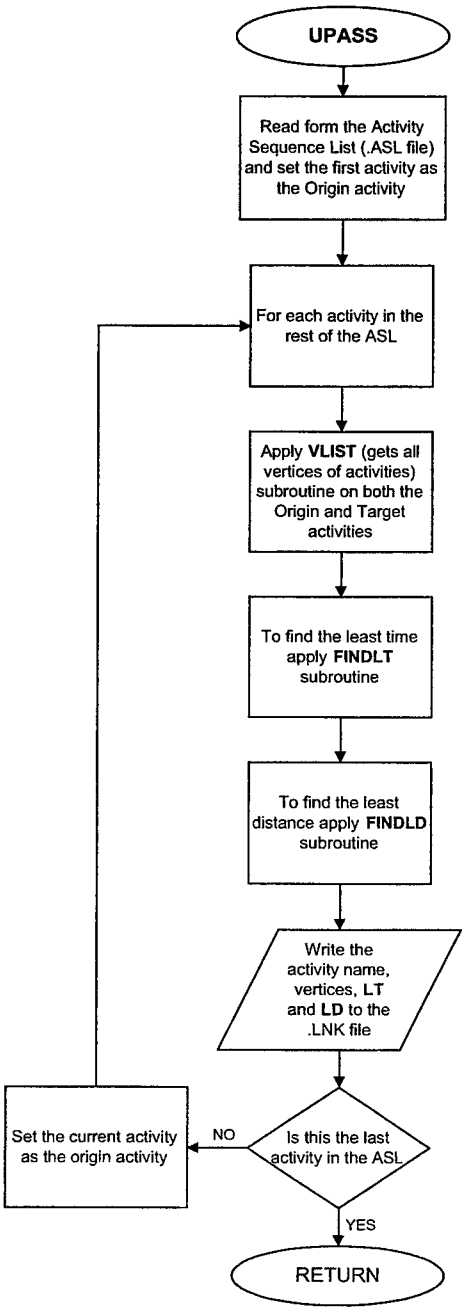
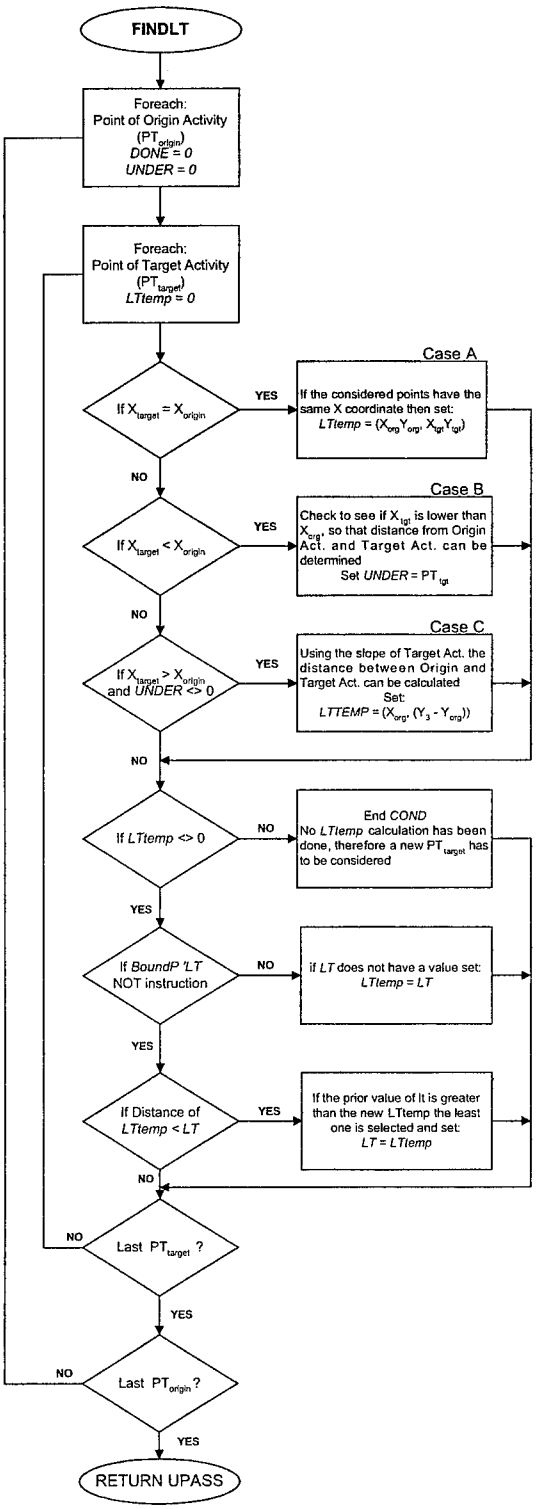
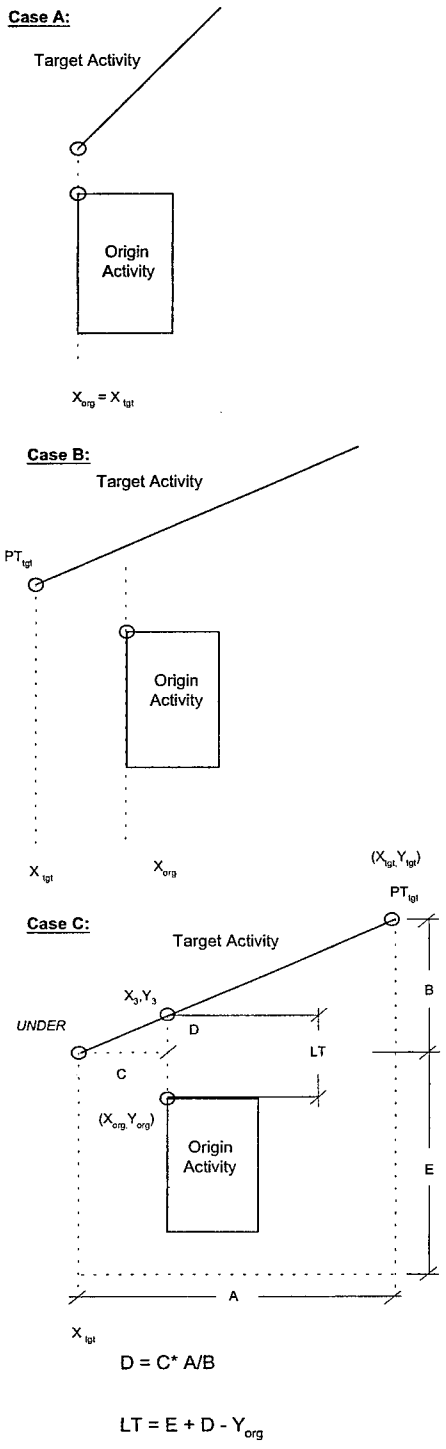


Figure B.3 - FINDLT Flow Diagram





```
;*****************************************************************
;****  This routine determines the path and name of the       ****
;****  drawing that is being used.                            ****
;****                                                         ****
;****                        DRWPATH                          ****
;*****************************************************************

      (setq acadObject (vlax-get-acad-object))
      (setq acadDocument (vla-get-ActiveDocument acadObject))
      (setq fullpath (vla-get-Path acadDocument))
      (setq path (substr fullpath 1 (- (strlen fullpath) 4)))

;*****************************************************************
;****  This routine takes the ename of a poly line and        ****
;****  returns a list of the vertices.                        ****
;****                                                         ****
;****                        VLIST                            ****
;*****************************************************************

;  **** this returns a list of vertices in a POLYLINE

(defun vlist (poly / ed en templist)
   (setq en (entnext (dxf -1 (entget (ssname poly 0)))))
   (setq ed (entget en))
   (setq templist (list (dxf 10 ed)))

   (while (and (setq en (entnext en)) (setq ed (entget en))
```

```
      (/= "SEQEND" (dxf 0 ed)))
      (setq templist (append templist (list (dxf 10 ed))))

   ) ;while
);vlist


;*****************************************************************
;****  This routine takes the vertice list of a the origin    ****
;****  and the target activities and returns the LT.          ****
;****                                                         ****
;****                       FINDLT                            ****
;*****************************************************************

(defun findlt (act1 act2 / pt1 pt2)
(setq lt nil
      act1rl nil
      act2rl nil)

(if (/= (car (car act1)) (car (last act1)));its a line
      (progn
       (if (< (cadr (car act1)) (cadr (last act1)))
             (setq act1rl 1)
       );if
      );progn
);if

(if (/= (car (car act2)) (car (last act2)));its a line
```

```
      (progn
         (if (< (cadr (car act2)) (cadr (last act2)))
             (setq act2rl 1)
       );if
      );progn
);if


(if (= (or (/= act1rl 1) (/= act2rl 1)) T)

;**** this nested foreach finds the LT from act1 to act2
     (progn
      (foreach pt1 act1
         (setq done nil)
         (setq under nil)

         (foreach pt2 act2

            (setq lttemp nil)
            (cond
               ((= done T))
               ((= (car pt2) (car pt1)) (setq lttemp (list pt1 pt2))
(setq done T))
               ((< (car pt2) (car pt1)) (setq under pt2))
               ((and (> (car pt2) (car pt1)) (/= under nil))
                  (setq lttemp (list pt1 (list (car pt1) (+ (* (- (car
pt1) (car under))
                     (/ (- (cadr pt2) (cadr under)) (- (car pt2) (car
under))))
                     (cadr under)) 0.0)))
```

```
                (setq done T))
            );cond

            (cond
               ((= lttemp nil))
               ((not (boundp 'lt)) (setq lt lttemp))
               ((< (distance (car lttemp) (cadr lttemp)) (distance (car
lt) (cadr lt)))
                  (setq lt lttemp))
            );cond

         );foreach)

      );foreach

      ;**** this nested foreach finds the LT from act2 to act1

      (foreach pt1 act2
         (setq done nil)
         (setq under nil)

         (foreach pt2 act1

            (setq lttemp nil)
            (cond
               ((= done T))
               ((= (car pt2) (car pt1)) (setq lttemp (list pt1 pt2))
(setq done T))
               ((< (car pt2) (car pt1)) (setq under pt2))
```

```
                ((and (> (car pt2) (car pt1)) (/= under nil))
                   (setq lttemp (list pt1 (list (car pt1) (+ (* (- (car
pt1) (car under))
                      (/ (- (cadr pt2) (cadr under)) (- (car pt2) (car
under))))
                      (cadr under)) 0.0)))
                   (setq done T))
             );cond

             (cond
                ((= lttemp nil))
                ((not (boundp 'lt)) (setq lt lttemp))
                ((< (distance (car lttemp) (cadr lttemp)) (distance (car
lt) (cadr lt)))
                   (setq lt lttemp))
             );cond

          );foreach)

       );foreach
    ); progn

;
;******* If both activities are right to left  *********
;

(progn
    (foreach pt1 act1
      (setq done nil)
      (setq under nil)
```

```
        (foreach pt2 act2

            (setq lttemp nil)
            (cond
               ((= done T))
               ((= (car pt2) (car pt1)) (setq lttemp (list pt1 pt2))
(setq done T))
               ((> (car pt2) (car pt1)) (setq under pt2))
               ((and (< (car pt2) (car pt1)) (/= under nil))
                  (setq lttemp (list pt1 (list (car pt1) (+ (* (- (car
pt1) (car under))
                     (/ (- (cadr pt2) (cadr under)) (- (car pt2) (car
under))))
                     (cadr under)) 0.0)))
                  (setq done T))
            );cond

            (cond
               ((= lttemp nil))
               ((not (boundp 'lt)) (setq lt lttemp))
               ((< (distance (car lttemp) (cadr lttemp)) (distance (car
lt) (cadr lt)))
                  (setq lt lttemp))
            );cond

         );foreach)

      );foreach
```

```
;**** this nested foreach finds the LT from act2 to act1


      (foreach pt1 act2

         (setq done nil)

         (setq under nil)


         (foreach pt2 act1


            (setq lttemp nil)

            (cond

               ((= done T))

               ((= (car pt2) (car pt1)) (setq lttemp (list pt1 pt2))
(setq done T))

               ((> (car pt2) (car pt1)) (setq under pt2))

               ((and (< (car pt2) (car pt1)) (/= under nil))

                  (setq lttemp (list pt1 (list (car pt1) (+ (* (- (car
pt1) (car under))

                     (/ (- (cadr pt2) (cadr under)) (- (car pt2) (car
under))))

                     (cadr under)) 0.0)))

                  (setq done T))

            );cond


            (cond

               ((= lttemp nil))

               ((not (boundp 'lt)) (setq lt lttemp))

               ((< (distance (car lttemp) (cadr lttemp)) (distance (car
lt) (cadr lt)))

                  (setq lt lttemp))

            );cond
```

```
        );foreach)

      );foreach
    );progn
  ); end if
);defun findlt


;*********************************************************************
;****  This routine takes the vertice list of a POLYLINE and     ****
;****  returns a list of vertices that are within the limits     ****
;****  of the LT and the coincident duration of the origin       ****
;****  and target                                                ****
;****                          VERTS                             ****
;*********************************************************************

(defun verts (act / pt1 pt2 pt3 pt4 dmin dmax templist templist2)

   (foreach pt1 act
      (if (and (or (<= (cadr pt1) (cadr (car lt)))
                   (<= (cadr pt1) (cadr (cadr lt))))
               (or (>= (cadr pt1) (cadr (car lt)))
                   (>= (cadr pt1) (cadr (cadr lt)))))
         (setq templist (append templist (list pt1)))

      );if
```

```
    );foreach

    (setq dmin (cadr (car tglist))
          dmax (cadr (car orlist)))

    (foreach pt2 tglist
       (if (< (cadr pt2) dmin) (setq dmin (cadr pt2)))); if, foreach
    (foreach pt3 orlist
       (if (> (cadr pt3) dmax) (setq dmax (cadr pt3)))); if, foreach

    (foreach pt4 templist
       (if (and (>= (cadr pt4) dmin)
                (<= (cadr pt4) dmax))
          (setq templist2 (append templist2 (list pt4)))

       );if

    );foreach

    ;(princ templist2)

);defun verts

;*****************************************************************
;****  This routine takes each point in the list of possible  ****
;****  LINK vertices that happen before in time that the LD   ****
;****  calculated in FINDLD.  A list with all the LINKS       ****
```

```
;****  between pairs of activities is returned.                ****
;****                                                          ****
;****                        FINDLINKS                         ****
;*************************************************************

(defun findlinks (orlist tglist / linktemp)
  (setq maxy (cadr(car ld)))
  (setq link nil)



  (foreach pt1 orlist
      (setq a (list 0.0 (cadr pt1) 0.0)
            b (list 1000.0 (cadr pt1) 0.0))
      (setq c (car tglist))


      (foreach pt2 tglist
         (setq linktemp nil)
         (if (inters a b c pt2) (setq linktemp (list (inters a b c pt2)
pt1)))
         (cond
            ((= linktemp nil))
          ((>= (cadr(car linktemp)) maxy))
            ((not (boundp 'link)) (setq link (append link (list
linktemp))))
         );cond
         (setq c pt2)


      );foreach
   );foreach
```

```
   (foreach pt1 tglist
      (setq a (list 0.0 (cadr pt1) 0.0)
            b (list 1000.0 (cadr pt1) 0.0))
      (setq c (car orlist))

      (foreach pt2 orlist
         (setq linktemp nil)
         (if (inters a b c pt2) (setq linktemp (list pt1 (inters a b c
pt2))))
         (cond
            ((= linktemp nil))
            ((>= (cadr(car linktemp)) maxy))
            ((not (boundp 'link)) (setq link (append link (list
linktemp))) )

         );cond
         (setq c pt2)

      );foreach
   );foreach

); defun findlinks

;*****************************************************************
;****  This routine takes each point in the list of possible  ****
;****  LD vertices and finds the point of intersection with   ****
;****  the appropriate activity.  The shortest LD is          ****
```

```
;****  returned.                                                    ****
;****                          FINDLD                               ****
;*****************************************************************

(defun findld (orlist tglist / ldtemp)
   (setq ld nil
         ldflag nil)
   (foreach pt1 orlist
      (setq a (list 0.0 (cadr pt1) 0.0)
            b (list 1000.0 (cadr pt1) 0.0))
      (setq c (car tglist))

      (foreach pt2 tglist
         (setq ldtemp nil)
         (if (inters a b c pt2) (setq ldtemp (list (inters a b c pt2)
pt1)))
         (cond
            ((= ldtemp nil))
            ((not (boundp 'ld)) (setq ld ldtemp))
            ((< (distance (car ldtemp) (cadr ldtemp)) (distance (car
ld) (cadr ld)))
               (setq ld ldtemp))
         );cond
         (setq c pt2)

      );foreach
   );foreach

   (foreach pt1 tglist
```

```
      (setq a (list 0.0 (cadr pt1) 0.0)

            b (list 1000.0 (cadr pt1) 0.0)) ; this will have to change
- give the end station instead

      (setq c (car orlist))


      (foreach pt2 orlist

         (setq ldtemp nil)

         (if (inters a b c pt2) (setq ldtemp (list pt1 (inters a b c
pt2))))

         (cond

            ((= ldtemp nil))

            ((not (boundp 'ld)) (setq ld ldtemp) (setq ldflag T))

            ((< (distance (car ldtemp) (cadr ldtemp)) (distance (car
ld) (cadr ld)))

               (setq ld ldtemp) (setq ldflag T))

         );cond

         (setq c pt2)


      );foreach

   );foreach


      (findlinks orlist tglist)


      (if (= (= link nil) T)

            (setq allld (list ld))

            (setq allld (list ld)

                  allld (append allld link))

      ); end if
```

```
);defun findld

;*********************************************************************
;****  This routing reads from the file *.tnk all the possible ****
;****  horizontal links between any two activities. It is      ****
;****  called only when the Least Time ASL is NOT the          ****
;****  controlling ASL                                         ****
;****                                                          ****
;****                    FINDOTHERASL                          ****
;*********************************************************************

(defun findotherasl (name / temp)

 (setq otherasl (open (strcat path ".TNK") "r"))
  (while
      (setq temp (read-line otherasl))
      (setq altlnk (append altlnk (list temp)))
   );while
 (close otherasl)

  (setq altlnklnum (length altlnk))
  (setq num 0
      numberlnk 0)
(while
  (= T (and (< num altlnknum) (/= altlnk nil)))

  ((setq num (+ num 1))
```

```
  (while (= T (and (/= (car altlnk) "other link") (/= (car altlnk)
nil)))

       (setq temp (car altlnk)

             altlnk (cdr altlnk)

             altlnk (append altlnk (list altlnk))

             flag 1)

  ); loop do to end of file

  )

  ); end while

); defun findotherasl



;*****************************************************************

;****  This routine automatically builds the activity          ****

;****  sequence list in an external file (drawing_name.ASL).   ****

;****  .ASL stands for activity sequence list.                 ****

;****  It uses the file drawing_name.tsl  created by a VBA     ****

;****  macro called auto-asl                                   ****

;****                                                          ****

;****                    MULTIPASS                             ****

;*****************************************************************


(defun multipass (/ temp)

   (setq asl1 nil

       asl 0

       temp nil

       temp1 nil

       lt1 nil)
```

```
  (setq aslfile (open (strcat path ".TSL") "r"))

  (while
      (setq temp (read-line aslfile))
      (setq asl1 (append asl1 (list temp)))
   );while
   (close aslfile)

  (setq aslnum (length asl1))
  (setq num 0
      numberasl 0)
(while
  (= T (and (< num aslnum) (/= asl nil)))
  (setq num (+ num 1)
        leasttime 0
      asl nil
      flag =0)

  (while (= T (and (/= (car asl1) "other asl") (/= (car asl1) nil)))
       (setq temp1 (car asl1)
             asl1 (cdr asl1)
             asl (append asl (list temp1))
             flag 1)
  ); loop do to end of file

  (while (= flag 1)
```

```
  (setq asl1 (cdr asl1)

          numberasl (+ numberasl 1))

   (setq listtemp asl)

   (setq org (car asl))

   (setq asl (cdr asl))

   (setq activ (car asl))

   (setq orlist (vlist (ssget "X" (list (cons 8 org) '(0 .
"POLYLINE")))))

   (setq tglist (vlist (ssget "X" (list (cons 8 activ) '(0 .
"POLYLINE")))))

   (findlt orlist tglist)

   (setq leasttime (distance (car lt)(cadr lt)))

   (setq org activ)

   (setq asl (cdr asl))


   (foreach activ asl

      (setq orlist (vlist (ssget "X" (list (cons 8 org) '(0 .
"POLYLINE")))))

      (setq tglist (vlist (ssget "X" (list (cons 8 activ) '(0 .
"POLYLINE")))))

      (findlt orlist tglist)

      (setq leasttime (+ leasttime (distance (cadr lt)(car lt))))

      (setq org activ)

   );foreach


    (setq flag 0)


    ;

    ; when two different ASLs have the same LT something must be done
here

    ;
```

```
        (if (/= leasttime nil)
             (if (/= lt1 nil)
              (if (< leasttime lt1)
                 (setq deftemp listtemp
                        lt1 leasttime)
              ); end if
               (progn
                 (setq lt1 leasttime
                       deftemp listtemp)
               );progn
             ); end if
        ); end if



   );loop do to flag = 0


   ); loop


    (setq asllnkfile (open (strcat path ".ASL") "W"))
    (foreach activ deftemp
       (write-line (strcat activ) asllnkfile)
    ); foreach
       (close asllnkfile)

);defun multipass
```

```
;*****************************************************************
;****  This routine performs the upward pass using the .asl   ****
;****  file created with make_asl. It creates the .lnk file.  ****
;****  The .lnk file contains activity name and points and    ****
;****  LD and LT for each pair of activities.                 ****
;****                          UPASS                              ****
;*****************************************************************

(defun upass (/ temp)
         (setq asl nil)

   (setq aslfile (open (strcat path ".ASL") "r"))
   (while
      (setq temp (read-line aslfile))
      (setq asl (append asl (list temp)))
   );while
   (close aslfile)
   (setq aslnum (length asl))

   (setq org (car asl))
   (setq asl (cdr asl))
   (setq activ (car asl));test line
   (setq orlist (vlist (ssget "X" (list (cons 8 org) '(0 .
"POLYLINE")))))
   (setq tglist (vlist (ssget "X" (list (cons 8 activ) '(0 .
"POLYLINE")))))
   (findlt orlist tglist)
   (findld orlist tglist)
```

```
    (setq lnkfile (open (strcat path ".LNK") "w"))
    (write-line (strcat "(" "\"" org "\"" (etos orlist)")") lnkfile)
    (write-line (strcat "(" (etos allld) (etos lt) ")" ) lnkfile)
    (write-line (strcat "(" "\"" activ "\"" (etos tglist)")") lnkfile)
    (close lnkfile)
    (setq org activ)
    (setq asl (cdr asl))
    (foreach activ asl
       (setq orlist (vlist (ssget "X" (list (cons 8 org) '(0 .
"POLYLINE")))))
       (setq tglist (vlist (ssget "X" (list (cons 8 activ) '(0 .
"POLYLINE")))))
       (findlt orlist tglist)
       (findld orlist tglist)
       (setq lnkfile (open (strcat path ".LNK") "a"))
       (write-line (strcat "(" (etos allld) (etos lt) ")" ) lnkfile)
       (write-line (strcat "(" "\"" activ "\"" (etos tglist)")")
lnkfile)
       (close lnkfile)
       (setq org activ)
    );foreach

);defun upass

;*****************************************************************
;****  This routine performs the downward pass using the .lnk ****
;****  file created with upass.                                ****
;****                                                          ****
;****                      DPASS                               ****
```

```
;*****************************************************************


(defun dpass (/ templist)
   (ctline)

  ; this is done only for the last activity in the schedule

   (setq temp1 (read (getline aslnum))
         aslnum (1- aslnum)
         name (car temp1)
         pts (cadr temp1)
         temp2 (read (getline aslnum))
         aslnum (1- aslnum)
         ld (car(car temp2))
        temp2 (cdr temp2)
         templist pts)

   (setq ctlfile (open (strcat path ".CTL") "w"))
   (write-line (strcat "(" "\"" name "\"" (etos templist)")") ctlfile)
   (write-line (strcat (etos ld)) ctlfile)
   (close ctlfile)
   (setq templist (cadr ld))

  ;** for activities between the last and the first activity

   (while
      (> aslnum 1)
```

```
(setq temp1 (read (getline aslnum))
   aslnum (1- aslnum)
   name (car temp1)
   pts (cadr temp1)
   temp2 (read (getline aslnum))
   aslnum (1- aslnum)
   allld (car temp2)
   ldnext (car (car temp2))
   otherld (cdr (car temp2))
   templist nil)

; the original control.lsp lacked distinguishing ldnexts. These
lines solve
; part of the problem

(while (= T (and (<= (cadr (car ld)) (cadr (car ldnext)))
            (/= otherld nil)))
      (setq ldnext (car otherld)
            otherld (cdr otherld))
); while

; here a problem arises when ldnext is above ld and the pt
considered
; is below both. To solve the problem we need to take other link
from the org
; activity to the tgt activity until that link is below ld

; If there is no other link between activities in the existing
; Asl calculated by Multipass, other asl must be calculated using
; links between activities.
```

```
     ; This process is dynamic and will use the *.tnk file

     ; in which links between any pair of activities with conicident
duration

     ; have been determined.


     ; FINLDL would be applied to activity "name" and its corresponding
activity

     ; pair of the link in *.tnk




     ;if

      (if (= T (and (<= (cadr (car ld)) (cadr (car ldnext)))

                 (= otherld nil)))

     ;then

           (findotherasl name)) ; this line directs to a subroutine
that tries to find

                              ; links from the activity "name" to other
activity

     ;end if




      (cond


       ; FOR ACTIVITIES THAT ARE LINES

         (((/= (car (car pts)) (car (last pts)));its a line

            (if (< (cadr (car pts)) (cadr (last pts)))

               (setq pts (reverse pts)));if

            (setq pcount 0
```

```
  flagpt 0)
(while (and
   (setq pt (nth pcount pts))
   (> (cadr pt) (cadr (car ldnext))));and
      (if (<= (cadr pt) (cadr (car ld)))
    (if (= flagpt 0)
    (setq templist (append templist (list (cadr ld)))
            templist (append templist (list pt))
            flagpt 1)
    (setq templist (append templist (list pt)))
      );if
  );if
   (setq pcount (1+ pcount))
);while

(if (= (cadr pt) (cadr (car ldnext)))
(if (= flagpt 0)
    (setq templist (append templist (list (cadr ld)))
            templist (append templist (list pt))
            flagpt 1)
    (setq templist (append templist (list pt)))
      )
    (setq templist (append templist (list (car ldnext))))
);if


(setq ctlfile (open (strcat path ".CTL") "a"))
(write-line (strcat "(" "\"" name "\"" (etos templist)")")
ctlfile)
```

```
            (write-line (strcat (etos ldnext)) ctlfile)
            (close ctlfile))
        ; End line activities

        ; FOR ACTIVITIES THAT ARE BLOCKS
          ((= (cadr (car pts)) (cadr (cadr pts)));its a block
            (foreach pnt pts
               (if (< (cadr pnt) (cadr (car ldnext)))
                  (setq pnt (list (car pnt) (cadr (car ldnext))))
                  (if (> (cadr pnt) (cadr (car ld)))
                     (setq pnt (list (car pnt) (cadr (car ld))))))));if,
if
               (setq templist (append templist (list pnt)))
            );foreach
            (setq templist (append templist (list (car templist))))
            (setq ctlfile (open (strcat path ".CTL") "a"))
            (write-line (strcat "(" "\"" name "\"" (etos templist)")")
ctlfile)
            (write-line (strcat (etos ldnext)) ctlfile)
            (close ctlfile))
         ; End block activities

      );cond
      (setq ld ldnext)
   );while

   ;** For the last activity in the schedule which is ALWAYS FIRSTDAY

   (setq temp1 (read (getline aslnum))
```

```
         aslnum (1- aslnum)
         name (car temp1)
         pts (cadr temp1)
         templist (list pt))
      (setq ctlfile (open (strcat path ".CTL") "a"))
      (write-line (strcat "(" "\"" name "\"" (etos templist)")")
ctlfile)
      (close ctlfile)

);defun dpass




;*****************************************************************
;****  This routine draws the controlling activity path on    ****
;****  the linear schedule in a layer named "CONTROLLING".    ****
;****                                                         ****
;****                          SHOW                           ****
;*****************************************************************

(defun show ()
;   (setq curlayer (getvar "CLAYER"))
;   (setvar "CLAYER" "CONTROLLING")

   (setq ctlfile (open (strcat path ".CTL") "r"))
   (while
      (setq temp1 (read-line ctlfile))
      (setq temp (read temp1))
      (if (= (type (car temp)) 'STR)
```

```
         (setq temp (cadr temp)));if
      (entmake '((0 . "POLYLINE") (40 . 0.05) (41 . 0.05) (62 . 1) (8 .
"CONTROLLING")))
      (foreach pt temp
         (entmake (list '(0 . "VERTEX") (setq v (list 10 (car pt) (cadr
pt)))))
      );foreach
      (entmake '((0 . "SEQEND")))
   );while
   (close ctlfile)

;   (setvar "CLAYER" curlayer)
);defun show




;*****************************************************************
;****  This routine counts the number of lines in the file    ****
;****  "dwgname".LNK. This is needed for the downward pass    ****
;****  which needs the last line first. Sets ASLNUM to the    ****
;****  number of lines in the file.                           ****
;****                        CTLINE                           ****
;*****************************************************************



(defun ctline ()
   (setq ctfile (open (strcat path ".LNK") "r"))
   (setq aslnum 0)
   (while
```

```
      (read-line ctfile)
      (setq aslnum (1+ aslnum))
   );while
   (close ctfile)
);defun getline


;*********************************************************************
;****  This routine takes a number and returns the line       ****
;****  corresponding to that number from the "dwgname".LNK    ****
;****  file.                                                  ****
;****                        GETLINE                          ****
;*********************************************************************

(defun getline (num / lineinfo count)
   (setq getfile (open (strcat path ".LNK") "r"))
   (setq count 0)
   (while
      (< count num)
      (setq lineinfo (read-line getfile))
      (setq count (1+ count))
   );while
   (close getfile)
   (princ lineinfo)
);defun getline
```

```
;*****************************************************************
;**** ETOS (Expression TO String) takes any expression and    ****
;**** converts to a string.  "STRings" are returned as double ****
;**** ""STRings"". The READ function can be used to return    ****
;**** the original value of any string returned by ETOS.      ****
;****                          ETOS                          ****
;*****************************************************************


(defun etos (arg / file)
  (if (= 'STR (type arg)) (setq arg (strcat "\"" arg "\"")))
  (setq file (open "$" "w"))
  (princ arg file)
  (close file)
  (setq file (open "$" "r"))
  (setq arg (read-line file))
  (close file)
  (close (open "$" "w"))
  arg
);defun ETOS


;* DXF takes an integer dxf code and an entity data list.
;* It returns the data element of the association pair.
;*
(defun dxf(code elist)
  (cdr (assoc code elist))   ;finds the association pair, strips 1st
element
```

```
);defun
;*

;;;CAP shortcut to run all routines at once and draw the CAP

(defun CAP (/ lin)
  (multipass)
  (upass)
  (dpass)
  (show))
```

APPENDIX C

Linear Schedules from Alfa-test projects

Figure 5.1 – PHASE II I-465 Project Linear Schedule

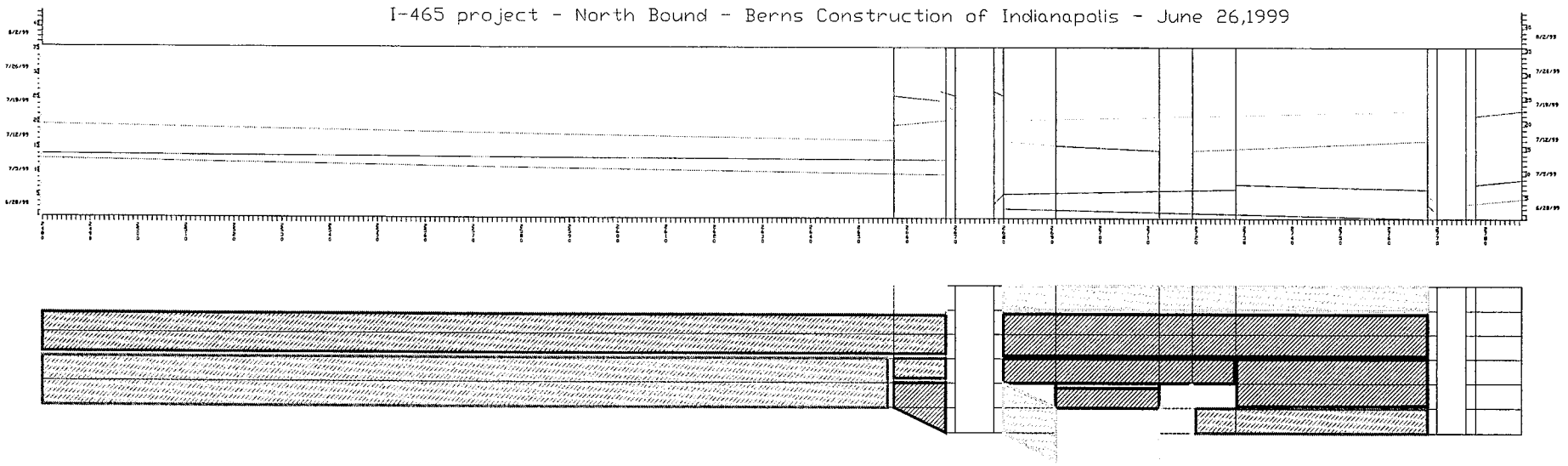


Figure 5.2 – US231 South River Road Projec