PRELIMINARY PLANNING OF FOREST ROADS USING ARC GRID

By

Shenglin Xu

In partial fulfillment of the requirements for the degree of

MASTER OF FORESTRY

A paper submitted to

Department of Forest Engineering

Oregon State University

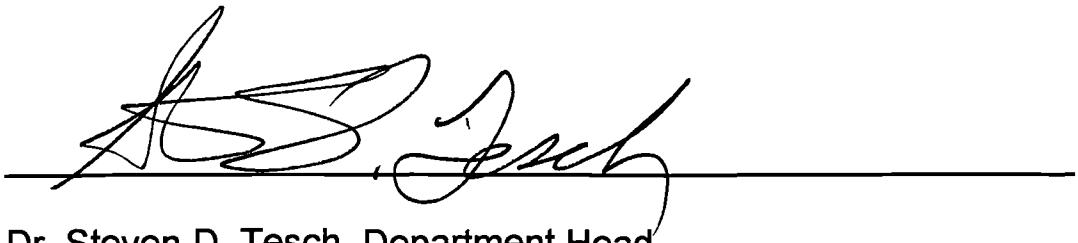
Corvallis, Oregon 97331

Completed February 16, 1996

Commencement March 1996

APPROVED:

Dr. John Sessions, Major professor
Forest Engineering



Dr. Steven D. Tesch, Department Head
Forest Engineering

Paper presented to graduate committee February 16, 1996

ACKNOWLEDGMENTS

I would like first to thank my major professor, Dr. John Sessions, for his guidance and great support for this project. I am also very grateful to the rest of my graduate committee, Dr. A. Jon Kimerling and Mr. Brian W. Kramer ,P.E. for their valuable suggestions and help.

I appreciate very much the assistance from the Forest Engineering Department, OSU and the financial support provided by Dr. Sessions for using the SUN workstation in the Forest Science Department.

My sincere thanks also goes to the following people:

Mr. George W. Lienkaemper, the GIS specialist in the GIS center at Forest Science Laboratory, who gave me a lot of help on GIS data collection and ARC/INFO operations.

Dr. Zhenyan Shen from Forest Engineering Incorporated, Corvallis, Oregon, who was very helpful in selecting the topic of the project.

My wife Peiqin Yu and son Di Xu, for their moral support.

ABSTRACT OF THE PAPER OF

Shenglin Xu for the degree of Master of Forestry in Forest Operations

presented February 16, 1996

Title: Preliminary Planning of Forest Roads Using ARC GRID

Abstract approved: Dr. John Sessions

Forest road planning is important in logging operations, because it can affect feasibility and cost of forest operations. Due to rapid advances in computer hardware, GIS software, and remote sensing techniques, computer forest road planning models using GIS are becoming practical. DEM data at 30 meter resolution are common and 10 meter resolution exists. Eventually 10 meter resolution will be common and and perhaps 5 meter or better resolution will be available.

The objective of this paper is to explore the usefulness of existing "off-the-shelf" GIS software in preliminary forest road planning. I chose ARC/INFO a widely used GIS in the United States by public and private forest managers. I tried two approaches to forest road location. In the first approach I used the available PATHDISTANCE and COSTPATH functions within ARC GRID. In the

second approach I developed a supplementary "C" program to calculate earthwork costs for each cell to improve the cost estimates in the shortest path calculation. The earthwork costs included considerations of excavation quantities and type of soil. I compared the results for the two approaches on a 1657 acre section of the H.J. Andrews Experimental Forest. I found the second approach that included calculation of the earthwork costs to give superior results.

The test results suggest that preliminary planning of roads using GIS coupled with supplementary programs may be useful for preliminary road planning, particularly as higher resolution data becomes available. Additional research needs to be done on refining horizontal and vertical geometry considerations within the GRID GIS structure in order to evaluate vehicle performance and to improve estimates of construction quantities.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF MAPS

LIST OF TABLES

1 INTRODUCTION

Forest road planning is important in logging operations, because it can effect feasibility and cost of forest operations. Due to rapid advances in computer hardware, GIS software, and remote sensing techniques, computer forest road planning models using GIS are becoming practical. DEM data at 30 meter resolution is common and 10 meter resolution exists. Eventually 10 meter resolution will be common and perhaps 5 meter or better resolution will be available. The objective of this paper is to explore the usefulness of existing "off-the-shelf" GIS software in preliminary forest road planning.

ARC/INFO is a widely used GIS in the United States by public and private forest managers. ARC/INFO can run both on UNIX and PC platforms. " ARC GRID is a raster - or cell - based geoprocessing toolbox that is integrated with ARC/INFO. Like all extensions of the ARC/INFO data model, ARC GRID uses a georelational model for geographic data. A grid is similar to an ARC/INFO coverage, data belonging to different themes are stored as separate grids. Because ARC GRID is integrated with ARC/INFO, many of the capabilities that are common between modules can be used. ARC GRID uses the graphic environments of ARCPLOT and ARCEDIT software for all visual display of grids. The ARC Macro Language (AML) can be used as a macro facility and to control the user environment. Metafiles and Plotter drivers that have been developed for the IMAGE INTEGRATOR are fully integrated with the ARC GRID software" (ESRI, 1995).

The two greatest advantages of a grid-cell-based structure for representing geographic space over other GIS structures are:

1. The ability to represent continuous surfaces.
2. The ability to store points, lines, polygons and surfaces uniformly.

A continuous surface is best represented by a cell-based system since the attributes of a location in the cell-based system are a function of the location relative to a phenomenon that is progressively changing as it moves across space.

Because all cells are uniform and cells can represent points, lines, polygons and surfaces, all four representations are identical. This uniformity becomes particularly important when you want to combine data types. In GRID, overlay operations are very fast. The main drawback to the uniformity is that some accuracy may be sacrificed when representing each data type as a cell.

"ARC GRID can be used in a multitude of applications, including land use planning, market research, disease control and famine relief. Because a cell-based structure divides the world into individual locations that it then describes, this technology best addresses location problems. These include problems where the location and its surroundings are as important , if not more important, than the feature makeup"(ESRI, 1995).

Forest road planning is a location problem. A planning area can be divided into individual grids. Each grid could be a section of road depending on it surroundings (slope, earthwork, etc.). DEMs (DTMs) are grid format data sets. Some research on forest road planning using DEMs has been done. For example: Burke (1974), Young and Lemkow (1976), Twito et al. (1982), Liu and Sessions(1993), and Jaeger and Becker (1995). As more GIS data become available now, forest road planning should use soils, streams, stands and existing road data, which can provide better and more accurate planning. The question I address here is "Are the existing functions in ARC GRID sufficient to locate forest roads?" If not, how can we make our own program

which can run in the ARC/INFO environment? This study has addressed these two questions.

2 LITERATURE REVIEW

A geographic information system (GIS) is a computerized database management system for the capture, storage, retrieval, analysis, and display of spatial data (Simkowitz, 1988). GIS has been used in transportation modeling (Lewis, 1990), highway analysis (Lutansky, 1989), and optimal transportation analysis zones (O'Neill, 1991).

GIS has also been used in computer-assisted forest road network analysis (Nieuwenhuis, 1987). The Washington State Department of Natural Resources (DNR), in cooperation with the University of Washington, has built an integrated harvesting planning system, which uses PC ARC/INFO and DNRGIS (the DNR Geographic Information System) to store transportation route coverages and generate road grade and road sideslope information (Cullen and Schiess, 1992). Epstein et al. (1994) made a forest operational planning software (PLANEX) for automating and optimizing the location of roads and landings. PLANEX is designed to accept files from ARC/INFO. Tucek (1994) used IDRISI (a GIS package) for digitizing forest topographic maps, checking the Euclidean distance from roads and displaying the perspective views of his study area. Tucek's study shows the possibility of using the GIS environment for opening-up forests. It would be desirable to use an available GIS package to do forest operational planning, because we can take advantage of the powerful data management and display abilities of GIS. However, a large GIS package usually can not meet all the requirements of a specific forest operational problem; therefore, there is the need to make custom programs. For example, Tucek (1994) made his own modules for calculating distances from each raster cell to the nearest defined object on the surface of the DTM in the direction of maximum slope gradient.

Digital terrain models (DTMs), as a data set of GIS, were introduced into forest harvest planning in the early 1970's. The term digital elevation model (DEM) is also commonly used. Burrough (1986) points out that because the term 'terrain' often implies attributes of a landscape other than the altitude of the land surface, the term DEM is preferred for models containing only elevation data. Burke (1974) suggested using a DTM to supply topographic data for harvest planning. Young and Lemkow (1976) developed a prototype timber harvest planning system that used a DTM. Their planning system, based on a limited computer system, suggested harvest planning using DTM data was practical. Twito et al. (1982) made an early version of a preliminary logging analysis system (PLANS) using DTMs. McGaughey (1992), author of the IBM-PC version of PLANS, points out that a fundamental concept underlying PLANS is the use of a digital terrain model to provide topographic data needed for harvest planning and transportation system development.

Reutebuch (1988) developed the ROUTES computer program using DTMs to assist preliminary locations of logging roads through mountainous terrain. This program simulates the field locator "walking" on the DTM surface by moving the digitizer cursor around the project-area map. Liu and Sessions (1993) studied preliminary planning of road systems using a DTM. They considered road hauling cost and road construction cost and used network analysis to choose among the possible road segments to form a transportation plan. Liu and Sessions (1993) point out that the DTM could not provide soil or slope stability information which affects road construction costs.

Jaeger and Becker (1995) tested four DTMs in part of the forest district of the Harz mountains (Central Germany) with a size of 70 ha. The first DTM is called the Terrestrial Model which is an exact copy of the topography of the area. An electronic theodolite was used to measure 6,292 terrain points. The

second DTM is called the Photogrammetric Model, which is a 10 meter spacing DTM generated from aerial photographs (scale 1:12,000 meter spacing). The third DTM is called the Atkis Model, a complete data set of a regional survey agency (12.5 m spacing grid). The fourth DTM is called the Topographic Model which was created by digitizing the 10 meter contour map with the scale 1:10,000. The last three DTMs were compared with the first. The Photogrammetric Model and Atkis Modle have mean of height differences less than 0.24 meter with a standard deviation less than 0.83 meter. The Topographic Model has a mean of height differences of 4.208 meter with a standard deviation of 12.279 meters. Their results show that the Photogrammetric and Atkis models provide terrain data with a high level of accuracy which they claim makes possible design construction projects such as roads in detail on the screen without major problems. However, the Topographic Model is not suited to this form of planning. They planned a route location on the Terrestrial Model and transferred the results to the terrain.

According to ESRI' s white paper (ESRI, 1995), GRID can solve optimal allocation problems. Examples include:

- Determining the optimal course of a new road over a nonnetworked surface, such as a soils map.
- Creating a drainage map from an elevation surface for input into an hydrologic model.
- Identifying along a road the locations that should be preserved for a visual quality assessment study.

3 OBJECTIVES AND SCOPE

3.1 OBJECTIVES

The objective of this paper is to explore the usefulness of existing "off-the-shelf" GIS software in preliminary forest road planning. Included in the objective, are four sub-objectives:

1. Develop strategies for using ARC GRID and ARC/INFO in forest road planning.

2. Make a program if necessary that can run in the ARC/INFO environment.

3. Locate forest roads from landings to existing roads with minimal construction cost and hauling cost.

4. Find a method to assist forest engineers in selecting landings.

At the present time, computer-based road planning models only use DTMs (Digital Terrain Models). Soil, vegetation, forest stand, stream and existing roads are not considered by these models. As GIS and remote sensing are developed, it is possible and necessary to consider these in forest road planning. Congalton, et al. (1994) produced a GIS database and map of old growth forest land on National Forest and Park lands in Oregon and Washington. Its GIS layers include slope, aspect, elevation, hydrology, location of research and inventory plots, crown closure, size class/stand structure, species, current vegetation type polygons, suitable lands for timber production, flight line maps, habitat conservation areas, forest boundaries, historical distribution of vegetation and old growth. Including these layers can make road planning by computer modeling more meaningful and useful.

As GIS technology is developing rapidly, it is worth doing research to discover the capability of existing GIS software for forest road planning. If the existing GIS software can do the forest road planning job well , then we can

make use of it instead of making our own programs, because making a good program usually requires much time and effort. If the existing GIS software can not do the forest road planning well, can we make a program which can run in the existing GIS environment? The objective of this study is to answer these questions.

3.2 SCOPE

In this study, the goal of road planning is assumed to find the good route from timber harvest landings to existing roads with minimal construction and hauling costs under the condition of 30 meter spacing DEM, soil, stream and stand data using ARC GRID functions. Landing selection is a complicated problem which is not within the scope of this study. However, a method for assisting landing selection has been developed in this study. For calculating road construction cost and hauling cost, it is necessary to know the timber volume in a landing which in turn depends on the logging system used and the stands in the area. In this study, we will assume that we know the timber volume coming into the landing and the type of logging system.

4 METHODS AND PROCEDURES

4.1 DATA COLLECTION

4.1.1 Selecting planning area

The planning area should be selected with sufficient GIS data, at least with DEM and existing road data. A planning area with DEM, existing road, stream, stand and soil data is recommended. Data could come from on-line GIS data bases, from air photos, satellite images, contour maps, soil maps or other data sources. The best way is from an on-line GIS data base. Usually, a GIS data set covers an area larger than a forest planning area. The following method is used to select a planning area from a GIS data set: First use ARCEDIT to make a BOX, then use the BOX to clip a planning area from each data set, such as roads, soils etc. The commands for creating a BOX (BOX.AML) are in Appendix A14. For clipping an area using BOX, use the ARC command CLIP (in this paper, commands are typed in capital letters. In actual use, the commands in ARC/INFO and ARC GRID are typed in lower case letters).

4. 1.2 Showing coverages

Use ARCPLOT to show the selected coverages. In order to show these coverages better, use the MAP command to map them together with the legend. Making a map can be done easily by an AML. The AMLs for making a map are in Appendices A5 to A8.

4.1.3 Converting Vector data to Raster (Grid) data

Usually, the soil, road, stream and stand data are in vector format in ARC/INFO. In order to use GRID, the data should be converted into raster (grid) format. this is easy to do so, because ARC/INFO has three commands for this: POINTGRID, LINEGRID and POLYGRID. All these three commands need a "cell size " argument which is determined by the cell size of the DEM.

For this study, a USGS 30 meter DEM is used, because it is easily available now.

To convert from DEM to GRID format, the ARC command DEMLATTICE is used. DEMLATTICE creates a lattice from Digital Elevation Models (DEM) in the USGS and TAME formats. Digital terrain data covering most regions of the United States are available in the USGS format from the U.S. Geological Survey. DEMs covering most regions of the world are available in the TAME and USGS formats from SPOT Image Corporation and STX Corporation.

Digital terrain data in the USGS format are available for two geographic extents: 7.5-minute quadrangles and 1-degree quadrangles. The DEMLATTICE command can convert either of these to a lattice. USGS 7.5-minute DEMs contain surface elevations at a constant spacing of 30 meters; the z values are usually expressed in meters. In contrast, 1-degree DEMs contain ground elevations at an interval of 3 arc-seconds; the z values are expressed in meters.

TAME DEMs are stored in the Terrain Access Made Easy format. The elevations are extracted from terrain models constructed from SPOT satellite image stereopairs using digital autocorrelation techniques. Typically, elevations are sampled at ten-meter intervals, the finest resolution of SPOT imagery (ESRI, 1992). If the 10 meter DEM is available, we only need to change cell size 30 into 10 when converting vector data to grid data.

One thing that needs to be pointed out is that when using the LINEGRID command, the last argument should be ZERO. Otherwise, only the line area has values; other areas are all NODATA. For example, if we convert a stream coverage into a grid, and do not use ZERO argument, then all streams will have values and the remaining areas have NODATA. In a GRID overlay

operation, if a cell has NODATA, the output will be NODATA, no matter what the other grid values are.

When using LINEGRID to convert a vector coverage to grid, the area might be smaller because of the length of roads or streams. One way to solve this problem is to use an overlay command in ARC/INFO, such as IDENTITY, to overlay the line coverage into a polygon coverage-for example, SOILS. After that, use the POLYGRID command to convert the vector coverage to grid. The input coverage will be the overlaid coverage and the line coverage's ID will be the converting item. The AML for converting vector data to a grid (POLY2GRID.AML) is in Appendix A3.

4.2 LANDING SELECTION

Landing selection is a complicated issue which relates to the logging system used, terrain, stand, timber volume etc. In this study, landing selection is not the main object, but we have made an AML to help forest engineers select landings. The LANDING.AML can select some possible landing areas considering the slope, soils and stand conditions. Existing roads and streams are not considered in the landing selection AML because these are shown on the screen when selecting landings. On the landing selection screen, contours, possible landing areas, existing roads and streams are shown. The final decision should be made by the forest engineers. The engineer could select a possible landing chosen by the AML, or the engineer could select a landing which the AML does not choose.

4.3 ROAD SELECTION

4.3.1 Three GRID functions

The algorithm for selecting the good road with minimal construction cost and hauling cost in this study is similar to the one used in ARC GRID (ESRI, 1994). In fact , it is the shortest path algorithm.

4.3.1.1 COSTDISTANCE function

COSTDISTANCE is a function in ARC GRID. In order to understand the algorithm used in my program, it is better to first describe briefly the COSTDISTANCE function. For detailed information, please see "Cell- based Modeling with GRID" (ESRI , 1992), or ARC/INFO version 7.0.3 on-line help ARCDOC.

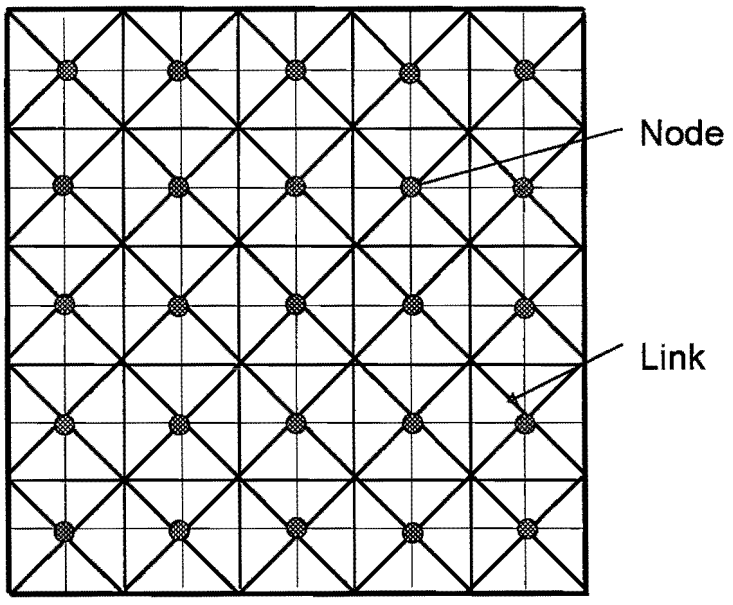


Fig. 1 Nodes and links: a view of a grid through graph theory.

The COSTDISTANCE function creates an output grid in which each cell is assigned the cumulative cost to the closest source cell. The algorithm utilizes the node/link cell representation. In the node/link representation, each center of a cell is considered a node and each node is connected by links to its adjacent nodes.

Every link has an impedance associated with it. The impedance is derived from the costs associated with the cells at each end of the link (from the cost surface) and from the direction of movement. When moving from a cell to one of its four directly connected neighbors, the cost to move across the links to the neighboring node is the sum of 1 times the cost of cell 1 plus the cost cell 2 divided by 2.

$$a1 = \frac{\cos t1 + \cos t2}{2}$$

where cost1 is the cost of cell1, cost2 is the cost of cell 2 and a1 is the length of the link from cell 1 to cell 2.

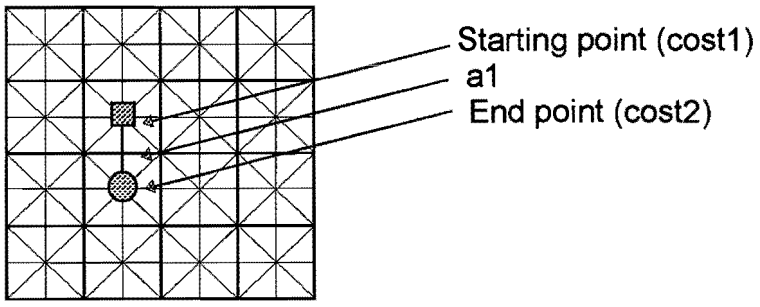


Fig. 2 Horizontal and vertical node calculations.

The cumulative cost is determined by the following formula:

$$cum\_\cos t = a1 + \frac{\cos t2 + \cos t3}{2}$$

where cost2 is the cost of cell 2, cost 3 is the cost of cell 3 and cum_cost is the cumulative cost to move into cell 3 from cell 1.

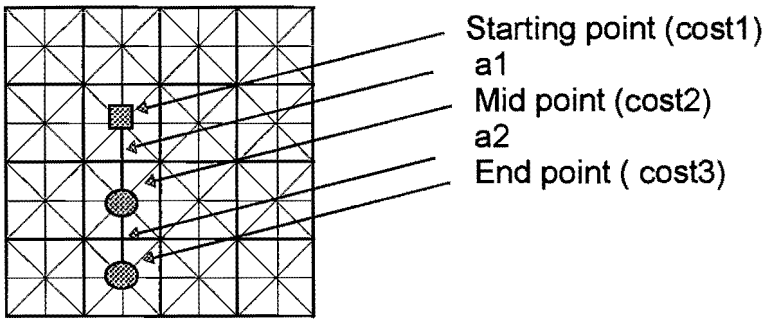


Fig. 3 Cumulative cost node calculations.

If the movement is diagonal, the cost travel over the link is 1.4142 ($\sqrt{2}$) times the cost of cell 1 plus the cost of cell 2 divided by 2.

$$a1 = \frac{1.4142(\cos t1 + \cos t2)}{2}$$

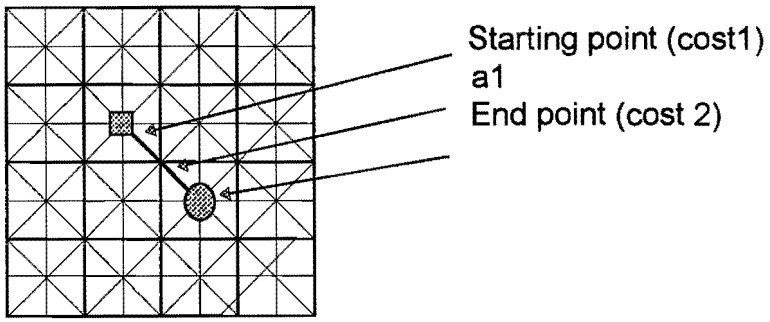
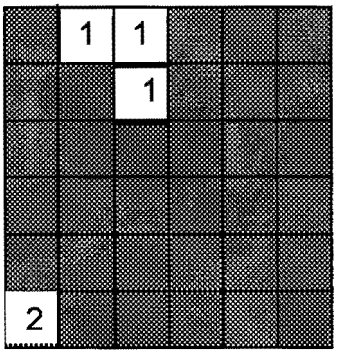


Fig. 4 Diagonal node calculations.

But when determining the cumulative cost for diagonal movement, the following formula must be used:

$$cum\_\cos t = a1 + \frac{1.4142(\cos t2 + \cos t3)}{2}$$

Creating a cumulative cost-distance grid using graph theory can be viewed as an attempt to identify the lowest cost cell and adding it to an output list. It is an iterative process that begins with the source cells. The goal is for each cell to be assigned quickly to the output cost-distance grid.



1	3	4	4	3	2
4	6	2	3	7	6
5	8	7	5	6	6
1	4	5		5	1
4	7	5		2	6
1	2	2	1	3	4

White Cells
Value = 0

Value = NODATA

Fig. 5 Source Grid and Cost Grid.

In the first iteration, the source cells are identified and assigned to zero since the source cell's neighbors are activated and a cost is assigned to the links between the source cell's nodes and the neighboring cells' nodes using the above cumulative cost formulas. Each of these neighboring cells can now reach a source; consequently, they can be chosen or assigned to the output

cumulative cost grid. To be assigned to the output grid, a cell must have the next-cost path to a source.

The cumulative cost values are arranged in a list from the lowest cumulative cost to the highest. The lowest cost cell is chosen from the active cost list and the value for that cell location is assigned to the output cost-distance grid. The list of active cells is now expanded to include the neighbors of the chosen cell, because

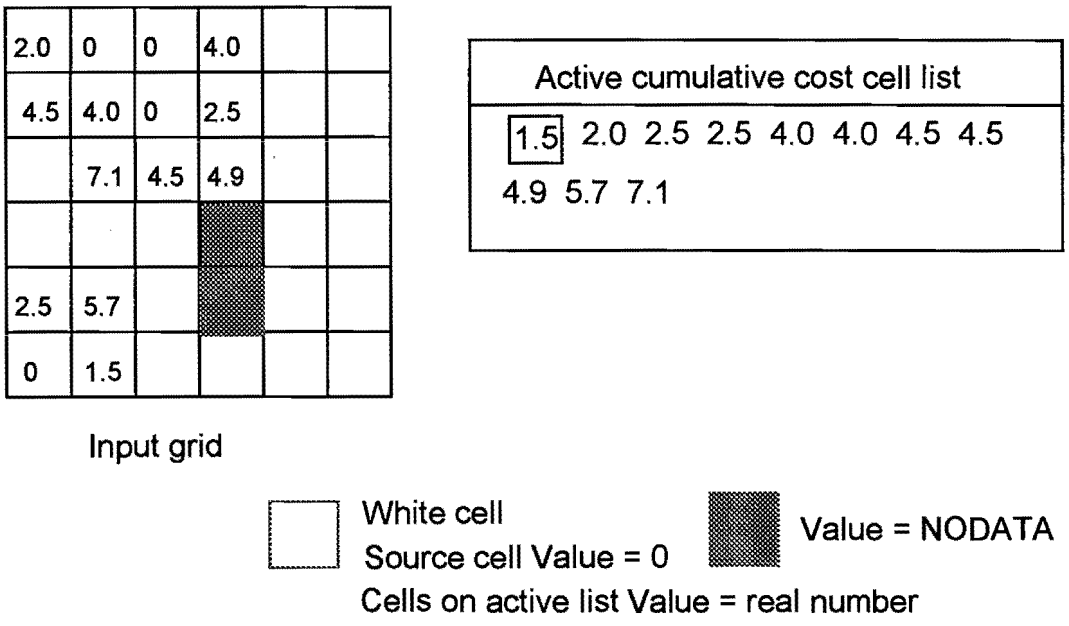


Fig. 6 Cumulative Cost Calculation.

those cells now have a way to reach a source. Only those cells that can possibly reach a source can be active in the list. The cost to move into these cells is calculated using the cumulative cost formulas.

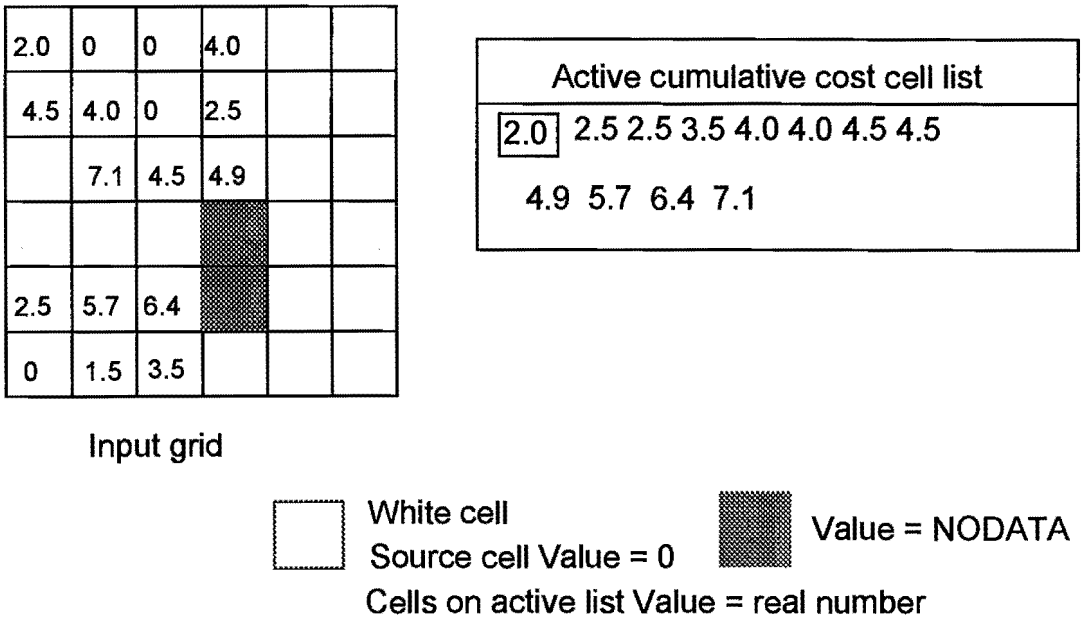


Fig. 7 Cumulative Cost Calculation.

Again, the active cell on the list with the lowest cost is chosen, the neighborhood is expanded, the new costs are calculated and these new cost cells are added to the active list. Furthermore, cells on the active list are updated if a new, lower cost route is created by the addition of new cell locations to the output grid.

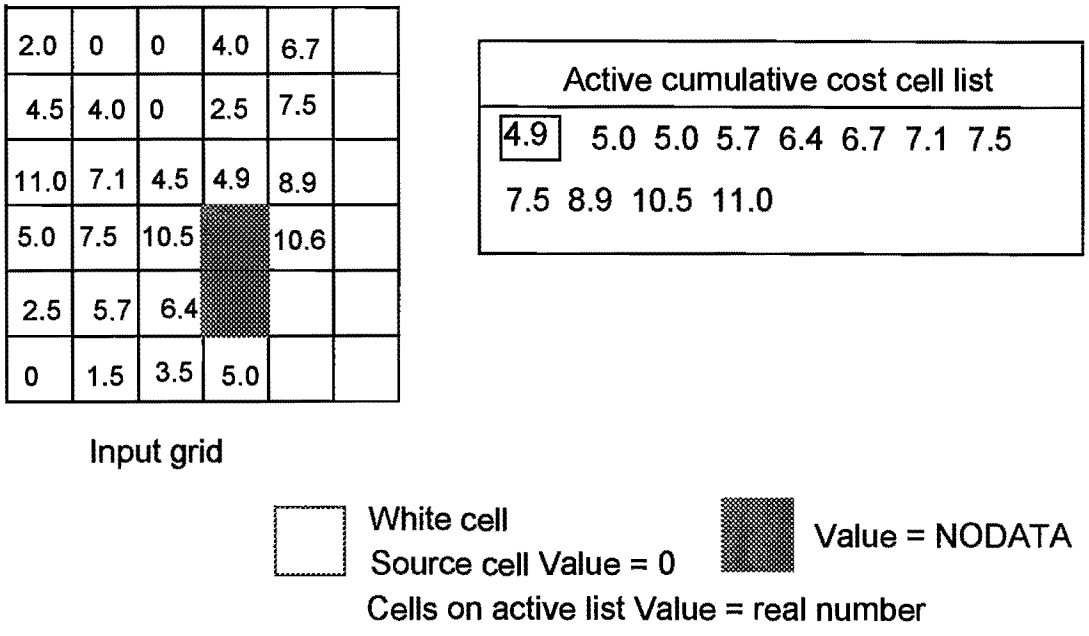


Fig. 8 Cumulative Cost Calculation.

This updating can occur with the advent of new paths for cells on the active list as more cells are allocated to the output grid. When the cell with lowest value on the active cumulative cost list is allocated to the output grid, all the cumulative costs are calculated.

If the new cumulative cost for the locations on the active list is greater than the one that the cell currently has, the value is ignored. If the cumulative cost is less, the old cumulative cost for the location is replaced on the active list with the new value. That cell, which has discovered a lower cost and more desirable path to a source, then moves up on the active chosen list. In the example below, the cell location at row 3, column 1 had a cumulative cost of 11.0 when it was put on the active list to reach the source at the top of the grid. But because the lower source expanded to this location, the cell had access to a lower cumulative cost (8.0) path to reach a source. The value for the location was updated on the active list and allocated to the output earlier, because of this lower cumulative cost .

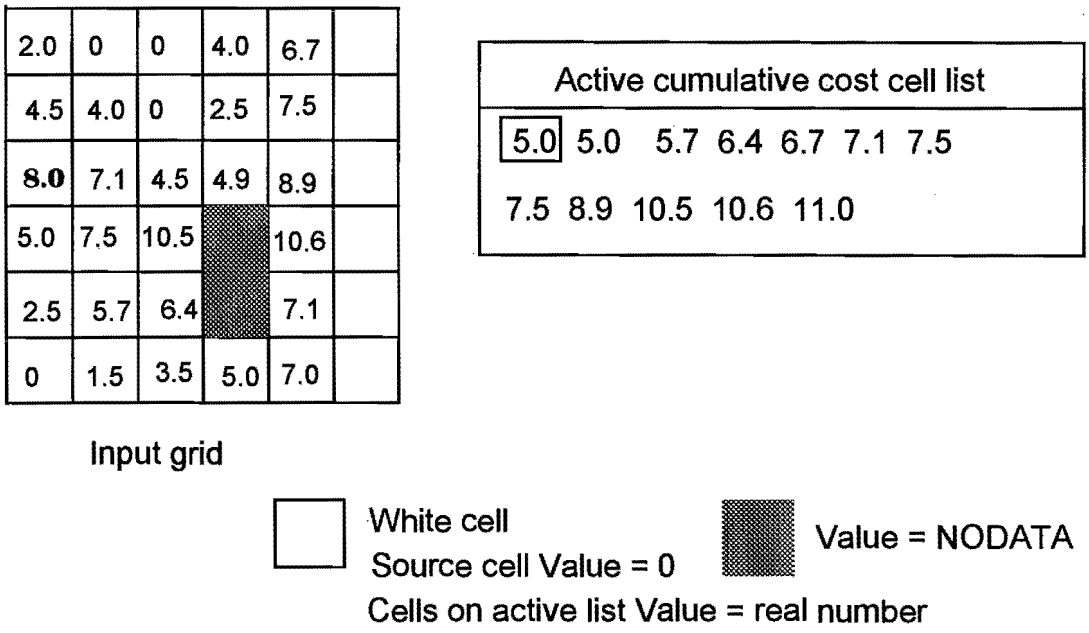


Fig. 9 Cumulative Cost Calculation.

If there are multiple zones or disconnected sets of source cells on the input source grid, the growing process continues and allocates the lowest cost cell from the active list, regardless of which source it is from. When the growth fronts meet, the least-cost path back to the source proceeds until all eligible cells have received a cost value.

When all cells have been chosen from the active list, the result is the cumulative-cost or weighted-distance grid. The procedure used ensures that the lowest cumulative cost is guaranteed for each cell. This process continues for all cells until the edge of the grid is encountered, the boundary of the window is found or the maximum distance is reached.

2.0	0.0	0.0	4.0	6.7	9.2
4.5	4.0	0.0	2.5	7.5	13.1
8.0	7.1	4.5	4.9	8.9	12.7
5.0	7.5	10.5		10.6	9.2
2.5	5.7	6.4		7.1	11.1
0.0	1.5	3.5	5.0	7.0	10.5

Fig 10 Cost distance output grid.

6	7	8
5	0	1
4	3	2

a) Back-link positions

1	0	0	5	4	5
7	1	0	5	5	6
3	8	7	6	6	3
3	5	7		3	4
3	4	4		4	5
0	5	5	5	5	5

b) Cost allocation output grid

Fig. 11 Road Directory Grids.

The cost-distance grid identifies the cumulative cost for each cell to return to the closest cell in the set of source cells. It does not show which source cell to return to or how to get there. The cost back link returns a grid with a value range from 0 to 8 that can be used to reconstruct the route to the source. Each value (0 through 8) identifies which neighboring cell to move into to get back to the source.

If the cell is assigned '5' as part of the least cost path to a source, the path should move to the left neighboring cell. If the cell has '7', the path should then move due north.

According to the description above, we made a test program TEST.C. I used the same source grid and cost grid as in the above example and got the same result as in the above example. My TEST.C program is in Appendix B2 and the result in Appendix C1.

4.3.1.2 COSTPATH function

Once the cumulative cost and back-link grids are created. least-cost path routes can be derived from any designated destination cell or zone(s). The COSTPATH function retraces the destination cells through the back-link grid to a source.

I have tried using the COSTDISTANCE and COSTPATH function to locate forest roads. We used SLOPE grid as the cost grid. Slope in GRID is the maximum average slope of the eight grids of the processing grid. In this way, the COSTDISTANCE function can find the best route from a landing to a road cell with minimal average slope; but the road selected can not follow along the contour line. One difficulty in using GRID to locate forest roads is how to assign the cost grid for the COSTDISTANCE function. In the cost grid, each cell has a cost value which means that when a path is passing the cell, it will cost so much, no matter what direction the path comes from. In the words of ESRI[1], each grid cell location is given a weight proportional to a relative cost which is incurred by the phenomena being modeled when passing through a cell. The weightings are usually based on inherent features in the location that are static

[1] Euclidean, cost distance and other global functions, p26, Cell-based Modeling with GRID ARCDOC online help version 7.

prior to the movement of the feature or phenomena. In forest road location, the direction does really matter. From one direction to a cell, the slope might be over 35% and has much earthwork, but from another direction, following the contour line for example, the slope might be less than 5% and has little earthwork.

4.3.1.3 PATHDISTANCE function

The PATHDISTANCE function is one of several GRID tools used for distance analysis. PATHDISTANCE is similar to COSTDISTANCE in that both functions determine the minimum cumulative-travel cost from a source to each cell location grid. But PATHDISTANCE not only calculates the cumulative cost over a cost surface, it does so while compensating for the actual surface distance that must be traveled and the horizontal and vertical factors influencing the total cost of moving from one location to another. The cumulated-cost surface produced by PATHDISTANCE can be used in dispersion modeling, flow movement and least-cost path analyses (ESRI, 1992).

We have also tried to use PATHDISTANCE to locate forest roads. PATHDISTANCE has a horizontal factor which takes care of the total cost of moving into a cell by accounting for any horizontal friction encountered but it seems impossible to me to use it to calculate earthwork. Therefore, If we only consider the road slope and do not care about earthwork, PATHDISTANCE can be used for forest road preliminary planning. The way of doing so is to consider only the vertical factor in the PATHDISTANCE function. By using the DEM as a surface factor grid and establishing an ASCII file considering the effect of slope as a vertical factor parameter, PATHDISTANCE can find the good route. The AML using PATHDISTANCE to locate roads is in Appendix A13. The example result and vertical table used are in Appendices C2 and C3.

The vertical table is made according to the variable cost calculation in section 4.3.2.3 below.

4.3.2 ROAD.C program

4.3.2.1 The algorithm

As a result of exploring the three GRID functions mentioned above, we decided to write my own program which considers both slope and earthwork. The program runs in the ARC/INFO environment. The program's outputs, after converting them to grids, can be used in the COSTPATH function.

The algorithm for ROAD.C is similar to that described in the COSTDISTANCE function.

4.3.2.2 The flow chart of ROAD.C

The ROAD.C program (Appendix B1) works as follows:

1. Read in ASCII files: DEM, road, soil, stream, and template data.
2. Calculate construction cost and variable cost for the neighboring cells of a road cell.
3. Find the cell with the smallest construction and variable cost (total cost). This cell is called the selected cell.
4. Put the value and the location of the selected cell to output arrays.
5. Ask whether it is the last selected cell.
6. If no, calculate the construction cost and variable cost of the neighboring cells of the selected cell, go to step 3.
7. If yes, write out the output arrays as ASCII files.

ROAD.C runs in 28 seconds on the SUN workstation SPARC server 20, and in 42 seconds on the SUN workstation SPARC server 10. The size of the study area is 1657 acres (105 x 71 cells).

The flow chart of ROAD.C is shown in Fig. 12.

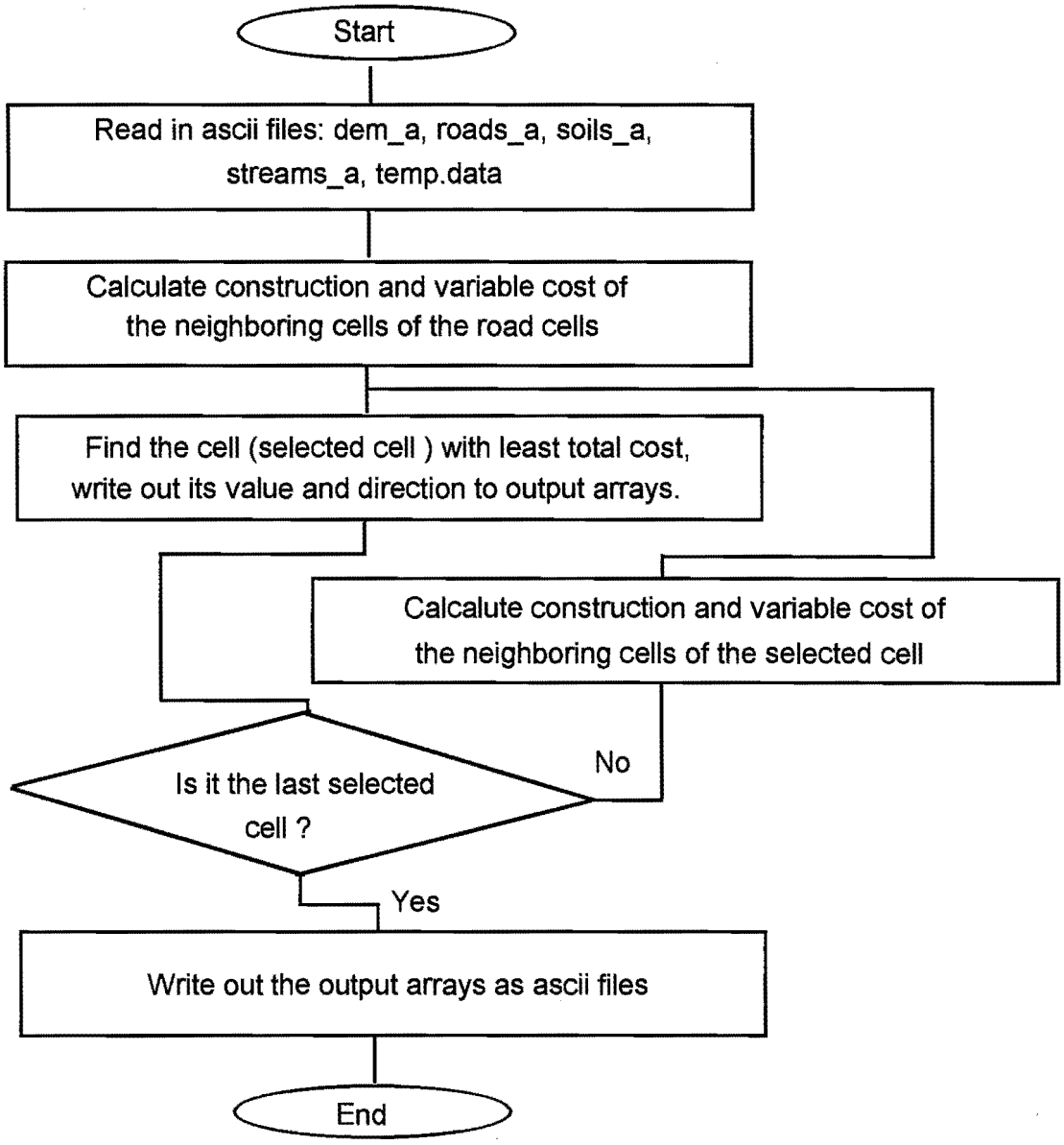


Fig. 12 Flow chart of ROAD.C.

4.3.2.3 Main functions of ROAD.C

1. Get data function

To get data, function (get_data) reads 5 ASCII files into the program. Four files are converted from grids using the GRIDASCII command in ARC/INFO. These files are: DEM (hdem_a), roads (roads_a), soils (soil_a), and streams (stream_a). Each file is stored in a two dimensional array. The array's dimensions (row and column) are determined by the ASCII file header. The ASCII file header comes from the grid, which contains row, column, longitude, latitude, cell size and NODATA information. If these files have a different numbers of rows and columns, the output grid of this program will take the smallest of the rows and columns. It is just like the grid overlay operation: if two grids with different numbers of rows and columns overlay, the smaller number of rows and columns will be taken. Another ASCII file (temp.dat) contains the data about road template and prices of construction and hauling. If the user wants to change these data, he can use a text edit to do so. In this way, ROAD.C does not need to be compiled again.

2. Total cost calculation function

This function calculates the construction cost and variable cost for each road section (cell). It starts from a cell (for the convenience of understanding, I use the term cell and grid here, though it is a number in an array), first judging whether this cell's value has been sent to the output grid. If not, it takes one neighboring cell (8 neighboring cells, Fig. 13) , and calculates the construction cost and variable cost if the neighboring cell does not exceed the boundary of the grid. The formula and methods used to calculate construction cost and variable cost are taken from SNAP (Sessions, 1994).

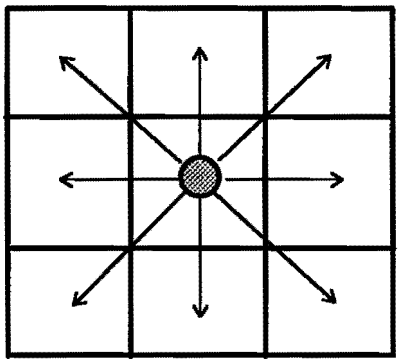
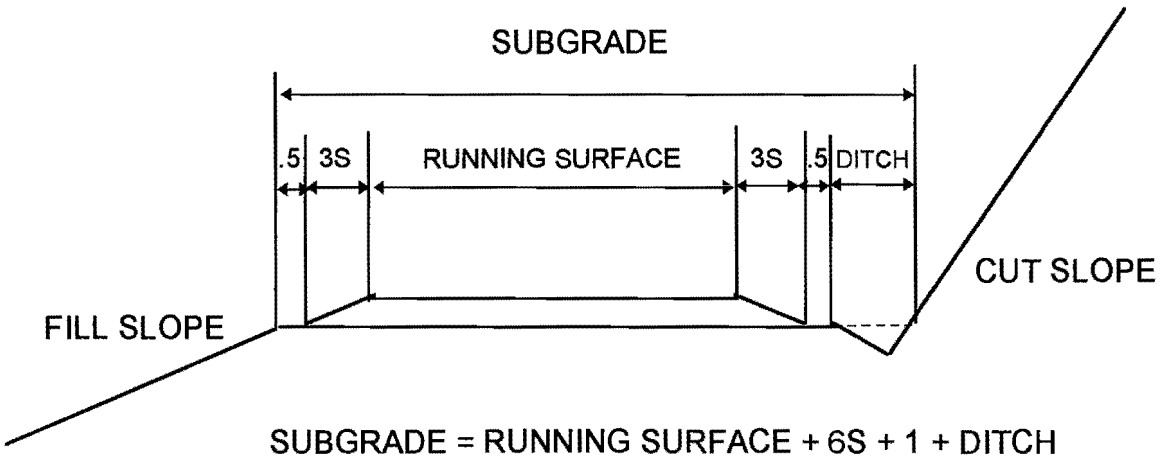


Fig. 13 Possible road directions.

a) Construction cost calculation

The cross section of the road section used in ROAD.C is the same as the one of the road sections used in SNAP (Fig. 14).



SUBGRADE = RUNNING SURFACE + 6S + 1 + DITCH

Fig. 14 Road Cross Section Used in ROAD.C.

(Adapted from SNAP II USER'S GUIDE Ver. 2.071.)

Road sections are assumed to be either balanced or full bench. If the slope is greater than 60%, the full bench section will be used. As with SNAP,

ROAD.C assumes that if a full bench will be built, endhaul will begin. Fig. 15 shows road slope and ground slope calculation. For example, the formulas for calculating slope and ground slope are as follows:

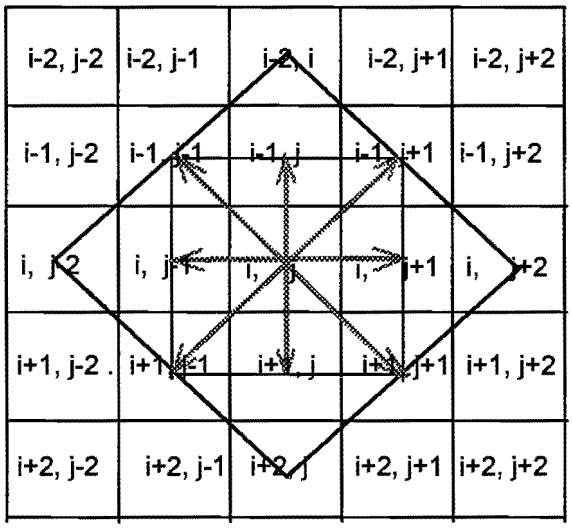


Fig.15 Slope and earth work calculation.

——— Earth work calculation

▬▬▬ Slope calculation

$$slope(i, j+1) = \frac{(DEM(i, j) - DEM(i, j+1))}{Cellsize} \times 100$$

$$gd\_slope(i, j+1) = \frac{asb(DEM(i-1, j+1) - DEM(i+1, j+1))}{Cellsize}$$

Where: DEM(i, j) - Elevation in cell (i,j)

asb ( ) - absolute value

gd_slope - ground slope

slope angles are in percent, but ground slope angles are in decimal percent.

If the direction is diagonal, the cell size for slope calculation is *1.4142*cell-size* and for ground slope calculation is *2*1.4142*cell-size*.

Once we calculate the ground slope and the road template data such as fill slope, cut slope, subgrade width etc., we can calculate the earthwork as follows (Sessions, 1994):

Balanced Section

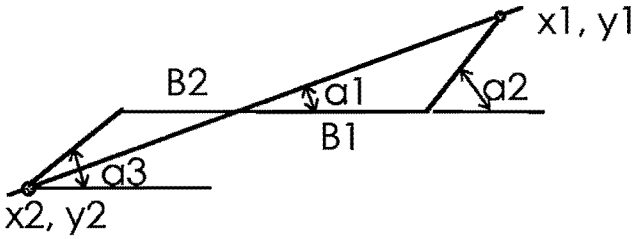


Fig. 16 Cut and Fill in Balanced Section.

$$cut = {B_1 y_1}/{2} = {B_1 \alpha_1 x_1}/{2} \qquad y_1 = \alpha_1 x_1 = \alpha_2 (x_1 - B_1)$$

$$Fill = {B_2 y_2}/{2} = {B_2 \alpha_1 x_2}/{2} \qquad y_2 = \alpha_1 x_2 = \alpha_3 (x_2 - B_2)$$

$$B_1 = \{(\alpha_2 - \alpha_1)/\alpha_2\} x_1 = D x_1$$

$$B_2 = \{(\alpha_3 - \alpha_1)/\alpha_3\} x_2 = G x_2$$

and $B_1 \alpha_1 x_1 = k B_2 \alpha_1 x_2$ Where k is the shrinkage factor; relationships for x_1, x_2 are:

$$x_1 = w/D - (G/D) x_2$$

$$(kG - G^2/D) x_2^2 + 2(wG/D) x_2 - w^2/D = 0$$

Where: w - subgrade width, $w = B_1 + B_2$

α_1 - ground slope

α_2 - cut slope

α_3 - fill slope

X_2 is solved by the quadratic formula and the other output quantities follow directly.

Full Bench Section

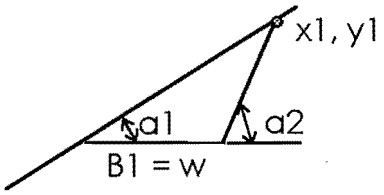


Fig. 17 Cut in Full Bench Section.

$$cut = {B_1 y_1}/{2} = {B_1 \alpha_1 x_1}/{2} = {w \alpha_1 x_1}/{2}$$

$$y_1 = \alpha_1 x_1 = \alpha_2 (x_1 - w)$$

$$x_1 = \left\{ {\alpha_2}/{(\alpha_2 - \alpha_1)} \right\} w$$

All angles in earthwork calculation are in decimal percent. For example a 30% ground slope is recorded as 0.30; a 0.5:1 cut slope is recorded as 2.0, a 1.5:1 fill slope is recorded as 0.67.

In ROAD.C, type of excavation does not need to be assumed. For every cell, ROAD.C will check the soil type of that cell. If it is common soil, it will use common excavation cost. If it is rock, it uses rock excavation cost. In ROAD.C, only two types of excavation cost are used. If the information about type of excavation cost is available, it is easy to use excavation cost considering the soil types.

For bridge cost, ROAD.C will check the stream data at every possible road direction. If the selected cell crosses the stream, the cell will be assigned the bridge cost. In ROAD.C, the roads to be selected are allowed to cross streams, because bridges can be built. But ROAD.C does not allow roads to go

along the cells where the streams are. This is done by checking two cells in the stream layer. If the two cells (in the same road direction) are both stream data, then it will not use this direction's selection.

b) Variable cost calculation

The variable cost formula used in ROAD.C is as follows:

$$\text{var cost} = Tv * Tc/60 * t/5280 * d/Lt$$

varcost - Log hauling cost per link ($)

Tc - Estimated truck cost ($/hr)

t - tranportation time per mile (min./mile)

d - link length (ft)

Lt - Load per truck (ton)

Tv - Total volume in a landing (ton)

Transport time per mile is determined by the combination of road type and grade. The total transport times allow for the loaded truck to travel in one direction on the road segment and for the unloaded truck to return empty in the other direction. The transport time used in ROAD.C is from SNAP II (Sessions, 1994). Table 1 shows the transport time for trucks in a single lane, gravel, with turnouts and ditch fair alignment.

Table 1 Truck travel Time

Grade (loaded direction)	Time (minutes/mile)	Grade (loaded direction)	Time (minutes/mile)
1	5.1	0	5.1
2	5.1	-1	5.1
3	5.8	-2	5.1
4	6.4	-3	5.1
5	7.1	-4	5.1
6	7.8	-5	5.1
7	8.5	-6	5.1
8	9.1	-7	5.1
9	9.8	-8	5.1
10	10.7	-9	5.3
11	11.6	-10	5.6
12	12.5	-11	6.0
13	13.4	-12	6.4
14	14.3	-13	6.8
15	15.2	-14	7.2
16	16.1	-15	7.6
17	17.0	-16	8.0
18	17.9	-17	8.4
		-18	8.8

Links with grades over 18% (adverse or favorable) are assigned a very high cost. This allows for a road's accessibility to a landing if there is no other way to it, but with a much higher cost. It is necessary for the source to be connected in some way to the destination for the shortest path algorithm to work but a very high cost will signal the solution is not feasible.

c) Total cost

The total cost is the sum of the construction cost and variable cost. Similar to the GRID (ESRI, 1992) algorithm, ROAD.C calculates the cumulative cost as follows:

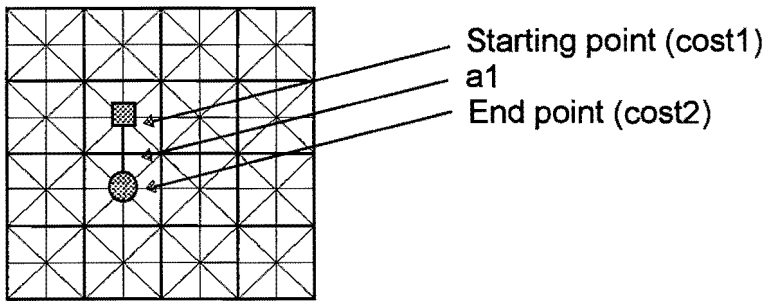


Fig.18 Horizontal and vertical node calculations.

$$a1 = Total\_cost1 + Total\_cost2$$

where Total_cost1 is the total cost of cell1, Total_cost2 is the cost of cell 2 and a1 is the value of the link form cell 1 to cell 2.

The cumulative cost is determined by the following formula:

$$a2 = a1 + Total\_cost3$$

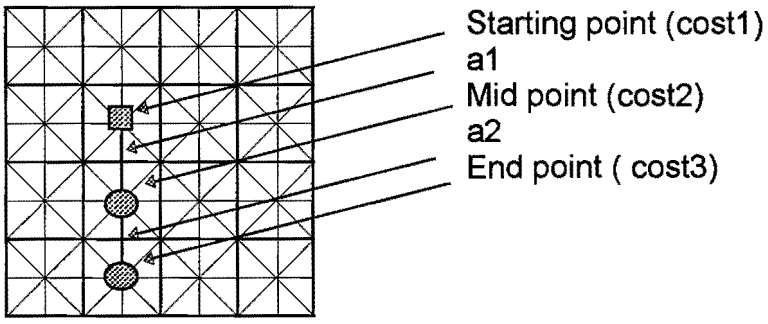


Fig. 19 Cumulative cost node calculations.

If the movement is diagonal, the distance factor 1.4142 is considered in construction cost and variable cost calculations.

3. Find the cell with smallest total cost function

The program first calculates the construction cost and variable cost for the neighboring cells of the road cells. Then it finds the cell with smallest cost (selected cell). In the COSTDISTANCE function, it is the smallest number in the active cumulative list. After that, it puts the value of the selected cell into an output grid named "earth-out", and the location of the selected cell to the other grid called "direct-out". The location of the selected cell is then passed to the total cost calculation function to calculate the construction and variable cost of the neighboring cells of the selected cell. Then, the Find-small function works again. The same procedure continues until the last selected cell is found.

4. Put data function

This function writes out two ASCII files for GRID. The first one is "earth-out", which has three types of values: 0 represents the source cell (road); -9999 represents the NODATA; and the remaining positive numbers represent the smallest cumulative cost from the cell to a source cell (road). The second output file is "direct-out". The direct-out" file contains the travel direction of the cell to a source cell.

4.3.3 DO-ROADS.AML

DO-ROADS.AML (Appendix A1) is the control part of this road planning program.

The DO-ROADS.AML works as follows:

1. Sets the display screen using command DISPLAY 9999 3, which is the full screen display;

2. Displays the existing roads and slope grids;

3. Displays a contour map and possible landing areas for the engineer to select;

4. After the landing is selected, runs ROAD.C to get "earth-out" and "direct-out" ASCII files;

5. Converts "earth-out" and "direct-out" as grids using command ASCIIGRID;

6. Runs the COSTPATH function to get the route from the landing to a road cell and shows the route on the screen in grid form;

7. Adds the selected road to existing roads;

8. Converts the new existing road in vector form, using the command GRIDLINE;

9. Asks the user if the planning is to continue, if yes, go step 2.

10. If no, show the new existing road on the screen and stop.

The user can select one landing or several landings at a time. The advantage of selecting one landing at a time is that the next selection can treat the selected road as an existing road. The disadvantage of selecting one landing at a time is the longer running time. The problem in selecting more than one landing at a time is that all the selected landings will be connected to the original existing roads. That is, there is no opportunity for the landing to use a proposed road which connected another landing to the existing road systems. This might not be economical. When selected landings are obviously far away (one could not make use of another if the other were built earlier), it is move time efficient to select more than one landing at a time.

Every time an existing road is shown, the contour map follows, so that the user can evaluate the selected road according to the contour map.

In step 7, the selected road cannot be added to the existing road directly, because NODATAs are in these two grids. For example, in the selected road grid, only the cells showing the route have values; all other cells are NODATA, the same pattern as in the existing road grid. Therefore, if we add these two grids together, the output grid will be all NODATA. The ISNULL command can be used to solve this problem, ie, ISNULL the existing road grid and selected road grid first, then add them together to form a new existing grid.

DO-ROADS.AML runs in about 2.5 minutes on a SUN SPARC 20 workstation and about in 3.5 minute s on a SUN SPARC 10 workstation. The size of the study arces is 1657 acrs (105 x 71 cells).

The flow chart of the DO-ROADS.AML is shown in Fig. 20.

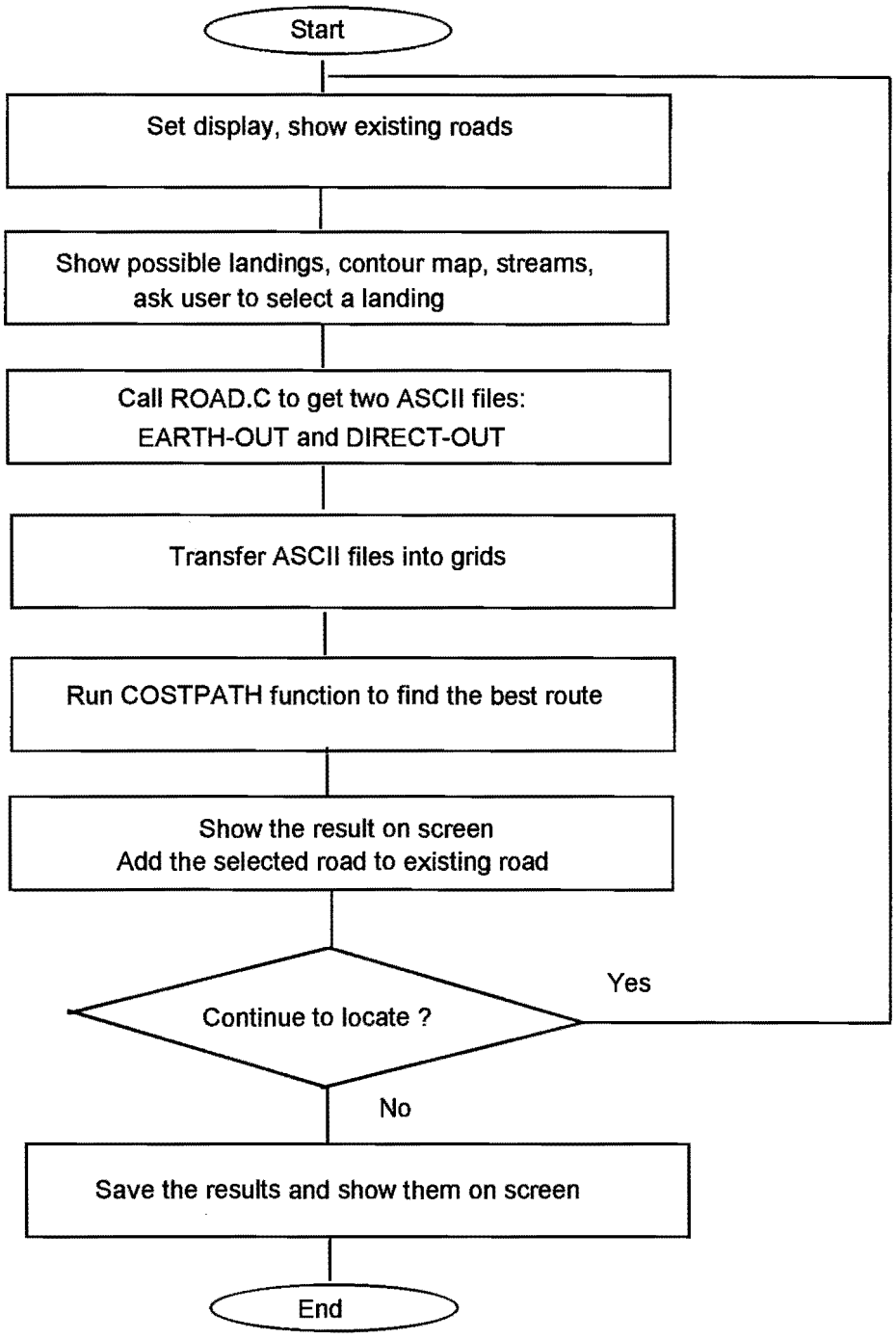


Fig. 20 Flow chart of DO-ROADS.

4.3.4 FINAL MAP AND STATISTICS

A map of the new existing road (selected road and the original existing road) was made using ROAD_MAP.AML. The final road map, including contours and streams can be used to do field testing. During road selection, each intermediate selection and the final results are saved. After the final selection has been made, each saved selection will be converted to vector form. The final maps are of two kinds: one shows the existing roads and the selected roads in grid form; the other shows the existing roads and selected roads in vector form. In the vector form map, each road selection is shown in different color or line symbols. The selection number (order) of each road indicates the building sequence of the road. In the GRID operation, it is easy to show the total cost of the selected road, using the CELLVALUE command , but it is difficult to show the length of the road easily and directly. Therefore, the length of each selected road is calculated in ARCPLOT using the STATISTICS command.

5 APPLICATION

5.1 THE SELECTED AREA

5.1.1 H.J. Andrews Experimental Forest

A study area from the H. J. Andrews Experimental Forest was selected because that the GIS data (DEM, roads, soils, streams. and stands) are available on-line.

The H. J. Andrews Experimental Forest is located in the rugged Cascade Mountains approximately 50 miles (80 km) east of Eugene, Oregon. It is 15,815 acres (6400 hectares) in size and ranges from 1350 feet (412 m) to 5350ft (1630 m) in elevation. The landscape is deeply dissected and heavily forested. Pristine stands of old-growth forest with dominant trees in excess of 400 years of age cover about 45 percent of the Andrews Forest with the remainder in younger age-class forests; the most common forest types at lower elevations are Douglas-fir, western hemlock and western red cedar. Going up in elevation, western hemlock is gradually replaced by Pacific silver fir. Upper elevation stands consist of a mixtures of true firs and mountain hemlock. Approximately one-third of the Andrews Forest has been logged or used for research.

The maritime climate is mild with wet winters and cool, dry summers. Annual precipitation normally exceeds 100 inches (2540 mm) and is concentrated in the winter. Deep snowpacks are common above 3300 feet (1000 m). Little or no rain falls during July and August. Rapidly flowing mountain streams are the primary type of aquatic ecosystems in the Andrews Forest. Streamflow follows the precipitation pattern with winter maximum flows three orders of magnitude larger than summer minimum flows.

Soils in the area have loamy surface horizons, ranging from silty-clays to sandy and gravely loams. Because of aggregation of primary soil particles by

organic matter and other agents, porosity of surface soils is 60 to 70 percent, over half of which is macro space (Swanson et al, 1987).

5.1.2 Data on the selected Area

The selected area is from the middle lower part of the H. J, Andrews Experimental Forest. It is 1657 acres (671 hectares) in size. The reasons for selecting this area are: 1) terrain good for cable logging (most of the area's slope is over 35%), 2) numerous streams and existing roads, and 3) numerous forest stands. All the GIS layers used in this study are from the Corvallis Forestry Sciences Laboratory Network, directory /data/cascade/hja. The GIS data format is ARC/INFO.

5.1.2.1 DEM

The DEM data are named HJALATTICE in the directory. The HJALATTICE is in grid format, converted from USGS DEMs by the DEMLATTICE command in ARC/INFO. The related data: 10 meter and 50 meter contours (CONTOUR10, CONTOUR50) are available. The SLOPE grids are also available there. There are two slope grids on-line: one is in percent format (slppct_g) and the other in degree format (slpdeg_g). The DEM in the selected area is clipped from the H. J. Andrews' DEM using command LATTICECLIP.

5.1.2.2 Stands

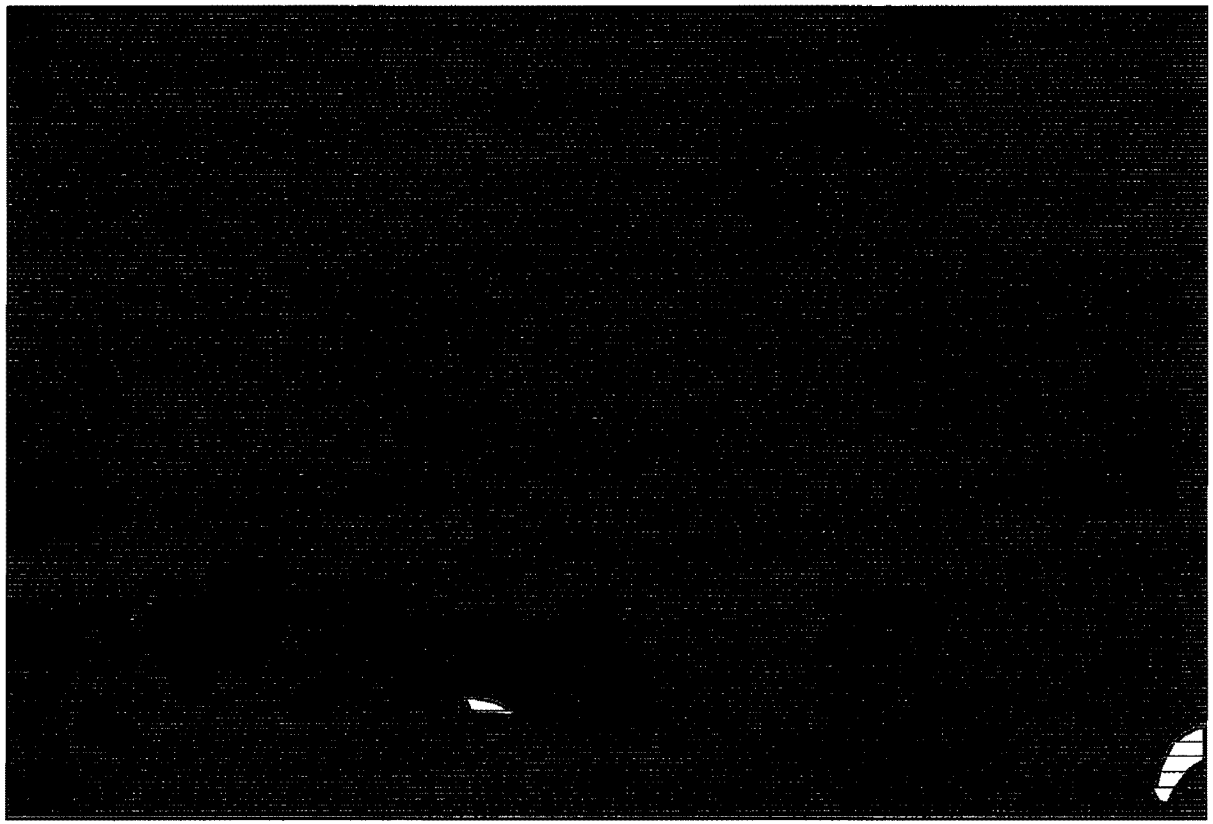
The stand data of the selected area are on the WNFVEG layer which contains vegetation, managed stands and natural vegetation. There is an on-line document WNFVEG.DOC describing this layer in the same directory mentioned above. When we converted the vector data to grid, we used the FEATURE-CODE (vegetation code) in this layer. The FEATURE-CODE in this

layer are ASCII characters , but the POLYGRID command needs numerical codes, therefore, we used the numerical code instead. The character and the numeric codes are listed in Table 2.

Table 2 Feature Code

C (1)	managed stand (clear cut or overstory removal)
S (2)	managed stand (shelterwood)
P (3)	managed stand (partial cut - with or without regeneration)
N (4)	natural forested stand
D (5)	developed land
M (6)	meadow (less than 10% tree cover)
R (7)	rock
W (8)	water
E (9)	exchanged land, previously private (often no data)
O (10)	outside proclaimed forest boundary
X (11)	sliver

In the selected area only four types of stands exist: 1) code C, managed stand; 2) code N, natural stand; 3) code M, meadow; and 4) code R, rock. The STAND.MAP is show in Map 1.



ROAD PLANNING PROJECT

Stand Map

This map shows the stands in the area.

Legend

- Clear cut
- Natural
- Meadow
- Rock

SCALE 1:20,000

0 1250

ft

N

5.1.2.3 Soils

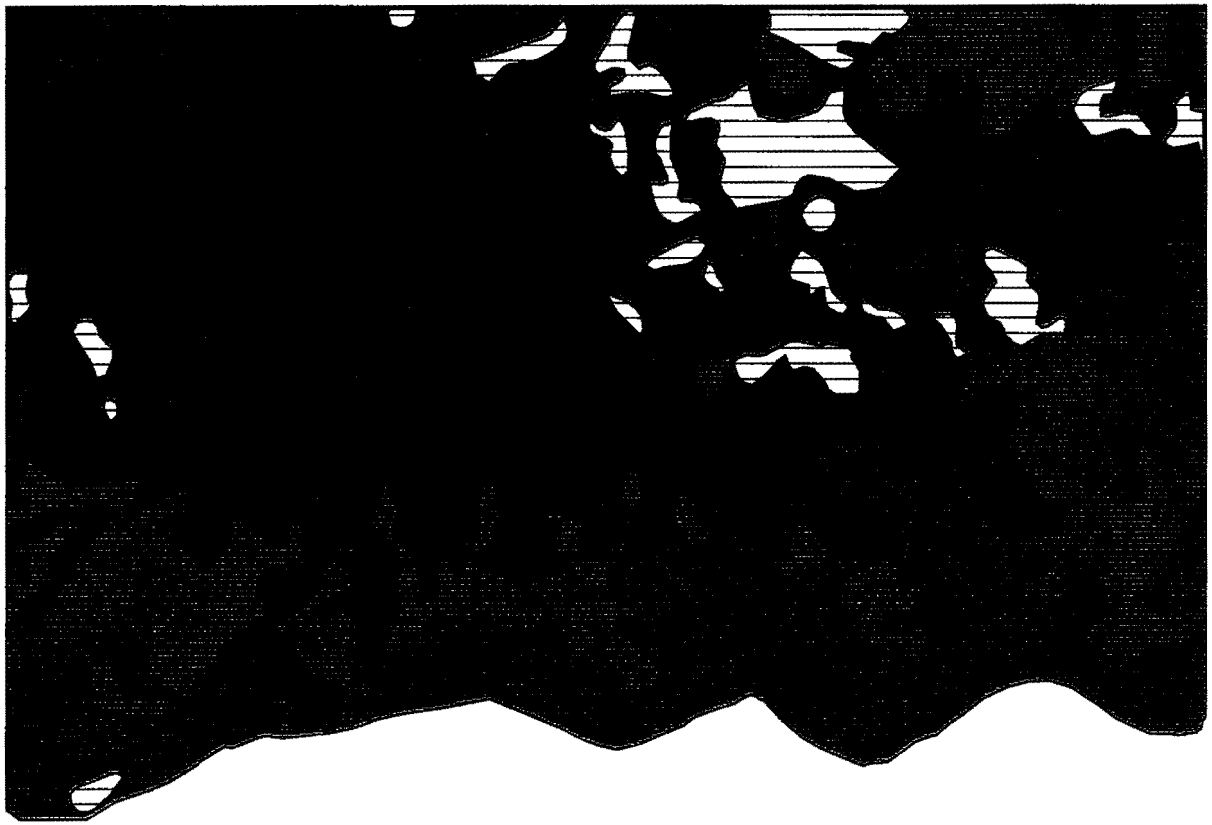
The soils GIS layer is named SOILS64 in the network. There is a document Soils64.doc in the same directory. TEXTURE CODE was used when converting vector data to grid form.

Table 3 Texture code

1	quarries
2	cobbly_sandy_loam
3	gravelly_loam
4	dark_poorly_drained
5	gravelly_sandy_loam
6	bedrock_talus
7	cobbly_heavy_loam
8	gravelly_clay_loam
9	light_clay_loam

In the selected area only four types of soils exist: 1) code 3, gravelly_loam; 2) code 4, dark_poorly_drained 3) code 5, gravelly_sandy_loam; and 4) code 6, bedrock_talus.

In ROAD.C, rock type soil code (RockCode) is needed when calculating the earthwork. In this example, the RockCode is 6. This code should be put in Temp.dat file. Considering that soil data are now difficult to get, ROAD.C is designed not to consider soil data if they are not available. If the soil data are not available, a -9999 (NODATA) is used for RockCode in Temp.dat file. The SOIL.MAP is shown in map 2.



ROAD PLANNING PROJECT

Soil Map

This map shows the soil texture in the area.

Legend

- gravelly loam
- dark poorly drained
- gravelly sandy loam
- bedrock talus

SCALE 1:20,000

0 1250

ft

N

5.1.2.4 Roads

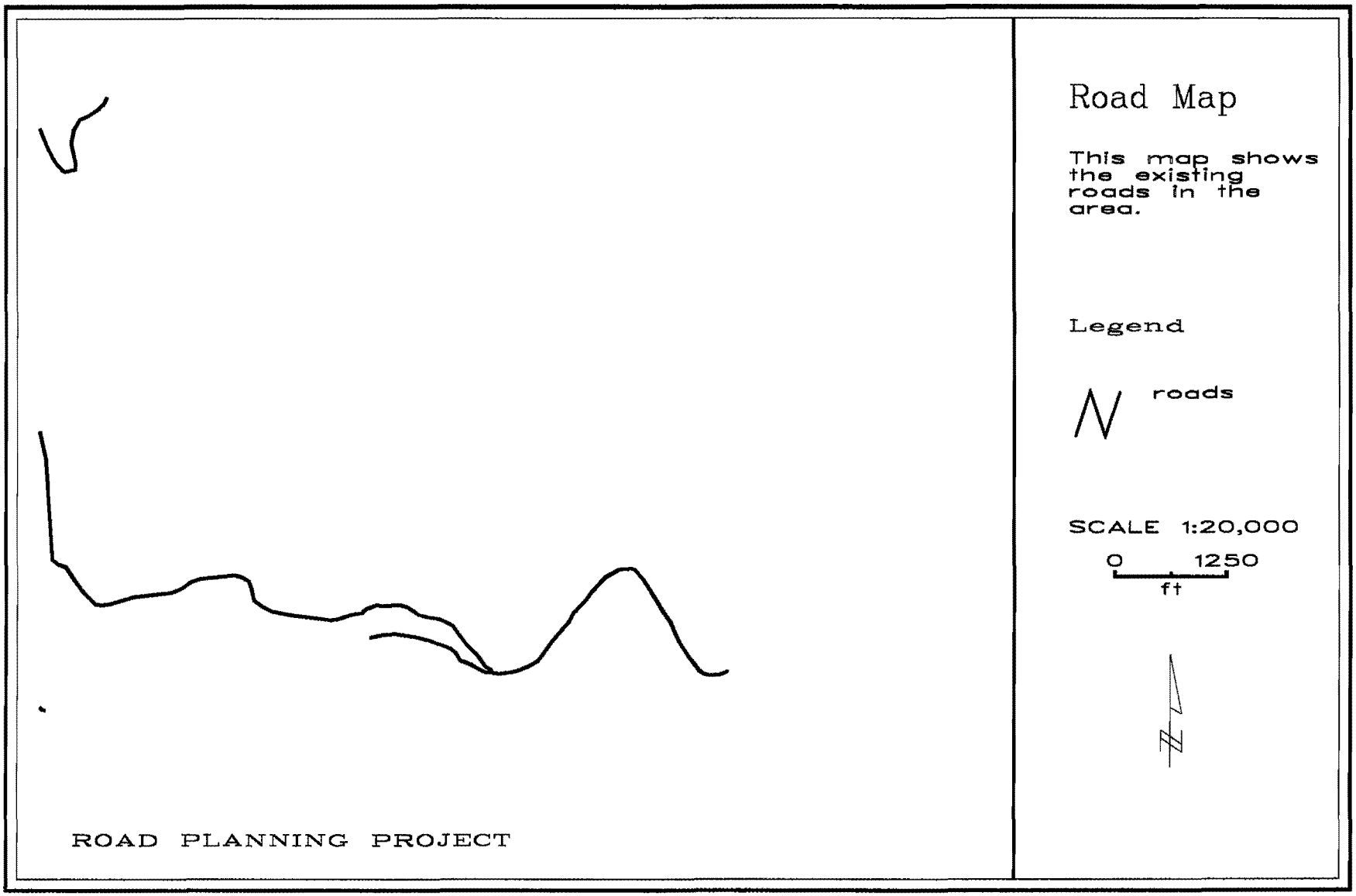
The road data are in the HJAROADS layer. There are three sections of existing roads in the selected area. The total length of the existing roads is 14258 ft (4346 m). The ROADS.MAP is shown in Map 3.

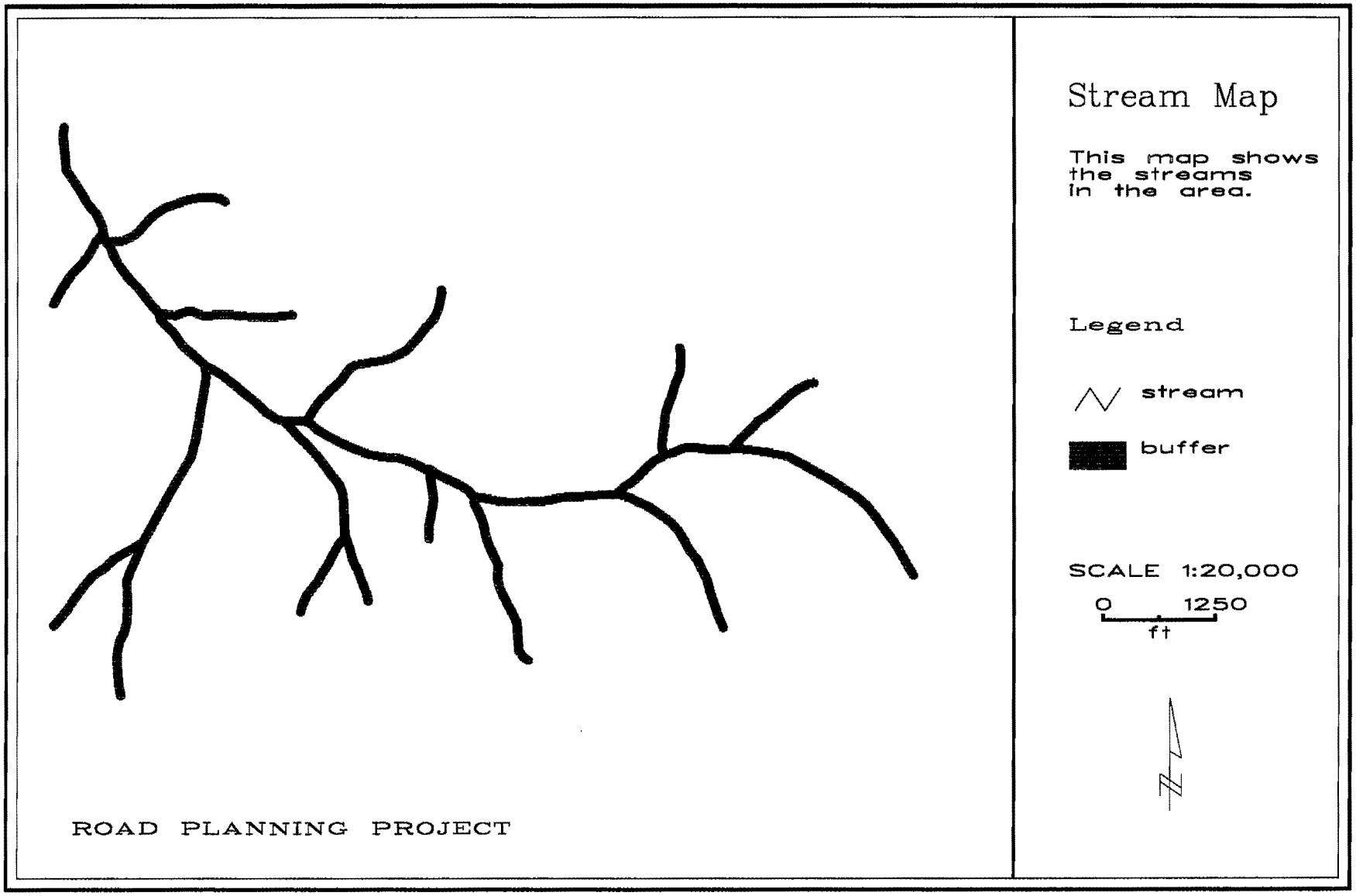
5.1.2.5 Streams

The Stream data are in the GEOHYDRO layer. The streams in the selected area are of the first and second order. The total length of streams is 34350 ft (10470 m). According to the Forest Practices Administrative Rules of Oregon (Oregon Department of Forestry, 1995) and the grid space, a 30 meter buffer was used in this study. The STREAM.MAP is shown in Map 4.

5.1.2.6 Slopes

Slopes in the selected area are very steep. From the logging point of view, we made a slope map which shows the slopes of the area. From the slope map we can see that most of the selected area's slope is over 35%, which means it is good for cable logging but difficult for selecting roads with less than 18% slope. The SLOPE.MAP is shown in Map 5.


Road Map
This map shows the existing roads in the area.
Legend
roads
SCALE 1:20,000
0 1250
ft
ROAD PLANNING PROJECT


Stream Map
This map shows the streams in the area.
Legend
stream
buffer
SCALE 1:20,000
0
1250
ft
ROAD PLANNING PROJECT

ROAD PLANNING PROJECT

Slope Map

This map shows the slopes of the area.

Legend

over 35%

less than 35%

SCALE 1:20,000

0 1250

ft

5.1.2.7 Road template and other data

The road template data, total volume in a landing, construction cost, variable cost and etc. are in the ASCII file TEMP.DAT. These data are listed in Table 4.

Table 4 Road template and Cost data

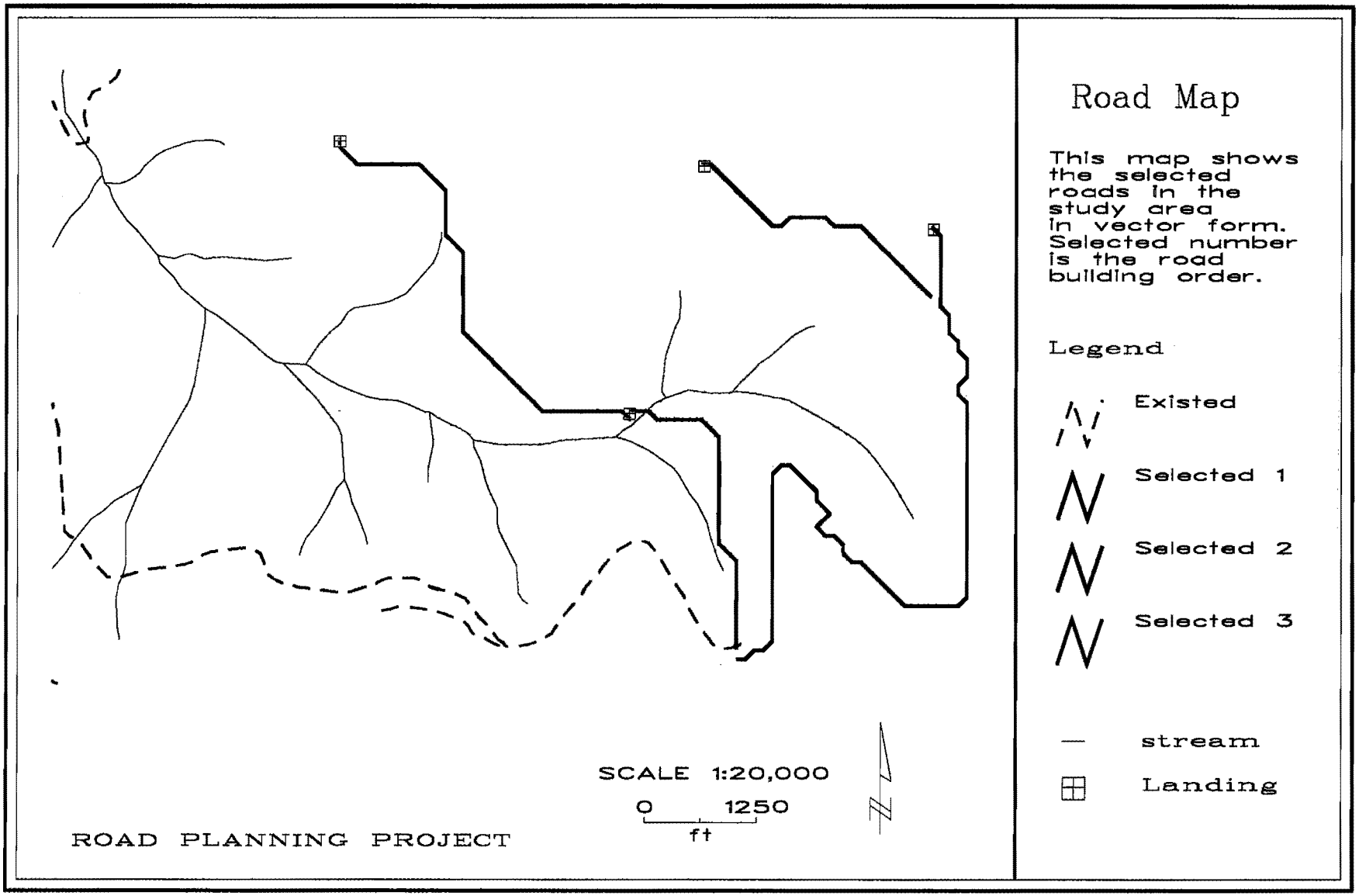
Running Width	12 ft	Surface Depth	8 ft
Ditch	3 ft	Turnouts	5 ft
Turnout Length	100 ft	CutSlope (.5:1)	2
FillSlope (1.5:1)	.67	Pipes Per mile	6
Pipes cost	$300/pipe	Excavation Cost (rock)	$2/mile
Excavation Cost (common)	$0.5/mile	Bridge Cost	$60000/mile
Clear Grub	$1500/acre	Seeding	$200/acre
Miscellaneous(survey_and_design) $10000/mile		Truck Cost	$51.34/h
Truck Load	25 ton	RockCode	6
Surfacing cost 1	$0.1/sy	Surfacing cost 2	$5/cy
Endhaul cost	$1/cy	Total volume	50,000 ton

5. 2 RESULTS

The results are shown in Maps 5, 6 and Table 5.

Table 5 Length and cost of the selected roads

Road	Length (m)	Build Order	Cost ($)
Existing	4346 (14255 ft) (2.7 mi)		
Selected 1	3058 (10,030 ft) (1.90 mi)	1	195,981
2	1082 (3549 ft) (0.67mi)	2	249,675
3	991 (3250 ft) (0.62 mi)	2	52,750
4	1651 (5415 ft) (1.03 mi)	3	82,833


Road Map
This map shows the selected roads in the study area in vector form. Selected number is the road building order.
Legend
Existed
Selected 1
Selected 2
Selected 3
stream
Landing
SCALE 1:20,000
0 1250
ft
ROAD PLANNING PROJECT

ROAD PLANNING PROJECT

Road Map

This map shows the selected roads in the study area in grid form.

Legend

Existing roads

Selected roads

SCALE 1:20,000

0 1250

ft

6 DISCUSSION AND CONCLUSION

6.1 EXPLANATIONS OF THE RESULTS

In the above example, I made three selections. In order to demonstrate this program, I assigned a large variable cost ($30,000 in the example) for each road section with a grade over 18%. This allows access to the selected landing but the selected road might be infeasible for truck performance. The first selection is one landing (selected 1 in Road.map). The selected road is 1.90 miles long and its total cost is $195,581. Some of the grades of the selected road are over 18%. In actual planning, infeasible roads are not allowed and we block the links by assigning them an very large cost such as $999999 per link. Interestingly in selection 1, when a variable cost larger than $30,000 per link was used to signal infeasibility, a feasible location with respect to grade was found, but the road location repeatedly crossed from side to side of the stream and was considered environmentally unacceptable. Modification of the algorithm to limit stream crossing or the use of higher costs for stream crossing (including environmental costs) should be considered in future development. In selection one, there are three links with over 18% grade, resulting in a $90,000 cost. The remaining cost of $105,581 for 1.9 mile road is $52,791/mile. For a total volume of 50,00 tons, the range for variable cost is from $8775 (grade 0) to $30626 (grade 17.8%). The excavation cost varies according to the ground slope. In this example it is about $3,000 to $20,000 in most cases. The surfacing, miscellaneous and pipe cost are fixed ($21,800/mile). Including clearing and grubbing and seeding cost, a common total cost range is from $40,000 to 75,000 using the data in this example. In the second selection (selected 2 in ROAD.MAP), we chose two landings. The 0.67 mile road ($249,675) has 5 links over 18% grade and crossed a stream one time resulting in a $210,000 cost. The remaining $49,675 is a reasonable cost. The 0.62 mile road ($57,250) has one link over 18% grade, the remaining $27,250 is also in the common range. The last selection has one landing. The

road selected is 1.03 miles with a $82832 cost. It also has one link with over 18% grade. From above analysis, we see that the method is feasible in selecting good routes according to the variable and construction cost.

Comparing the result from the PATHDISTANCE function (Appendix C 2) with the selection 1 made by ROAD.C , the results are almost the same. This is because the ASCII file VER_TABLE used in PATHDISTANCE is made according to the variable cost calculation in ROAD.C. In PATHDISTANCE function, only road slope is considered, no earthwork calculation. The advantage of the ROAD.C over the PATHDISTANCE function is that ROAD.C has considered the earthwork cost, as well as other costs such as clearing and grubbing, seeding , bridge, pipe, and surfacing cost. ROAD.C has also considered other road planning requirements such as soils and streams. All of these considerations can make the routes better than those made only by considering variable cost.

6.2 THE ADVANTAGES OF THIS ROAD PLANNING PROGRAM

The advantages of this method are as follows:

1. More GIS data are used. DEM, stand, soil, road and stream data are used in this study. Similar preliminary road planning studies in the past only used DEM (DTM) data.

2. Visualizations. Using the powerful graphics of ARC/INFO, DEM, Stand, soil, road, stream layers can be easily shown on the screen, color printer and plotter. It is very helpful when selecting a landing to use DEM and to have existing road and streams shown on screen.

3. Flexibility. In order to meet the specific needs of forest road

planning, a custom program is used to calculate the smallest cumulative cost from landings to roads. This program takes DEM, road, stream and soil data from GRID and produces two ASCII files with least cumulative costs and directions.

4. Efficiency. The program runs on a UNIX SUN workstation, so larger planning areas can be handled. ARC GRID can handle 100,000 cells in memory. If a 30 meter DEM is used, 100,000 cells is 22,197 acres (8987 hectares) in size.

6.3 DISADVANTAGES

6.3.1 Restrictions from ARC GRID

1. No GRID functions could be used to calculate the earthwork, Therefore, ROAD.C was developed.

2. It is difficult to record the length and grade of the selected road in GRID directly.

6.2.2 Problems from ROAD.C

1. The selected road might have a very short radius because the algorithm is based on cell by cell selection.

2. For earthwork calculation, only two points are used for ground slope calculation. These two points are 60 meters apart in shortest direction and 85.85 meters apart in diagonal direction. They are not accurate enough.

6.4 FURTHER STUDIES

This project is just a starting point for the computer modeling of forest road planning using ARC GRID . There is much room for improving and exploring. Some further studies and possible approaches are discussed bellow:

1. Logging vehicle requirements need to be considered, including considerations of horizontal and vertical alignment. For a logging vehicle the minimal design radius for horizontal curves is 45 feet and 50 feet for vertical curves (Kramer, 1993). For vehicle performance, the radius should be as large as possible. In order to get better horizontal and vertical alignment, one can use tabu strategy; that is, once a road section is selected, make tabu the two possible road directions which make a 45^0 angle with the selected one (Fig. 21). In this way, the roads have better horizontal and vertical alignment. This tabu might be more effective when using high spatial resolution DEMs (10 meter, for example). Tabu strategy can also be used in road sections with over 18% grades or multiple stream crossing.

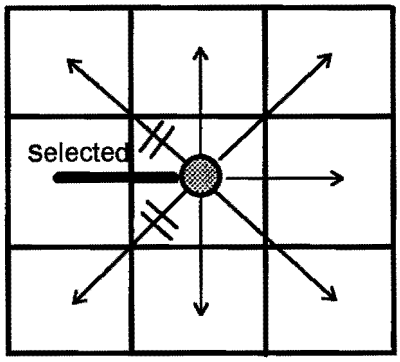
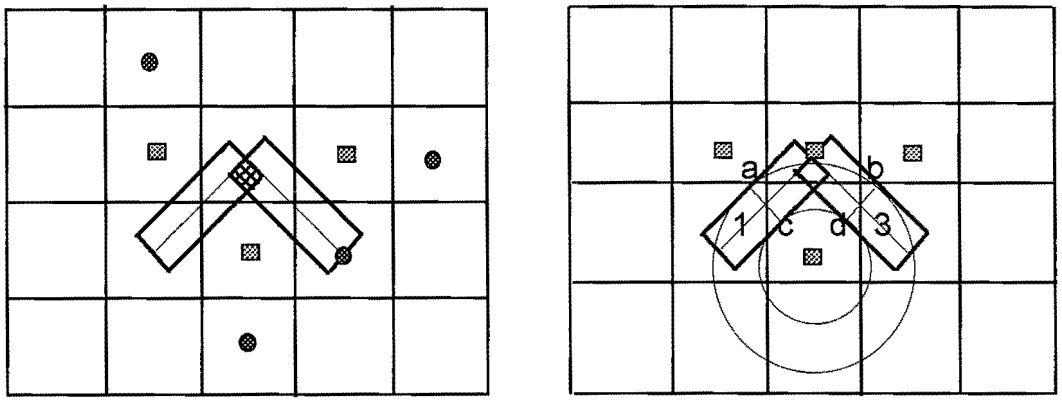


Fig. 21 Tabu directions.
// tabu

2. Spatial resolution needed for forest road planning. For forest road planning, the suitable spatial resolution is about 10 meters (Jaeger D. and Gero Becker, 1995) . USGS 30 meter DEMs are now available, and10 meter

DEMs are also available from SPOT Corporation for certain areas. There are no commercial DEMs with less than 10 meter resolution available now. One way to get more data based on 30 meter or 10 meter DEMS is to add more points according to the air photos taken in the same area. Too coarse resolution DEMs can not provide enough surface information, but too high resolution DEMs could be expensive in calculation time.

3. Earthwork calculations around horizontal curves and through vertical curves. One of the situations for earthwork calculation in ROAD.C is shown in Fig. 22 (a). The data points (dots) used in ROAD.C are 84.85 meters between each other. The data points (squares) are only 42.43 meters between each other; therefore, the accuracy of slope calculations will be high if these points have been used. The shaded area in Fig. 22 (a) indicates that the area has been calculated twice in earthwork calculations. Considering the horizontal curves, the earthwork calculation should be done as shown in Fig. 22 (b). That is, find the intersection points of the radius and the tangent lines (a, b, c, d), and calculate the earthwork in three parts (1, 3, and the part made by a, b, c and d).



a b

Fig. 22 Horizontal curve effect on earthwork calculations.

● data point used in ROAD.C for ground slope calculation
▨ data point should have been used in ROAD.C for ground slope calculations

Considering the effect of vertical curves in earthwork calculation, the distance from one cell to another should be adjusted (Fig. 23). In ROAD.C, after one section of the earthwork is calculated, the cell's earthwork is

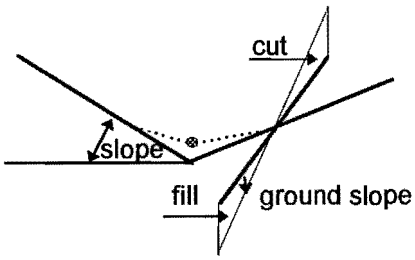


Fig. 23 Vertical effect on earthwork calculation.

calculated by multiplying the spacing (30 meter) of the cell for the shortest direction or 1.4142*30 meters for diagonal direction. Even for the shortest direction, the distance should be adjusted for the slope. Further adjustment is needed when considering the vertical curve.

4. Algorithmic modifications to provide feasible solutions for vehicle performance. The algorithm used in ROAD.C is to select routes from one cell to

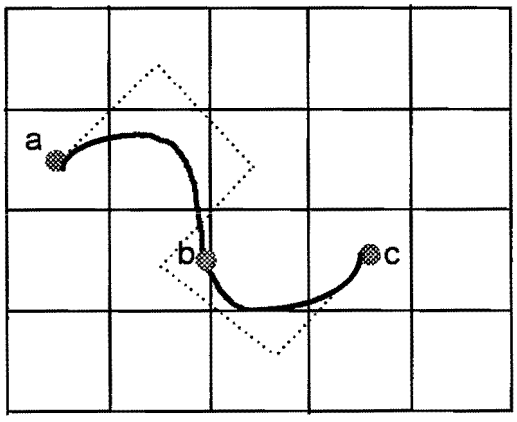


Fig. 24 Spline smoothing.

..... possible routes selected by ROAD.C

—— Spline smoothing line

its 8 neighboring cells. Therefore, one possible route from cell a to cell c might be shown as the dashed line in Fig. 24. This route is not good for vehicle performance because of the zigzag path, especially when the grid spacing is small. One alternative is to use a spline to smooth it. For spline line smoothing, the critical point is how to select points the spline must pass through. In the example in Fig. 24, the points should be less than the number of cells from a to c and more than two. So, three points (a, b, c) might be good for this example. The points should be selected according to slopes, soils, streams and other factors.

5. Develop alternative acceptable paths to provide choices for the forest engineer. This could be done by blocking the good path and doing the road selection again. To find the second best path, one way is to sequentially cut each link in the path and record the total cost of each potential second best path. At the end, find the path with smallest cost among them. For a long path with many links, this takes a long calculation time.

6. Solve the volume to landing problem. A program like PLANS might be useful in determining the volume for a landing. That is, if we know the working area of the cable system and the average volume per acre (or per hectare), we could know the volume for the landing.

7. Solve the problem of recording the length and grade of selected roads. One way is to make another program, this will take more running time and lose the advantages of using GRID.

8. Field test of the selected road. It is useful to go to the field to test whether the selected roads based on the 30 meter DEM can be located on a particular terrain.

9. Explore the usefulness of using the TIN module to calculate the ground slope. The TIN module uses irregular triangles to represent the surface (ESRI, 1992). It will be more accurate to calculate ground slope using the TIN model.

6.5 CONCLUSION

This study shows that ARC GRID can be used in preliminary planning of forest roads but with limitations. If I only use DEMs and consider only the slope factor, GRID can be used directly using PATHDISTANCE and COSTPATH functions. The vertical factor grid should be the DEM, and the vertical factor parameter should be customized ASCII table. The table should be built according to the hauling cost calculation. For considering both construction cost and hauling cost and other road planning requirements, a custom program is needed. Because it is easy to run a custom program using ARC Macro Language (AML) in ARC/INFO, the combination of a custom program with the functions in GRID can do the preliminary planning of forest roads which takes into consideration the DEM, stands, soils, streams and the existing road GIS data.

This program is a tool for assisting forest road planning engineers. It cannot replace the role of the forest road planning engineer. One of the shortcomings of this program is that the selected roads might have too sharp turns (too little radius) and zigzag paths.

For preliminary planning, the important thing is the usefulness of road routes, not the exact cost figure. Even if the route selected is not perfect from an engineering point of view, it still can provide good information for the forest road planning engineer. The routes selected by computer should be modified by the knowledge and experiences of the engineer. Using this program can

certainly reduce the road planner's hand work. As more GIS data are produced, for example, higher resolution DEMs and this program is further improved, this program will be more useful in the future.

LITERATURE CITED

Burke, D. 1974. Skyline logging profiles from a digital terrain model. In Proc. of 1974 Skyline logging Symposium. January 23-24, 1974. Univ. of Washington, Seattle, WA. pp. 52-55

Burrough, P. A. 1986. Principles of Geographical Information Systems for Land Resources Assessment. Clarendon Press. OXFORD

Congalton, Russell G., Kass Green and John Teply. 1994. Mapping old Growth Forests on National Forest and Park Lands in the Pacific Northwest from Remotely Sensed Data. *Photogrammitric Engineering & Remote Sensing,* Vol. 59, No. 4, April 1993, pp. 529-535.

Cullen, J. and P. Schiess. 1992. Integrated Computer Aided Timber Harvest Planning System. *Proceedings of the International Mountain Logging and Eighth Pacific Northwest Skyline Symposium*, Bellevue, Washington, 14-16 December 1992

Epstein, R., A. Weintraub, P. Sapunar, J. Sessions and J.B. Sessions. 1994. PLANEX Software for Operational Planning. *International Seminar on Forest Operations under Mountainous Conditions*, July 24-27, 1994, Harbin, P.R. of China

ESRI. 1992. Surface modeling with TIN. *ARC/INFO User's Guide*. Version 6. Environmental Systems Research Institute, Inc.

ESRI. 1992. Data Conversion. *ARC/INFO User's Guide*. Version 6. Environmental Systems Research Institute, Inc.

ESRI. 1992. Cell-based Modeling with GRID. *ARC/INFO User's Guide*. Version 6. Environmental Systems Research Institute, Inc.

ESRI. 1995. ARC GRID - Raster Geoprocessing Extensions for ARC/INFO ID. *ESRI White Paper Series*, January 1995.

Hoffer, Roger M. 1994. Challenges in Developing and Applying Remote Sensing to Ecosystem Management. *Remote sensing and GIS in ecosystemmanagement:* Edited by V. Alaric Sample. Island Press.

Howard, A.F. 1989. Highlead yarding productivity and cost in coastal British Columbia: Predicted vs. Actual. Western Journal of Applied Forestry. Vol. 4, No. 3, pp. 98-101

Jaeger D. and Gero Becker. 1995. Computer supported, interactive detailed planning of opening-up and harvesting operations on the base of Digital Terrain Models (DTM). Proceedings of the Technical Sessions of Subject Group 3.06, IUFRO XX World Congress, Tamper, Finland. August 6-12, 1995. pp. 25-32.

Kramer, B.W. 1993. A Road Design Process for Low Volume Recreation and Resource Development Roads. Oregon State University, Corvallis

Lasdon, L. S., A. D. Waren, A. Jain and M. Ratner. 1978. Design and testing of a centralized Reduced Gradient code for nonlinear programming. *ACM Trans. Math. Software 4*, 34-50.

Lewis, S. 1990. Use of geographical information systems in transportation modeling. ITE Journal, March 1990 pp. 34- 38

Liu, K. and J.Sessions. 1993. Preliminary planning of road systems using Digital terrain models. Journal of Forest Engineering, January, 1993.

Lutansky,Jr., L. 1989. GIS in Alaska's highway analysis system. Presented at the AASHTO Geographical Information Systems for Transportation Symposium in Orlando, Fla., February 1989

McGaughey, R. and R. Twito. 1988. large-area timber harvest planning with digital terrain models. In proc. of International Mountain Logging and Pacific Northwest Skyline Symposium. Dec. 12-16, 1988, Portland, Oregon State Univ. Corvallis, OR. pp.87-92

Nicholson, A. J., D. G. Elma and A. Williman. 1976. A variational approach to optimal route location. *Highway Engineering*, **23**, 22-25

Nieuwenhuis, M.A. 1987. The use of a geographic information system in computer-assisted forest road network analysis. In proc. of 10th Annual COFE Meeting, Syracuse, NY. August 3-6, 1987. pp. 177-184

Oregon Department of Forestry, 1995. Forest Practices Administrative Rules, January, 1995.

O'neill, E.A. 1991. Developing optimal transportation analysis zones using GIS. ITE Journal. Dec. 1991. pp. 33-36

Rauscher, H.M. and T.M. Cooney. 1986. Using expert-system technology in a forestry application: the CHAMPS experience. J. of Forestry 84(3):14-17

Reutebuch, S. 1988. ROUTES: A computer program for preliminary route location. USDA Gen. Tech. Publication. PNW-GTR-216, Portand, OR. 18p.

Schittkowski K. 1985. NLPQL: A FORTRAN subroutine for solving constrained nonlinear problems. Oper. Res. Ann. **5**, 485-500

Sessions, J. 1987. A heuristic algorithm for the solution of the variable and fixed cost transportation problem. In. Proc. of 1987 Symposium on system Analysis in Forest Resources, Univ.of Georgia, Athens. PP. 324-336

Sessions, J. and J.B. Sessions. 1994. Secheduling and Network Analysis Program (SNAP II) User's Guide Ver. 2.071.

Simkowitz, H.J. 1988. Transportation applications of geographical information systems. Computer, Environment and Urban Systems. pp. 253-271

Swanson, F.J, G.E. Grant and G. W. Lienkaemper. 1987. Field Trip Guide to the H. J. Andrews Experimental Forest. International Symposium on Erosion and Sedimentation in the Pacific Rim. August 2, 1987.

Tucek, J. 1994. Using the GIS Environment for Opening-up of Forest. *International Seminar on Forest Operations under Mountainous Conditions*, July 24-27, 1994, Harbin, P.R. of China

Twito, R.H. and R.W. Mifflin. 1982. Computer assisted evaluation of skyline thinning opportunities. In Proc. of 7306, The small tree resource: a materials handling challenge; April 19-21, 1982, Portland, OR. Madison, WI: Forest Products Research Society. pp. 73-79

Walton, D. J. and D. S. Meek. 1989. Computer-aided design for horizontal alignment. Journal of Transportation Eng. 115, No. 4, 411-424

Young, G. and D. Lemkow. 1976. Digital terrain simulators and their application to forest development planning. In proc.of 1976 Skyline Logging Symposium.Dec. 8-10, Vancouver, B.C. Univ. of B.C. Press:81-89

APPENDIX A **AML** (ARC MACRO LANGUAGE)

1 DO_ROADS.AML

```
/*********************************************************
*                                                        *
/*          Forest road location using GRID AML           *
/*                        by                              *
/*                   Shenglin Xu                          *
/* ------------------------------------------------------ *
/* starting June              Last modified   9/18/95     *
/*********************************************************
/* Run in GRID first

disp 9999 3
/* set full screen display
/* show coverages ------------------------
 mape roads_gs
 gridshades roads_gs
 &pause

&sv counter = 1
&sv name = 1
&sv flag = y
&run aml/kill_path.aml
&type Welcome to Forest Road Location Program!
&do &while %flag% = y  /* set control point
&run aml/initial.aml

  /* select new landing -------------------------
  mape hjadem
  gridshades landings
  linecolor cyan
  arcs cont10
  linecolor magenta
  arcs roadsx0
  linecolor yellow
  arcs streamsx
  landing = selectpoint(hjadem, *)
  mape landing
  gridshades landing 3
  &pause
  gridshades roads_gs
  &pause
  q
  &run convert.aml
  grid
  opt_path = costpath(landing, earth_out, direc_out)
  /* save the intermideam results for final statistics
  copy opt_path path_g%name%

  /* show the result ----------------------
```

```
  clear
  mape opt_path
  gridshades opt_path
  linecolor cyan
  arcs cont10
  linecolor magenta
  arcs roadsx0
  linecolor yellow
  arcs streamsx
  cellvalue earth_out *

  /* add the new road to existing road
  /* (including previous selected roads
  /* for next selecting.
    opt_pathn = isnull(opt_path)
    roads_gsn = isnull(roads_gs)
  &if [exists roads_gs -grid] &then kill roads_gs
  roadsxn = roads_gsn + opt_pathn
  roads_gs = select(roadsxn, 'value = 1')   /* new
  &if [exists roadsx0 -line] &then kill roadsx0
  roadsx0 = gridline(roads_gs)
/*   copy roadsx0 path_v%name%
  &s name = %name% + 1
   /* ask if next locating is neccessary ----------------
  &type Engter y (yes) to locate or n (no) to quit.
  &sv flag = [response 'y or n (yes/no) ? ']
&end                                  /* Locationn done

/* program back to original status for next use ----------
/* suffix _f means final result
copy roads_gs roads_g_f
copy roadsx0 roadsx_f
&if [exists roads_gs -grid] &then kill roads_gs
&if [exists roadsx0 -line] &then kill roadsx0
/* back to original status. File end with x is original.
copy roads_gx roads_gs
copy roadsx roadsx0
clear
linecolor red
arcs roadsx_f
linecolor cyan
arcs cont10
linecolor yellow
arcs streamsx

/* The following part convert saved selected roads to
/* vector coverages
&run path_g2vec.aml
&return
```

2. CONVERT.AML

```
/* Convert AML          Shenglin Xu  8/25/95
/* This AML converts ASCII file to grid and vice versa
/* and call ROAD.C

rm /data/pspace/mx/roads_a
gridASCII roads_gs roads_a
&type old ASCII file has been removed!
&type Now is runing ROAD.C program to produce ASCII files.
&system /data/pspace/mx/ROAD.C
&type ASCII files have been done. Now transfering ASCII file to GRID.
ASCIIgrid ear_out earth_out float
ASCIIgrid dir_out direc_out
&type GRIDs earth_out and direc_out have been done!
```

---

3. POLY2GRID.AML

```
/*********************************************************
/* POLY2GRID AML        Shenglin Xu      June 95          *
/* This AML converts vector coverage into grid            *
/*********************************************************
/* When using linegrid to convert coverage to grid, such as  *
/* roads_g = linegrid(roadsx, roadsx-id, #, #, 30, zero)     *
/* the area of roads_g will become smaller because of the length of  *
/* the roads. One way to solve this problem is to identity roads to  *
/* polygon coverage like soils, or stands. Then covert polygon       *
/* coverage to grid                                          *
/*********************************************************

identity soilsx streams_buf soils_str ploy # join
identity soilsx roadsx soils_rd poly # join
streams_g = polygrid(soil-str, soils_str-id, #, #, 30)
roads_g = polygrid(soils_rd,  soils_rd-id, #, #, 30)
soils_g = polygrid(soilsx, dcode, #, #, 30)
tands_g = polygrid(standsx, feature_code, #, #, 30)
```

4. LANDINGS.AML

```
/* Landing selecting  AML  by Shenglin Xu  7/25/95

&if [exists landings -grid] &then kill landings
if (slope_gp == 4 & soils_g == 4 & stands_g <= 4) landings = 1
else if (slope_gp == 4 & soils_g == 3 & stands_g <= 4) landings = 2
else if (slope_gp == 4 & soils_g >= 5 & stands_g <= 4) landings = 3
else if (slope_gp == 2 & soils_g == 4 & stands_g <= 4) landings = 4
else if (slope_gp == 2 & soils_g == 3 & stands_g <= 4) landings = 5
else if (slope_gp == 2 & soils_g >= 5 & stands_g <= 4) landings = 6
else if (slope_gp == 3 & soils_g == 4 & stands_g <= 4) landings = 7
else if (slope_gp == 3 & soils_g == 3 & stands_g <= 4) landings = 8
else if (slope_gp == 3 & soils_g >= 5 & stands_g <= 4) landings = 9
else if (slope_gp == 5 & soils_g == 4 & stands_g <= 4) landings = 10
else if (slope_gp == 5 & soils_g == 3 & stands_g <= 4) landings = 11
else if (slope_gp == 5 & soils_g >= 5 & stands_g <= 4) landings = 12
else if (slope_gp == 6 & soils_g == 4 & stands_g <= 4) landings = 13
else if (slope_gp == 6 & soils_g == 3 & stands_g <= 4) landings = 14
else if (slope_gp == 6 & soils_g >= 5 & stands_g <= 4) landings = 15
else if (slope_gp == 7 & soils_g == 4 & stands_g <= 4) landings = 19
else if (slope_gp == 7 & soils_g == 3 & stands_g <= 4) landings = 20
else if (slope_gp == 7 & soils_g >= 5 & stands_g <= 4) landings = 21
else if (slope_gp == 8 & soils_g == 4 & stands_g <= 4) landings = 22
else if (slope_gp == 8 & soils_g == 3 & stands_g <= 4) landings = 23
else if (slope_gp == 8 & soils_g >= 5 & stands_g <= 4) landings = 24
else if (slope_gp == 9 & soils_g == 4 & stands_g <= 4) landings = 25
else if (slope_gp == 9 & soils_g == 3 & stands_g <= 4) landings = 26
else if (slope_gp == 9 & soils_g >= 5 & stands_g <= 4) landings = 27
else if (slope_gp == 10 & soils_g == 4 & stands_g <= 4) landings = 28
else if (slope_gp == 10 & soils_g == 3 & stands_g <= 4) landings = 29
else if (slope_gp == 10 & soils_g >= 5 & stands_g <= 4) landings = 30
else if (slope_gp == 11 & soils_g == 4 & stands_g <= 4) landings = 31
else if (slope_gp == 11 & soils_g == 3 & stands_g <= 4) landings = 32
else if (slope_gp == 11 & soils_g >= 5 & stands_g <= 4) landings = 33
else if (slope_gp == 12 & soils_g == 4 & stands_g <= 4) landings = 34
else if (slope_gp == 12 & soils_g == 3 & stands_g <= 4) landings = 35
else if (slope_gp == 12 & soils_g >= 5 & stands_g <= 4) landings = 36
else if (slope_gp == 13 & soils_g == 4 & stands_g <= 4) landings = 37
else if (slope_gp == 13 & soils_g == 3 & stands_g <= 4) landings = 38
else if (slope_gp == 13 & soils_g >= 5 & stands_g <= 4) landings = 39
else if (slope_gp == 14 & soils_g == 4 & stands_g <= 4) landings = 40
else if (slope_gp == 14 & soils_g == 3 & stands_g <= 4) landings = 41
else if (slope_gp == 14 & soils_g >= 5 & stands_g <= 4) landings = 42

else if (slope_gp == 15 & soils_g == 4 & stands_g <= 4) landings = 16
else if (slope_gp == 15 & soils_g == 3 & stands_g <= 4) landings = 17
else if (slope_gp == 15 & soils_g >= 5 & stands_g <= 4) landings = 18

endif
disp 9999 1
mape landings
```

```
gridshades landings
linecolor yellow
arcs cont10
linecolor white
arcs roadsx0
&return
```

---

5. STANDS_MAP.AML

```
/*  Stands map AML     by Shenglin Xu    7/30/95

clear
kill stands.map
pagesize 13.5 9
map stands.map

/*    This section specifies neatlines and borders for the main map
linesymbol 9
box 1 1 13 8.9
linesymbol 1
box 1.1 1.1 12.9 8.8
linesymbol 9
line 10 1.1 10 8.8

/*  This section draws the map titles
textsymbol 41
move 10.5 8
text 'Stands Map'

/*  This section draws the descriptive text block.
textsymb 33
move 10.5 7.5
textfile stand_text

/*  Here, the legend titles are positioned and drawn.
textsymbol 89
move 10.5 6
text 'Legend'

/*  Perform map to page transformations for the main map
/*  and draw its polygons.
maplim 1.1 1.1 10 8.5
  mapscale AUTOMATIC
```

```
  mapunits feet
  mappos ll 1 2.3
  mapex standsx
  polys standsx

reselect standsx poly stand_code = 1
polygonshades standsx 1
clearselect
reselect standsx poly stand_code = 4
polygonshades standsx 4
clearselect
reselect standsx poly stand_code = 6
polygonshades standsx 6
clearselect
reselect standsx poly stand_code = 7
polygonshades standsx 7
clearselect

/* Here, the legend keys are drawn

textsym 33
keypos 10.5 5.5
keybox .5 .25
keysep .15 .25
keyshade stand.key

/*  This section draws the name, student # and project #

textsymbol 37
move 1.6, 1.4
text 'ROAD LOCATION PROJECT '

map end
```

6. SOILS_MAP.AML

```
/*  Streams map AML     by Shenglin Xu    7/30/95
disp 9999 1
kill soils.map
pagesize 13.6 9
map soils.map

/*    This section specifies neatlines and borders for the main
/*    maps
linesymbol 9
box 1 1 13 8.9
linesymbol 1
box 1.1 1.1 12.9 8.8
linesymbol 9
line 10 1.1 10 8.8
/*  This section draws the map titles

textsymbol 41
move 10.5 8
text 'Soils Map'

/*  This section draws the descriptive text block.
textsymb 33
move 10.5 7.5
textfile soil_text

/*  Here, the legend titles are positioned and drawn.
textsymbol 89
move 10.5 6
text 'Legend'

/*  Perform map to page transformations for the main map
/*  and draw its polygons.
maplim 1.1 1.1 10 8.5
  mapscale AUTOMATIC
  mapunits feet
  mappos ll 1 2.3
  mapex soilsx
  polys soilsx

reselect soilsx poly class = 1
polygonshades soilsx 1
clearselect
reselect soilsx poly class = 2
polygonshades soilsx 2
clearselect
reselect soilsx poly class = 3
polygonshades soilsx 3
clearselect
reselect soilsx poly class = 4
polygonshades soilsx 4
```

```
clearselect

/* Here, the legend keys are drawn
textsym 33
keypos 10.5 5.5
keybox .5 .25
keysep .15 .25
keyshade soil.key

/* Here, the scale is drawn
markersize 1
barscale 500 ft # *

/* Here, northstar is drawn
markerset municipal.mrk
markersymbol 328
markersize 1
marker *

/*  This section draws the name, student # and project #
textsymbol 37
move 1.6, 1.4
text 'ROAD LOCATION PROJECT '
map end  /* end of soil.map AML
```

7. STREAMS_MAP.AML

```
/*  Streams map AML     by Shenglin Xu    7/30/95

clear
kill streams.map
pagesize 13.5 9
map streams.map

/*    This section specifies neatlines and borders for the main map
linesymbol 9
box 1 1 13 8.9
linesymbol 1
box 1.1 1.1 12.9 8.8
linesymbol 9
line 10 1.1 10 8.8

/*  This section draws the map titles
textsymbol 41
move 10.5 8
text 'Streams Map'

/*  This section draws the descriptive text block.
textsymb 33
move 10.5 7.5
textfile stream_text

/*  Here, the legend titles are positioned and drawn.
textsymbol 89
move 10.5 6
text 'Legend'

/*  Perform map to page transformations for the main map
/*  and draw its polygons.
maplim 1.1 1.1 10 8.5
  mapscale AUTOMATIC
  mapunits feet
  mappos ll 1 2.3
  mapex streams_buf
  polys streams_buf
reselect streams_buf poly inside = 100
polygonshades streams_buf 2
clearselect

  arcs streamsx
reselect streamsx line length > 0
linecolor white
arcs streamsx
clearselect

/* Here, the legend keys are drawn
textsym 33
```

```
keypos 10.5 5.5
keyline stream.key nobox

keysep .15 .25
keybox .5 .25
keyshade stream_buf.key

/* Here, the scale is drawn
markersize 1
barscale 500 ft # *

/* Here, northstar is drawn
markerset municipal.mrk
markersymbol 328
markersize 1
marker *

/*  This section draws the name, student # and project #
textsymbol 37
move 1.6, 1.4
text 'ROAD LOCATION PROJECT '

map end
```

8. ROADS_MAP.AML

```
/*  Road map AML     by Shenglin Xu    7/30/95

clear
disp 9999 3
pagesize 13.5 9
map roads.map

/*    This section specifies neatlines and borders for the main map
linesymbol 9
box 1 1 13 8.9
linesymbol 1
box 1.1 1.1 12.9 8.8
linesymbol 9
/line 10 1.1 10 8.8

/*  This section draws the map titles
textsymbol 41
move 10.5 8
text 'Selected Area'

/*  This section draws the descriptive text block.
textsymb 33
move 10.5 7.5
textfile road_text

/*  Here, the legend titles are positioned and drawn.
textsymbol 89
move 10.5 6
text 'Legend'

/*  Perform map to page transformations for the main map
/*  and draw its polygons.
maplim 1.1 1.1 10 8.5
  mapscale AUTOMATIC
  mapunits feet
  mappos ll 1 2.3
  mapex roadsx
reselect roadsx line length > 0
linecolor red
arcs roadsx

/* Here, the legend keys are drawn
textsym 33
keypos 10.5 5.5
keyline road.key nobox

/* Here, the scale is drawn
markersize 1
barscale 500 ft # *
```

```
/* Here, northstar is drawn

markerset municipal.mrk
markersymbol 328
markersize 1
marker *


/*  This section draws the name, student # and project #

textsymbol 37
move 1.6, 1.4
text 'ROAD LOCATION PROJECT '

map end
```

---

9. SHOW_MAP.AML

```
/* This AML shows Maps in ARCPLOT

disp 9999 3
mape standsx
map first.map
&pause
clear
map stands.map
&pause
clear
map soils.map
&pause
clear
map streams.map
&pause
clear
map roads.map
clear
map wait.map
&pause
&return
```

10. INITIAL.AML

```
/* Initializing  AML  by Shenglin Xu  7/27/95
/* delete existing grids , because GRID can create a grid
/* if a grid with same name already exists.

&if [exists road_cost -grid] &then kill road_cost
&if [exists opt_path -grid]  &then kill opt_path
&if [exists road_back -grid] &then kill road_back
&if [exists road_allo -grid] &then kill road_allo
&if [exists af_road_s -grid] &then kill af_road_s
&if [exists roadsxn -grid]   &then kill roadsxn
&if [exists roads_gsn -grid] &then kill roads_gsn
&if [exists opt_pathn -grid] &then kill opt_pathn
&if [exists roads_g_f -grid] &then kill roads_g_f
&if [exists roadsx_f -line] &then kill roadsx_f
&if [exists landing -grid]  &then kill landing
&return
```

---

11. BYSLOPE.AML

```
/* forest road location AML  by Shenglin Xu  8/15/95
/* This AML locates forest road considering the slope, soils,  and stands.
/* Slope in a cell is the max average of its neighboring cells

disp 9999 3
     /* set full screen display

/* show coverages --------------------------
  mape roads_gs
  gridshades roads_gs
  &pause
  gridshades slope_g35
  &pause

&sv flag = x
&type Welcome to Forest Road Location Program!
&do &while %flag% <> n   /* set control point
&run /aml/initial.aml

   /* select new landing ----------------------
   mape hjadem
   gridshades landings
   linecolor cyan
   arcs cont10
   linecolor magenta
   arcs roadsx0
```

```
  landing = selectpoint(hjadem, *)
  mape landing
  gridshades landing 3
  &pause
  gridshades roads_gs
  &pause

  /* find the least cost path ------------------
   road_cost = costdistance(roads_gs, af_road_s, road_back, road_allo)
  opt_path = costpath(landing, earth_out, back_out)       /* least cost path

  /* show the result ---------------------------
  clear
  mape opt_path
  gridshades opt_path
  linecolor cyan
  arcs cont10
  linecolor magenta
  arcs roadsx0
  &pause

  /* add the new road to existing road (including previous selected roads
  /* for next selecting.
  roads_gsn = isnull(roads_gs)            /* NODATA must be removed for adding
  opt_pathn = isnull(opt_path)            /* new road to existing roads
  &if [exists roads_gs -grid] &then kill roads_gs
  roadsxn = roads_gsn + opt_pathn
  roads_gs = select(roadsxn, 'value = 1')   /* new
  &if [exists roadsx0 -line] &then kill roadsx0
  roadsx0 = gridline(roads_gs)

  /* ask if next locating is neccessary ----------------
  &type Engter y (yes) to locate or n (no) to quit.
  &sv flag = [response 'y or n (yes/no) ? ']
&end                                /* Locationn done

/* program back to original status for next use ----------
copy roads_gs roads_g_f                 /* suffix _f means final result
copy roadsx0 roadsx_f
&if [exists roadsx0 -line] &then kill roadsx0
&if [exists roads_gs -grid] &then kill roads_gs
copy roadsx roadsx0                     /* back to original status. File end
copy roads_gx roads_gs                  /* with x is original.

&return
```

12. COST.AML

```
/* Cost Value AML          Shenglin Xu   8/11/95
/* This AML produces the cost grid for COSTDISTANCE function in BYSLOPE.AML

if (slope_gp <= 10 & stands_g <= 4 & soils_g == 4 ~
    & streams_g <> 1) af_road_s = 1
 else if (slope_gp <= 10 & stands_g <= 4 & soils_g == 3 ~
    & streams_g <> 1) af_road_s = 2
 else if (slope_gp <= 10 & stands_g <= 4 & soils_g >= 4 ~
    & streams_g <> 1) af_road_s = 4
 else if (slope_gp <= 10 & stands_g <= 4 & soils_g == 2 ~
    & streams_g <> 1) af_road_s = 10

 else if (slope_gp >= 10 & slope_gp <= 18 & stands_g <= 4 & soils_g == 3 ~
    & streams_g <> 1) af_road_s = 14
 else if (slope_gp >= 10 & slope_gp <= 18 & stands_g <= 4 & soils_g == 4 ~
    & streams_g <> 1) af_road_s = 16
 else if (slope_gp >= 10 & slope_gp <= 18 & stands_g <= 4 & soils_g >= 3 ~
    & streams_g <> 1) af_road_s = 18
 else if (slope_gp >= 10 & slope_gp <= 18 & stands_g <= 4 & soils_g == 2 ~
    & streams_g <> 1) af_road_s = 24

 else if (slope_gp >= 18 & slope_gp <= 25 & stands_g <= 4 & soils_g == 3 ~
    & streams_g <> 1) af_road_s = 50
 else if (slope_gp >= 18 & slope_gp <= 25 & stands_g <= 4 & soils_g == 4 ~
    & streams_g <> 1) af_road_s = 55
 else if (slope_gp >= 18 & slope_gp <= 25 & stands_g <= 4 & soils_g >= 3 ~
    & streams_g <> 1) af_road_s = 60
 else if (slope_gp >= 18 & slope_gp <= 25 & stands_g <= 4 & soils_g == 2 ~
    & streams_g <> 1) af_road_s = 65

 else if (slope_gp >= 26 & slope_gp <= 35 & stands_g <= 4 & soils_g == 3 ~
    & streams_g <> 1) af_road_s = 100
 else if (slope_gp >= 26 & slope_gp <= 35 & stands_g <= 4 & soils_g == 4 ~
    & streams_g <> 1) af_road_s = 110
 else if (slope_gp >= 26 & slope_gp <= 35 & stands_g <= 4 & soils_g >= 3 ~
    & streams_g <> 1) af_road_s = 120
 else if (slope_gp >= 26 & slope_gp <= 35 & stands_g <= 4 & soils_g == 2 ~
    & streams_g <> 1) af_road_s = 130

 else if (slope_gp >= 36 & slope_gp <= 60 & stands_g <= 4 & soils_g == 3 ~
    & streams_g <> 1) af_road_s = 200
 else if (slope_gp >= 36 & slope_gp <= 60 & stands_g <= 4 & soils_g == 4 ~
    & streams_g <> 1) af_road_s = 250
 else if (slope_gp >= 36 & slope_gp <= 60 & stands_g <= 4 & soils_g >= 3 ~
    & streams_g <> 1) af_road_s = 300
 else if (slope_gp >= 36 & slope_gp <= 60 & stands_g <= 4 & soils_g == 2 ~
    & streams_g <> 1) af_road_s = 350

 else if (slope_gp >= 61 & stands_g <= 4 & soils_g == 3 ~
    & streams_g <> 1) af_road_s = 500
```

```
else if (slope_gp >= 61 & stands_g <= 4 & soils_g == 4 ~
    & streams_g <> 1) af_road_s = 550
else if (slope_gp >= 61 & stands_g <= 4 & soils_g >= 3 ~
    & streams_g <> 1) af_road_s = 600
else if (slope_gp >= 61 & stands_g <= 4 & soils_g == 2 ~
    & streams_g <> 1) af_road_s = 700

else af_road_s = 9999
endif
&return
```

---

13. PATHDISTANCE.AML

```
/*****************************************************************
/*                  Forest road location using GRID AML          *
/*                                by                             *
/*                            Shenglin Xu                        *
/* ------------------------------------------------------------- *
/* starting June 1995          Last modified  8/18/95            *
/*****************************************************************
/*  This AML locates forest road using the PATHDISTANCE function in grid,  *
/*   only vertical factor was considered                          *
/*****************************************************************
disp 9999 3
/* set full screen display

/* show coverages ---------------------------
  mape roads_gs
  gridshades roads_gs
  &pause
  gridshades slope_g35
  &pause

&sv flag = x
&type Welcome to Forest Road Location Program!
&do &while %flag% <> n   /* set control point
&run aml/initial.aml

   /* select new landing ----------------------
   mape hjadem
   gridshades landings
   linecolor cyan
   arcs cont10
   linecolor magenta
   arcs roadsx0
   linecolor yellow
   arcs streamsx
```

```
  linecolor magenta

  landing = selectpoint(hjadem, *)
  mape landing
  gridshades landing 3
  &pause
  gridshades roads_gs
  &pause

  path_dist = pathdistance(roads_gs, #, #, #, #,  hjadem, "TABLE ver_table", pd_back)
 opt_path = costpath(landing, path_dist, pd_back)       /* least cost path

  /* show the result --------------------------
  clear
  mape opt_path
  gridshades opt_path
  linecolor cyan
  arcs cont10
  linecolor magenta
  arcs roadsx0
  cellvalue earth_out *
  &pause

  /* add the new road to existing road (including previous selected roads
  /* for next selecting.
  roads_gsn = isnull(roads_gs)              /* NODATA must be removed for adding
  opt_pathn = isnull(opt_path)              /* new road to existing roads
  &if [exists roads_gs -grid] &then kill roads_gs
  roadsxn = roads_gsn + opt_pathn
  roads_gs = select(roadsxn, 'value = 1')    /* new
  &if [exists roadsx0 -line] &then kill roadsx0
  roadsx0 = gridline(roads_gs)

  /* ask if next locating is neccessary ----------------
  &type Engter y (yes) to locate or n (no) to quit.
  &sv flag = [response 'y or n (yes/no) ? ']
&end                                /* Locationn done

/* program back to original status for next use ----------
copy roads_gs roads_g_f                  /* suffix _f means final result
copy roadsx0 roadsx_f
&if [exists roadsx0 -line] &then kill roadsx0
&if [exists roads_gs -grid] &then kill roads_gs
copy roadsx roadsx0                     /* back to original status. File end
copy roads_gx roads_gs                   /* with x is original.
clear
linecolor cyan
arcs cont10
linecolor magenta
arcs roadsx_f
&return
```

14 BOX.AML

```
/* Make box AML      Shenglin Xu   July 1995
/* This AML make box for clip selected area

arcedit
drawenv arc
backenv arc
backcover /data/cascade/hja/soils 3
create box  /data/cascade/hja/soils
draw
/* ef editfeature
ef arc
add
ef label
add
save
q
build box
clip  /data/cascade/hja/soils box soilsx

/* The following part is needed when info files are separeted.
relate restore /data/cascade/hja/info!arc!soil.rel
relate add
dat
/data/cascade/hja/info!arc!soil.data
info
soil_unit
ordered
rw
des
/data/cascade/hja/info!arc!soil.desc
info
map_unit
ordered
rw
relate save soils.rel
ap
mape soilsx
arcs /data/cascade/hja/soils
q
des box
ap
```

15. PATH_G2VEC. AML

```
/* Path_g2vec AML       Shenglin Xu    9/24/95
/* This AML converts grids path_g to vector path_vec
/* This AML runs in GRID

  &do i := 1 &to 15
     &if [exists path_vet%i% -line] &then kill path_vet%i%
        &if [exists path_g%i% -grid] &then ~
          path_vet%i% = gridline(path_g%i%)

  &end
  &return
```

---

16. ROADS_F_G.AML

```
/*  Final Road map (grid)  AML     by Shenglin Xu    9/24/95
clear

clear
kill roads_f_g.map
pagesize 13.6 9
map roads_f_g.map

/*    This section specifies neatlines and borders for the main map
linesymbol 9
box 1 1 13 8.9
linesymbol 1
box 1.1 1.1 12.9 8.8
linesymbol 9
line 10 1.1 10 8.8

/*  This section draws the map titles
textsymbol 41
move 10.5 8
text 'Road Map'

/*  This section draws the descriptive text block.
textsymb 33
move 10.5 7.5
textfile map_f_g.txt

/*  Here, the legend titles are positioned and drawn.
textsymbol 89
move 10.5 6
text 'Legend'

/* Perform map to page transformations for the main map
```

```
/*  and draw its polygons.
maplim 1.1 1.1 10 8.5
  mapscale 15000
  mapunits meter
  mappos ll 1 2.3
  mapex soilsx
gridshades roads_g_f 3
linesymbol 9
linecolor red
arcs roadsx
linesymbol 1
linecolor yellow
arcs cont10

/* Here, the legend keys are drawn
textsym 33
keypos 10.5 5.5
keyline road_f.key nobox

/* Here, the scale is drawn
linecolor green
move 10.5 3.8
text 'SCALE 1:20,000'
barscale 1250 ft # 11.2 3.4
/* Here, northstar is drawn

markerset municipal.mrk
markersymbol 328
markersize 1
marker 11.2 2.2

/*  This section draws the name, student # and project #

textsymbol 37
move 1.6, 1.4
text 'ROAD PLANNING PROJECT '

map end
```

17. ROADS_F_V.AML

```
/* Final Road map (vector)  AML by Shenglin Xu     9/24/95
clear
kill roads_f_v.map
pagesize 13.6 9
map roads_f_v.map
 /*    This section specifies neatlines and borders for the main map
linesymbol 9
box 1 1 13 8.9
linesymbol 1
box 1.1 1.1 12.9 8.8
linesymbol 9
line 10 1.1 10 8.8

/*  This section draws the map titles
textsymbol 41
move 10.5 8
text 'Road Map'

/*  This section draws the descriptive text block.
textsymb 33
move 10.5 7.5
textfile map_f_v.txt

/*  Here, the legend titles are positioned and drawn.
textsymbol 89
move 10.5 6
text 'Legend'

/* Perform map to page transformations for the main map
/* and draw its polygons.
maplim 1.1 1.1 10 8.5
  mapscale 15000
  mapunits meter
  mappos ll 1 2.3
  mapex soilsx
textsym 33
linesymbol 8
linecolor blue
arcs roadsx
keypos *
keyline road_f_v.key nobox

/* symbolscale .5
linesymbol 13
&do i := 1 &to 4
 linecolor %i%
 &if [exists path_vet%i% -line] &then arcs  path_vet%i%
 &if [exists path_vet%i% -line] &then keypos *
 &if [exists path_vet%i% -line] &then keyline road_f_v.key%i% nobox
&end
```

```
linesymbol 1
 linecolor 1
 &if [exists path_vet5 -line] &then arcs  path_vet5
 &if [exists path_vet5 -line] &then keypos *
 &if [exists path_vet5 -line] &then keyline road_f_v.key5 nobox
linecolor 2
 &if [exists path_vet6 -line] &then arcs  path_vet6
 &if [exists path_vet6 -line] &then keypos *
 &if [exists path_vet6 -line] &then keyline road_f_v.key6 nobox
linecolor 3
 &if [exists path_vet7 -line] &then arcs  path_vet7
 &if [exists path_vet7 -line] &then keypos *
 &if [exists path_vet7 -line] &then keyline road_f_v.key7 nobox
linecolor 4
 &if [exists path_vet8 -line] &then arcs  path_vet8
 &if [exists path_vet8 -line] &then keypos *
 &if [exists path_vet8 -line] &then keyline road_f_v.key8 nobox
 linesymbol 5
linecolor 1
 &if [exists path_vet9 -line] &then arcs  path_vet9
 &if [exists path_vet9 -line] &then keypos *
 &if [exists path_vet9 -line] &then keyline road_f_v.key9 nobox
linecolor 2
 &if [exists path_vet10 -line] &then arcs  path_vet10
 &if [exists path_vet10 -line] &then keypos *
 &if [exists path_vet10 -line] &then keyline road_f_v.key10 nobox

symbolscale 1
linesymbol 1
linecolor yellow
arcs cont10

/* Here, the scale is drawn
linecolor green
move 6.7 2
textsym 33
text 'SCALE 1:20,000'
barscale 1250 ft # 7.2 1.6

/* Here, northstar is drawn
markerset municipal.mrk
markersymbol 328
markersize 1
marker 8.6 2

/*  This section draws the name, student # and project #
textsymbol 37
move 1.6, 1.4
text 'ROAD PLANNING PROJECT '

map end
```

18. PRINT_MAP.AML

```
/************************************************
/* print_map AML        Shenglin Xu   9/21/95    *
/************************************************
/* printer could be fsl217pscolor (no charge) or *
/* calcomp color (charge)                        *
/* in ARC prompt, run SEPARATOR to create size.par *
/************************************************

postscript soils.map soils.eps .75 size96.par
postscript stands.map stands.eps .75 size96.par
postscript streams.map streams.eps .75 size96.par
postscript roads.map roads.eps .75 size96.par
postscript roads_f_g.map roads_f_g.eps .75 size96.par
postscript roads_f_v.map roads_f_v.eps .75 size96.par
postscript slope.map slope.eps .75 size96.par
lpr -Pfsl217pscolor stands.eps
lpr -Pfsl217pscolor soils.eps
lpr -Pfsl217pscolor streams.eps
lpr -Pfsl217pscolor roads.eps
lpr -Pfsl217pscolor roads_f_g.eps
lpr -Pfsl217pscolor roads_f_v.eps
lpr -Pfsl217pscolor slope.eps


&return
```

APPENDIX B C CODE

1. ROAD.C

```
/*****************************************************
/*                        Road.c                     *
/*                          by                       *
/*                      Shenglin Xu                  *
/*****************************************************
 * Starting: 8/1/95        Last Modified:   1/31/96   *
 *****************************************************
 *     This program reads four gridascii files (DEM, roads, soil   *
 * and streams ) from ARC/INFO. After calculation, two ascii  *
 * files are made by this program: one is earth_ouf file, which   *
 * assign ecah cell the least cost to a road cell; the other is   *
 * direc_out, which gives the direction of each cell to reach a   *
 * road cell. These files will be transfered as GRID then will be *
 * used in COSTPATH function in ARC/INFO to find the least   *
 * cost path from a landig to a road cell.                    *
 *****************************************************/
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include <string.h>

#define B 999999.0    /* represent infinite number */
#define NA 900000.0   /* After a smallest is chosen,it will be assigned */
#define SPACE 30   /* grid space is 30 meters */
#define DS 1
#define BS 0.01
#define MAX 15
#define ROW 75
#define COLUMN 120
#define NOEARTH 999999.0
#define PNULTY 30000

int dtm[ROW][COLUMN], road[ROW][COLUMN], direc[ROW][COLUMN],RockCode;
int  soil[ROW][COLUMN], stream[ROW][COLUMN], direc_out[ROW][COLUMN];
int column = 500, column2, row = 500, row2, cellsize, nodata,tem_dir;
int is = 0, js = 0, ct  = 1;  /* is js are the smallest earth_work index */
float latitude, longitude, earth_o, ExcavComm, BridgeCost, Miscell;
float sum_es[ROW][COLUMN], earth_out[ROW][COLUMN], tem_sum;
char str1[MAX],str2[MAX],str3[MAX],str4[MAX],str5[MAX],str6[MAX];
float RunningWidth, SurfaceDepth, Ditch, Turnouts, TurnoutLenght, CutSlope;
float FillSlope,  ExcavRock, Pipe, PipeCost, ClearGrub, Seeding;
float TruckCost, TruckLoad, SubgradeWidth,TotalVolume;
float SurfaceCost1, SurfaceCost2, EndhaulCost;
/* TotalVolume is the total volume in a landing. The more the volume,
 *  the more effective of the varible cost
 */
void get_data();   /*   get data from ascii file */
void earth_rd();   /*   calculate Total cost around road cells*/
void earth_ew(int, int);       /*calculate 8 ways of total cost  */
```

```
                        /* around a cell*/
float slope_cal(float, float);   /* Variable cost ( hauling cost) */
float construc_cal(int, int, float, float, float); /* construction cost */
void find_small();  /*  find smallest earth work cell */
void put_data();    /* final sum_es to ascii file */
void put_header(FILE *);
void get_header(FILE *);
void get_temp(FILE *);



void main()
{
  int flag = 1;
  get_data();
  earth_rd();
  find_small();
  while (flag < row*column - ct)  {
    earth_ew(is, js);
    find_small();
    flag++;
  }
  put_data();
}

/* Function to get data from ascii file and initial earth_w and slope */
void get_data()
{
  int i, j;
  FILE *ifpt1, *ifpt2, *ifpt3, *ifpt4, *ifpt5;
  if ((ifpt1 = fopen("hdem_a", "r")) == NULL) {  /* open for reading DErow file */
    printf("\nCannot open hdem_a - by");
    exit(1);
  }
        get_header(ifpt1);
        for (i = 0; i < row; ++i)  {
          for (j = 0; j < column; ++j) {
            fscanf(ifpt1, "%d", &dtm[i][j]);
/*          printf("dtm[%d][%d]    %d  ", i,j, *(*(dtm+i)+j));        */
          }
        }
        fclose(ifpt1);
        row2 = row;
        column2 = column;
/*--- read out road file ----------------------*/
        if ((ifpt2 = fopen("roads_a", "r")) == NULL) {  /*  existing road file */
          printf("\nCannot open roads_a - by");
          exit(1);
        }
        get_header(ifpt2);
        for (i = 0; i < row; ++i)  {
          for (j = 0; j < column; ++j) {
```

```
            fscanf(ifpt2, "%d", &road[i][j]);
            sum_es[i][j] = B;  /* initial all sum_es cells with 999999 */
            earth_out[i][j] = B; /* it is too bat that earth_out is 0 *
                                    /* when it is not a section of road*/
          }
        }
        fclose(ifpt2);
    if (row2 < row)          /* to keep the smallest of the */
        row = row2;          /* row and colume */
    if (column2 < column)
        column = column2;

/* get stream data */
 if ((ifpt4 = fopen("stream_a", "r")) == NULL) {
    printf("\nCannot open stream_a - by");
    exit(1);
  }
        get_header(ifpt4);
        for (i = 0; i < row; ++i)  {
          for (j = 0; j < column; ++j) {
            fscanf(ifpt4, "%d", &stream[i][j]);
 /*         printf("stream[%d][%d]    %d  ", i,j, *(*(stream+i)+j)); */
       }
        }
        fclose(ifpt4);
        if (row2 < row)          /* to keep the smallest of the */
          row = row2;          /* row and colume */
        if (column2 < column)
          column = column2;

/* get templete data */
  if ((ifpt5 = fopen("temp.dat", "r")) == NULL) {
    printf("\nCannot open temp.dat - by");
    exit(1);
  }
  get_temp(ifpt5);
  SubgradeWidth = RunningWidth + Ditch + 1 + ( 2*3*SurfaceDepth/RunningWidth);
/* get soil data   */
   if (RockCode != -9999)  {
     if ((ifpt3 = fopen("soil_a", "r")) == NULL)  {
          printf("\nCannot open soil_a - by");
          exit(1);
      }
        get_header(ifpt3);
        for (i = 0; i < row; ++i)  {
          for (j = 0; j < column; ++j) {
            fscanf(ifpt3, "%d", &soil[i][j]);
           }
        }
        fclose(ifpt3);

        if (row2 < row)          /* to keep the smallest of the */
```

```
            row = row2;            /* row and colume */
           if (column2 < column)
            column = column2;
  }
 return;
}

/* Function to get data from ascii file and initial earth_w and slope */
void earth_rd()
{
  int i, j;
    for (i = 0; i < row; ++i)  {
          for (j = 0; j < column; ++j) {

            if (road[i][j] != -9999)  {
              earth_o  = 0;
              earth_out[i][j] = 0;          /* out_put cell is 0  */
              ++ct;
              earth_ew(i, j);
             }
            if (dtm[i][j] == -9999)  {       /* if a DEM cell is nodata */
              earth_out[i][j] = -9999;      /* out put file is nodata */
              direc_out[i][j] = -9999;
            }

          }
   }
}

void earth_ew(int i, int j)
{
  int ip, jp; float slp;
  float earth_w[ROW][COLUMN], slope[ROW][COLUMN], ground_slp, distance;
  float cut_length;
  if (sum_es[i][j+1] > NA-DS && sum_es[i][j+1] < NA + DS)
  ;         /* once the smallest has been chosen, it can not chosen again. */
  else  {
    if ( ! (*(*(stream+i)+j+1) != -9999 && *(*(stream+i)+j+2) != -9999))  {
          if (j+1 < column && road[i][j+1] == -9999 && dtm[i][j+1] != -9999)  {
            ip = 1; jp = j + 1; distance = SPACE; cut_length = 2*distance;
            if (sum_es[i][j+1] <= B-DS)  {
              tem_sum = sum_es[i][j+1];
              tem_dir = direc[i][j+1];
              slp = 100*(dtm[i][j] - dtm[i][j+1])/distance;
/* Assign value to slope[i][j] according to the slope */
                   slope[i][j+1]= slope_cal(slp, distance);
              if (i-1>=0 && i+1< row && dtm[i-1][j+1] != -9999 && dtm[i+1][j+1] != -9999)  {
                    ground_slp = abs(dtm[i-1][j+1] - dtm[i+1][j+1])/cut_length;
                    earth_w[i][j+1] = construc_cal(ip, jp, distance, slp, ground_slp);
              }                          /* divided by 250 to match the unit */
              else  {                     /* of the slope */
                    earth_w[i][j+1] = NOEARTH;
```

```
                }
                sum_es[i][j+1] =  earth_o + earth_w[i][j+1] + slope[i][j+1];
                direc[i][j+1] = 5;
                if (tem_sum < sum_es[i][j+1]) {
                        sum_es[i][j+1] = tem_sum;
                        direc[i][j+1] = tem_dir;
                }
               }
              else {
                slp = 100*(dtm[i][j] - dtm[i][j+1])/distance;
               slope[i][j+1]= slope_cal(slp, distance);
               if (i-1>=0 && i+1< row && dtm[i-1][j+1] != -9999 && dtm[i+1][j+1] != -9999) {
                       ground_slp = abs(dtm[i-1][j+1] - dtm[i+1][j+1])/cut_length;
                       earth_w[i][j+1] = construc_cal(ip, jp, distance, slp, ground_slp);
               }
               else {
                       earth_w[i][j+1] = NOEARTH;
               }
/*             printf("earth_w[%d][%d]    %d  ", i,j, *(*(earth_w+i)+j+1)); */

             sum_es[i][j+1] =  earth_o + earth_w[i][j+1] + slope[i][j+1];
             direc[i][j+1] = 5;
             }
      }
    }
  }
 if (sum_es[i+1][j+1] > NA-DS && sum_es[i+1][j+1] < NA + DS)
 ;
 else {
   if (! (*(*(stream+i+1)+j+1) != -9999 && *(*(stream+i+2)+j+2) != -9999)) {
     if (i+1< row && j+1< column && road[i+1][j+1] == -9999  && dtm[i+1][j+1] != -9999) {
           ip = i+1; jp = j+1; distance = 1.4142*SPACE; cut_length = 2*distance;
           if (sum_es[i+1][j+1] <= B - DS) {
             tem_sum = sum_es[i+1][j+1];
             tem_dir = direc[i+1][j+1];
             slp = 100*(dtm[i][j] - dtm[i+1][j+1])/distance;
             slope[i+1][j+1]= slope_cal(slp, distance);
             if (j+2 < column && i+2< row && dtm[i][j+2] != -9999 && dtm[i+2][j] != -9999) {
                    ground_slp = abs(dtm[i][j+2] - dtm[i+2][j])/cut_length;
                    earth_w[i+1][j+1] = construc_cal(ip, jp, distance, slp, ground_slp);
              }
              else {
               earth_w[i+1][j+1] = NOEARTH;
              }
             sum_es[i+1][j+1] =earth_o +  earth_w[i+1][j+1] + slope[i+1][j+1];
             direc[i+1][j+1] = 6;
             if (tem_sum < sum_es[i+1][j+1]) {
               sum_es[i+1][j+1] = tem_sum;
               direc[i+1][j+1] = tem_dir;
             }
           }
           else {
```

```
            slp = 100*(dtm[i][j] - dtm[i+1][j+1])/distance;
            slope[i+1][j+1]= slope_cal(slp, distance);
            if (j+2 < column && i+2< row && dtm[i][j+2] != -9999 && dtm[i+2][j] != -9999) {
              ground_slp = abs(dtm[i][j+2] - dtm[i+2][j])/cut_length;
              earth_w[i+1][j+1] = construc_cal(ip, jp, distance, slp, ground_slp);
            }
            else {
              earth_w[i+1][j+1] = NOEARTH;
            }
            sum_es[i+1][j+1] = earth_o + earth_w[i+1][j+1] + slope[i+1][j+1];

            direc[i+1][j+1] = 6;
          }
      }
    }
}
 if (sum_es[i+1][j] > NA-DS && sum_es[i+1][j] < NA + DS)
 ;
 else {
   if ( ! (*(*(stream+i+1)+j) != nodata && *(*(stream+i+2)+j) != nodata)) {
         if ( i+1 < row && road[i+1][j] == -9999 && dtm[i+1][j] != -9999) {
         ip = i+1; jp = j; distance = SPACE; cut_length = 2*distance;
           if (sum_es[i+1][j] <= B-DS) {
             tem_sum = sum_es[i+1][j];
             tem_dir = direc[i+1][j];
             slp = 100*(dtm[i][j] - dtm[i+1][j])/distance;
             slope[i+1][j]= slope_cal(slp, distance);
             if (j-1 >=0 && j+1 < column && dtm[i+1][j-1] != -9999 && dtm[i+1][j+1] != -9999) {
                   ground_slp = abs(dtm[i+1][j-1] - dtm[i+1][j+1])/cut_length;
                   earth_w[i+1][j] = construc_cal(ip, jp, distance, slp, ground_slp);
             }
             else {
                   earth_w[i+1][j] = NOEARTH;
             }
             sum_es[i+1][j] =  earth_o + earth_w[i+1][j] + slope[i+1][j];
             direc[i+1][j] = 7;
             if (tem_sum < sum_es[i+1][j]) {
                   sum_es[i+1][j] = tem_sum;
                   direc[i+1][j] = tem_dir;
             }
           }
           else {
             slp = 100*(dtm[i][j] - dtm[i+1][j])/distance;
             slope[i+1][j]= slope_cal(slp, distance);
             if (j-1 >=0 && j+1 < column  && dtm[i+1][j-1] != -9999 && dtm[i+1][j+1] != -9999) {
                   ground_slp = abs(dtm[i+1][j-1] - dtm[i+1][j+1])/cut_length;
                   earth_w[i+1][j] = construc_cal(ip, jp, distance, slp, ground_slp);
             }
             else {
                  earth_w[i+1][j] = NOEARTH;
             }
             sum_es[i+1][j] =  earth_o + earth_w[i+1][j] + slope[i+1][j];
```

```
            direc[i+1][j] = 7;
          }
        }
    }
}
  if (sum_es[i+1][j-1] > NA-DS && sum_es[i+1][j-1] < NA + DS)
  ;
  else {
    if (! (*(*(stream+i+1)+j-1) != nodata &&  *(*(stream+i+2)+j-2) != nodata)) {
          if (i+1 < row && j-1 >=0 && road[i+1][j-1] == -9999 && dtm[i+1][j-1] != -9999) {
          ip = i+1; jp = j-1; distance = 1.4142*SPACE; cut_length = 2*distance;
          if (sum_es[i+1][j-1] <= B-DS) {
            tem_sum = sum_es[i+1][j-1];
            tem_dir = direc[i+1][j-1];
            slp = 100*(dtm[i][j] - dtm[i+1][j-1])/distance;
            slope[i+1][j-1]= slope_cal(slp, distance);
            if (j-2 >= 0 && i+2 < row  && dtm[i][j-2] != -9999 && dtm[i+2][j] != -9999) {
                  ground_slp = abs(dtm[i][j-2] - dtm[i+2][j])/cut_length;
                  earth_w[i+1][j-1] = construc_cal(ip, jp, distance, slp, ground_slp);
            }
            else {
                  earth_w[i+1][j-1] = NOEARTH;
            }
            sum_es[i+1][j-1] = earth_o + earth_w[i+1][j-1] + slope[i+1][j-1];
            direc[i+1][j-1] = 8;
            if (tem_sum < sum_es[i+1][j-1]) {
                  sum_es[i+1][j-1] = tem_sum;
                  direc[i+1][j-1] = tem_dir;
            }
          }
          else {
            slp = 100*(dtm[i][j] - dtm[i+1][j-1])/distance;
            slope[i+1][j-1]= slope_cal(slp, distance);
            if (j-2 >= 0 && i+2 < row && dtm[i][j-2] != -9999 && dtm[i+2][j] != -9999) {
                  ground_slp = abs(dtm[i][j-2] - dtm[i+2][j])/cut_length;
                  earth_w[i+1][j-1] = construc_cal(ip, jp, distance, slp, ground_slp);
            }
            else {
                  earth_w[i+1][j-1] = NOEARTH;
            }
                  sum_es[i+1][j-1] = earth_o + earth_w[i+1][j-1] + slope[i+1][j-1];
                  direc[i+1][j-1] = 8;
          }
    }
  }
}
  if (sum_es[i][j-1] > NA-DS && sum_es[i][j-1] < NA + DS)
  ;
  else {
    if (! (*(*(stream+i)+j-1) != nodata && *(*(stream+i)+j-2) != nodata)) {
          if ( j-1 >= 0 && road[i][j-1] == -9999 &&  dtm[i][j-1] != -9999) {
```

```
            ip = i; jp = j-1; distance = SPACE; cut_length = 2*distance;
            if (sum_es[i][j-1] <= B -DS)  {
              tem_sum = sum_es[i][j-1];
              tem_dir = direc[i][j-1];
              slp = 100*(dtm[i][j] - dtm[i][j-1])/distance;
              slope[i][j-1]= slope_cal(slp, distance);
              if (i+1 < row && i-1 >= 0 && dtm[i-1][j-1] != -9999 && dtm[i+1][j-1] != -9999)  {
                    ground_slp = abs(dtm[i-1][j-1] - dtm[i+1][j-1])/cut_length;
                    earth_w[i][j-1] = construc_cal(ip, jp, distance, slp, ground_slp);
              }
              else  {
                    earth_w[i][j-1] = NOEARTH;

              }
              sum_es[i][j-1] = earth_o + earth_w[i][j-1] +slope[i][j-1];
              direc[i][j-1] = 1;
             if (tem_sum < sum_es[i][j-1]) {
                   sum_es[i][j-1] = tem_sum;
                   direc[i][j-1] = tem_dir;
             }
            }
           else  {
             slp = 100*(dtm[i][j] - dtm[i][j-1])/distance;
             slope[i][j-1]= slope_cal(slp, distance);
               if (i+1 < row && i-1 >= 0 && dtm[i-1][j-1] != -9999 && dtm[i+1][j-1] != -9999)  {
                     ground_slp = abs(dtm[i-1][j-1] - dtm[i+1][j-1])/cut_length;
                     earth_w[i][j-1] = construc_cal(ip, jp, distance, slp, ground_slp);
               }
               else  {
                     earth_w[i][j-1] = NOEARTH;
               }
               sum_es[i][j-1] = earth_o + earth_w[i][j-1] +slope[i][j-1];
               direc[i][j-1] = 1;
           }
     }
   }
 }
 if (sum_es[i-1][j-1] > NA-DS && sum_es[i-1][j-1] < NA + DS)
 ;
 else  {
   if ( !(*(*(stream+i-1)+j-1) != nodata && *(*(stream+i-2)+j-2) != nodata))  {
         if (i-1 >= 0 && j-1 >= 0 && road[i-1][j-1] == -9999 && dtm[i-1][j-1] != -9999)  {
           ip = i-1; jp= j-1; distance = 1.4142*SPACE; cut_length = 2*distance;
           if (sum_es[i-1][j-1] <= B-DS)  {
             tem_sum = sum_es[i-1][j-1];
             tem_dir = direc[i-1][j-1];
             slp = 100*(dtm[i][j] - dtm[i-1][j-1])/distance;
             slope[i-1][j-1]= slope_cal(slp, distance);
             if (i-2 >= 0  && j-2 >= 0 && dtm[i-2][j] != -9999 && dtm[i][j-2] != -9999)  {
                   ground_slp = abs(dtm[i-2][j] - dtm[i][j-2])/cut_length;
                   earth_w[i-1][j-1] = construc_cal(ip, jp, distance, slp, ground_slp);
             }
```

```
                else {
                    earth_w[i-1][j-1] = NOEARTH;
                }
                sum_es[i-1][j-1] =  earth_o + earth_w[i-1][j-1] + slope[i-1][j-1];
                direc[i-1][j-1] = 2;
                if (tem_sum < sum_es[i-1][j-1]) {
                    sum_es[i-1][j-1] = tem_sum;
                    direc[i-1][j-1] = tem_dir;
                }
              }
              else {
                slp = 100*(dtm[i][j] - dtm[i-1][j-1])/distance;
                slope[i-1][j-1]= slope_cal(slp, distance);
                if (i-2 >= 0  && j-2 >= 0 && dtm[i-2][j] != -9999 && dtm[i][j-2] != -9999)  {
                    ground_slp = abs(dtm[i-2][j] - dtm[i][j-2])/cut_length;
                    earth_w[i-1][j-1] = construc_cal(ip, jp, distance, slp, ground_slp);
                }
                else {
                    earth_w[i-1][j-1] = NOEARTH;
                }
              sum_es[i-1][j-1] = earth_o +  earth_w[i-1][j-1] + slope[i-1][j-1];
              direc[i-1][j-1] = 2;
              }
            }
    }
}

  if (sum_es[i-1][j] > NA-DS && sum_es[i-1][j] < NA + DS)
  ;
  else {
    if ( ! (*(*(stream+i-1)+j) != nodata && *(*(stream+i-2)+j) != nodata))  {
          if (i-1 >= 0  && road[i-1][j] == -9999 && dtm[i-1][j] != -9999)  {
            ip = i-1; jp =j; distance = SPACE; cut_length = 2*distance;
            if (sum_es[i-1][j] <= B - DS)  {
              tem_sum = sum_es[i-1][j];
              tem_dir = direc[i-1][j];
              slp = 100*(dtm[i][j] - dtm[i-1][j])/distance;
              slope[i-1][j]= slope_cal(slp, distance);
              if (j-1 >= 0  && j+1 < column && dtm[i-1][j-1] != -9999 && dtm[i-1][j+1] != -9999)  {
                    ground_slp = abs(dtm[i-1][j+1] - dtm[i-1][j-1])/cut_length;
                    earth_w[i-1][j] = construc_cal(ip, jp, distance, slp, ground_slp);
              }
              else {
                    earth_w[i-1][j] = NOEARTH;
              }
              sum_es[i-1][j] =earth_o +  earth_w[i-1][j] +slope[i-1][j];
              direc[i-1][j] = 3;
              if (tem_sum < sum_es[i-1][j]) {
                sum_es[i-1][j] = tem_sum;
                direc[i-1][j] = tem_dir;
              }
            }
```

```
            else {
              slp = 100*(dtm[i][j] - dtm[i-1][j])/distance;
              slope[i-1][j]= slope_cal(slp, distance);
              if (j-1 >= 0  && j+1 < column && dtm[i-1][j-1] != -9999 && dtm[i-1][j+1] != -9999)  {
                    ground_slp = abs(dtm[i-1][j+1] - dtm[i-1][j-1])/cut_length;
                    earth_w[i-1][j] = construc_cal(ip, jp, distance, slp, ground_slp);
              }
              else {
                earth_w[i-1][j] = NOEARTH;
              }
              sum_es[i-1][j] =earth_o +  earth_w[i-1][j] +slope[i-1][j];
              direc[i-1][j] = 3;
            }
      }
    }
  }
  if (sum_es[i-1][j+1] > NA-DS && sum_es[i-1][j+1] < NA + DS)
  ;
  else {
    if (*(*(stream+i-1)+j+1) != nodata && *(*(stream+i-2)+j+2) != nodata)  {
          if (i-1>= 0 && j+1<column && road[i-1][j+1] == -9999 && dtm[i-1][j+1] != -9999) {
            ip = i-1; jp = j+1; distance = 1.4142*SPACE; cut_length = 2*distance;
            if (sum_es[i-1][j+1] <= B- DS)  {
              tem_sum = sum_es[i-1][j+1];
              tem_dir = direc[i-1][j+1];
              slp = 100*(dtm[i][j] - dtm[i-1][j+1])/distance;
              slope[i-1][j+1]= slope_cal(slp, distance);
              if (i-2 >= 0  && j+2 < column && dtm[i-2][j] != -9999 && dtm[i][j+2] != -9999)  {
                    ground_slp = abs(dtm[i-2][j] - dtm[i][j+2])/cut_length;
                    earth_w[i-1][j+1] = construc_cal(ip, jp, distance, slp, ground_slp);
              }
              else {
                    earth_w[i-1][j+1] = NOEARTH;
              }
              sum_es[i-1][j+1] =  earth_o + earth_w[i-1][j+1] + slope[i-1][j+1];
              direc[i-1][j+1] = 4;
              if (tem_sum < sum_es[i-1][j+1]) {
                   sum_es[i-1][j+1] = tem_sum;
                   direc[i-1][j+1] = tem_dir;
              }
            }
            else {
              slp = 100*(dtm[i][j] - dtm[i-1][j])/distance;
              slope[i-1][j]= slope_cal(slp, distance);
              if (i-2 >= 0  && j+2 < column && dtm[i-2][j] != -9999 && dtm[i][j+2] != -9999) {
                    ground_slp = abs(dtm[i-2][j] - dtm[i][j+2])/cut_length;
                    earth_w[i-1][j+1] = construc_cal(ip, jp, distance, slp, ground_slp);
              }
              else {
                    earth_w[i-1][j+1] = NOEARTH;
              }
              sum_es[i-1][j+1] = earth_o +  earth_w[i-1][j+1] + slope[i-1][j+1];
```

```
            direc[i-1][j+1] = 4;
          }
      }
    }
  }
  return;
}

/* Function to assign slope value */
float slope_cal(float sll, float trav_dist_m)
{                          /* sll for slp, trav_dist for distance */
  float vari_cost, vari_total,trav_dist;
  trav_dist = 3.28*trav_dist_m;  /* 1 meter = 3.28 feet */
  if (sll > -18.001 && sll <= -18 )
    vari_cost =  TruckCost*(8.8/60)*trav_dist*TotalVolume/TruckLoad/5280;
  else if (sll > -18 && sll <= -17)
    vari_cost =  TruckCost*(8.4/60)*trav_dist*TotalVolume/TruckLoad/5280;
  else if (sll > -17 && sll <= -16)
    vari_cost =  TruckCost*(8.0/60)*trav_dist*TotalVolume/TruckLoad/5280;
  else if (sll > -16 && sll <= -15)
    vari_cost =  TruckCost*(7.6/60)*trav_dist*TotalVolume/TruckLoad/5280;
  else if ( sll > -15 && sll <= -14)
    vari_cost =  TruckCost*(7.2/60)*trav_dist*TotalVolume/TruckLoad/5280;
   else if (sll > -14 && sll <= -13)
    vari_cost =  TruckCost*(6.8/60)*trav_dist*TotalVolume/TruckLoad/5280;
  else if (sll > -13 &&sll <= -12)
    vari_cost =  TruckCost*(6.4/60)*trav_dist*TotalVolume/TruckLoad/5280;
  else if (sll > -12 &&sll <= -11)
    vari_cost =  TruckCost*(6.0/60)*trav_dist*TotalVolume/TruckLoad/5280;
  else if (sll > -11 &&sll <= -10)
    vari_cost =  TruckCost*(5.6/60)*trav_dist*TotalVolume/TruckLoad/5280;
  else if (sll > -10 &&sll <= -9)
    vari_cost =  TruckCost*(5.3/60)*trav_dist*TotalVolume/TruckLoad/5280;
  else if (sll > -9  && sll <= 2)
    vari_cost =  TruckCost*(5.1/60)*trav_dist*TotalVolume/TruckLoad/5280;
  else if (sll > 2 && sll <= 3)
    vari_cost =  TruckCost*(5.8/60)*trav_dist*TotalVolume/TruckLoad/5280;
  else if (sll > 3 && sll <= 4)
    vari_cost =  TruckCost*(6.4/60)*trav_dist*TotalVolume/TruckLoad/5280;
   else if( sll > 4 && sll <= 5)
    vari_cost =  TruckCost*(7.1/60)*trav_dist*TotalVolume/TruckLoad/5280;
  else if (sll > 5 && sll <= 6)
    vari_cost =  TruckCost*(7.8/60)*trav_dist*TotalVolume/TruckLoad/5280;
  else if (sll > 6 && sll <= 7)
    vari_cost =  TruckCost*(8.5/60)*trav_dist*TotalVolume/TruckLoad/5280;
  else if (sll > 7 && sll <= 8)
    vari_cost =  TruckCost*(9.1/60)*trav_dist*TotalVolume/TruckLoad/5280;
  else if (sll > 8 &&sll <= 9)
    vari_cost =  TruckCost*(9.8/60)*trav_dist*TotalVolume/TruckLoad/5280;
  else if (sll > 9 &&sll <= 10)
    vari_cost =  TruckCost*(10.7/60)*trav_dist*TotalVolume/TruckLoad/5280;
  else if (sll > 10 && sll <= 11)
```

```
    vari_cost =  TruckCost*(11.6/60)*trav_dist*TotalVolume/TruckLoad/5280;
  else if (sll > 11 && sll <= 12)
    vari_cost =  TruckCost*(12.5/60)*trav_dist*TotalVolume/TruckLoad/5280;
   else if (sll > 12 && sll <= 13)
    vari_cost =  TruckCost*(13.4/60)*trav_dist*TotalVolume/TruckLoad/5280;
  else if (sll > 13 &&sll <= 14)
    vari_cost =  TruckCost*(14.3/60)*trav_dist*TotalVolume/TruckLoad/5280;
  else if (sll > 14 &&sll <= 15)
    vari_cost =  TruckCost*(15.2/60)*trav_dist*TotalVolume/TruckLoad/5280;
  else if (sll > 15 &&sll <= 16)
    vari_cost =  TruckCost*(16.1/60)*trav_dist*TotalVolume/TruckLoad/5280;
  else if ( sll > 16 && sll <= 17)
    vari_cost =  TruckCost*(17.0/60)*trav_dist*TotalVolume/TruckLoad/5280;
  else if (sll > 17 && sll <= 18)
    vari_cost =  TruckCost*(17.9/60)*trav_dist*TotalVolume/TruckLoad/5280;
  else
    vari_cost = PNULTY;

/*  printf("\%.1f ", vari_cost); */
  return(vari_cost);
}

/* Function to find the smallest in sum_es grid */
void find_small()
{
  /* put sum_es cell value to a one dimesion array */
  int i,j;
  tem_sum = B; earth_o = 0;
  for (i = 0; i < row; ++i)  {
     for (j = 0; j < column; ++j) {
/*     printf("sum_es[%d][%d]    %.3f  ", i,j, *(*(sum_es+i)+j)); */
          if (sum_es[i][j] < 0)
            sum_es[i][j] = NA -10;
          if (sum_es[i][j] > NA-DS && sum_es[i][j] < NA + DS)
          ;
          else  {
            if (sum_es[i][j] <= B - DS && sum_es[i][j] > 0.001) {
                  if (tem_sum > sum_es[i][j])  {
                     tem_sum = sum_es[i][j];
                     is = i; js = j;
                     tem_dir = direc[i][j];
                  }
            }
          }
     }
  }
  if (tem_sum > NA -DS && tem_sum < NA + DS)
  ;
  else  {
    if (tem_sum <= B - DS &&  tem_sum > 0.001) {
          earth_out[is][js] = tem_sum;
          direc_out[is][js] = tem_dir;
```

```
            earth_o = tem_sum;
            sum_es[is][js] = NA;
/*          printf("%.0f ", earth_out[is][js]); */
      }
    }
}


/* Function to put data as ascii files */
void put_data()
{
  int i,j;
  FILE *ofp1, *ofp2;
  ofp1 = fopen("ear_out", "w");     /* open for writing DErow file */
    put_header( ofp1);

  ofp2 = fopen("dir_out", "w");    /* open for writing back link  file */
    put_header( ofp2);

  for (i = 0; i < row; ++i)  {
  for (j = 0; j < column; ++j) {

            fprintf(ofp1, "%.1f ", earth_out[i][j]);
            fprintf(ofp2, "%d ", direc_out[i][j]);
      }
     fprintf(ofp1, "\n");
     fprintf(ofp2, "\n");
   }
   fclose(ofp1);
   fclose(ofp2);
}
/* Function to read header of gridascii file */
void get_header(FILE * ifp)
{

  fscanf(ifp, "%s%d", str1,&column);
  fscanf(ifp, "%s%d", str2,&row);
  fscanf(ifp, "%s%f", str3,&latitude);
  fscanf(ifp, "%s%f", str4,&longitude);
  fscanf(ifp, "%s%d", str5,&cellsize);
  fscanf(ifp, "%s%d", str6,&nodata);
}
/* Function to read templete data */
void get_temp(FILE * ifp)
{
  char NoUseStr[100];
  fscanf(ifp, "%f%s", &TotalVolume, NoUseStr);
  fscanf(ifp, "%f%s", &RunningWidth, NoUseStr);
  fscanf(ifp, "%f%s", &SurfaceDepth, NoUseStr);
  fscanf(ifp, "%f%s", &Ditch, NoUseStr);
  fscanf(ifp, "%f%s", &Turnouts, NoUseStr);
  fscanf(ifp, "%f%s", &TurnoutLenght, NoUseStr);
```

```
  fscanf(ifp, "%f%s", &CutSlope, NoUseStr);
  fscanf(ifp, "%f%s", &FillSlope, NoUseStr);
  fscanf(ifp, "%f%s", &ExcavComm, NoUseStr);
  fscanf(ifp, "%f%s", &ExcavRock, NoUseStr);
  fscanf(ifp, "%f%s", &Pipe, NoUseStr);
  fscanf(ifp, "%f%s", &PipeCost, NoUseStr);
  fscanf(ifp, "%f%s", &BridgeCost, NoUseStr);
  fscanf(ifp, "%f%s", &ClearGrub, NoUseStr);
  fscanf(ifp, "%f%s", &Seeding, NoUseStr);
  fscanf(ifp, "%f%s", &Miscell, NoUseStr);
  fscanf(ifp, "%f%s", &TruckCost, NoUseStr);
  fscanf(ifp, "%f%s", &TruckLoad, NoUseStr);
  fscanf(ifp, "%d%s", &RockCode, NoUseStr);
  fscanf(ifp, "%f%s", &SurfaceCost1, NoUseStr);
  fscanf(ifp, "%f%s", &SurfaceCost2, NoUseStr);
  fscanf(ifp, "%f%s", &EndhaulCost, NoUseStr);
  fclose(ifp);
 }

/* Function to write header for asciigrid file */
void put_header(FILE * ofp)
{
  fprintf(ofp,"%-14s%d", str1,column);
  fprintf(ofp,"\n%-14s%d", str2, row);
  fprintf(ofp,"\n%-14s%.4f", str3, latitude);
  fprintf(ofp,"\n%-14s%.1f", str4, longitude );
  fprintf(ofp,"\n%-14s%d", str5, cellsize);
  fprintf(ofp,"\n%-14s%d\n", str6, nodata);
}

/* Function to calculate construction cost */
float construc_cal(int ii, int jj, float Dist_m, float grade, float GroundSlp)
{
  float Turnout_f,CutAreaH, SlpDistan, Dist, Bridge_cost=0, Pipe_cost;
  float ConstrucCost, SeedingCost, SeedingAcre,  Mis_cost;
  double x1, x2, x21, x22, y1, y2, B1, B2, D, G, k, a, b, c;
  double CutVolume, ExcavCost, ClearGrubCost;
  float AreaA, AreaB, x1x2, y1y2, x1B1, x2B2, y1B1, y2B2;
  float Surface_cost, Surface_cost1, Surface_cost2;
  float WithTurnout,Surfacing,Endhaul_cost = 0;

  k = 1.15;            /* shrinkage factor */
  Dist = 3.28*Dist_m;   /* 1 mter = 3.28 ft */


/*
 * Turnout Factor
 */
  Turnout_f =1.5*Turnouts*TurnoutLenght/5280;
  WithTurnout = 1+Turnout_f;
/*
 * Ecavation Cost
```

```
*/
 if (GroundSlp <= 0.6) {
   D = (CutSlope - GroundSlp)/CutSlope;
   G = (FillSlope - GroundSlp)/FillSlope;
   a = k*G - G*G/D;
   b = 2*SubgradeWidth*G/D;
   c = -SubgradeWidth*SubgradeWidth/D;
   x21 = -(-b + sqrt(b*b - 4*a*c))/2*a;   /* X2 is negtive in cut_fill*/
   x22 = -(-b - sqrt(b*b - 4*a*c))/2*a;   /* graph */
   x2 = x21;
   if (x2 < x22)
        x2 = x22;    /* take the positive number */
   x1 = SubgradeWidth/D - (G/D)*x2;
   B1 = D*x1; B2 = G*x2;

   CutAreaH =  GroundSlp*B1*x1/2;
   CutVolume = WithTurnout*CutAreaH*Dist/27;  /*cubic feet */
   ExcavCost = ExcavComm*CutVolume;

   if (soil[ii][jj] == RockCode)
     ExcavCost = ExcavRock*CutVolume;
  }
  else  {  /*  ground slope > 0.6 */
   if (GroundSlp < CutSlope - 0.1)  {
         x1 = (CutSlope/(CutSlope - GroundSlp))*SubgradeWidth;
         CutAreaH = GroundSlp*SubgradeWidth*x1/2;
         CutVolume = WithTurnout*CutAreaH*Dist/27;  /*cubic feet */

         ExcavCost =  ExcavComm*CutVolume;                    /* yard */
         if (soil[ii][jj] == RockCode)
           ExcavCost = ExcavRock*CutVolume;

/* Endhaul cost for full Bench */
         Endhaul_cost = EndhaulCost*CutVolume;

        }
  else
         ExcavCost = B;  /* ground slope too big, so, cut will be too much. */
  }


/*
 * Clearing and Grubbing Cost
 */
  x1x2 = (x1 + x2)*(x1 + x2);
  y1y2 = pow((GroundSlp*x1 + GroundSlp*x2), 2);
  SlpDistan = sqrt(x1x2 + y1y2);
  ClearGrubCost = WithTurnout*ClearGrub*SlpDistan*Dist/43560;
/*
 * Seeding Cost
 */
  x1B1 = (x1 - B1)*(x1 - B1);
```

```
   y1B1 = (CutSlope*x1)*(CutSlope*x1);
   x2B2 = (x2 - B2)*(x2 - B2);
   y2B2 = (FillSlope*x2)*(FillSlope*x2);
   SeedingAcre = sqrt(x1B1 + y1B1) + sqrt(x2B2 + y2B2);
   SeedingCost = WithTurnout*Seeding*SeedingAcre*Dist/43560;

/* Bridge Cost */
  if (stream[ii][jj] >0 && stream[ii][jj] < 2) {  /*  if road cross stream  */
    Bridge_cost = BridgeCost;
  }
 else
  Bridge_cost = 0;

/*
 * Pipe  Cost
 */
  Pipe_cost = PipeCost*Pipe*Dist/5280; /* pipe cost per cell */

/*
 * Surfacing cost
 */
   Surfacing = (WithTurnout*RunningWidth +2)*Dist/9; /*sy/cell*/
   Surface_cost1 = SurfaceCost1*Surfacing;
   Surface_cost2 = SurfaceCost2*Surfacing*SurfaceDepth/RunningWidth/3;
   Surface_cost = Surface_cost1 + Surface_cost2;              /* cy/cell*/
/*
 *  Miscellanenous Cost
 */
   Mis_cost = Miscell*Dist/5280;  /* Mis_cost per cell */

/*
 * Total Construction Cost in $/grid
 */

    ConstrucCost = (ExcavCost + ClearGrubCost + SeedingCost + Bridge_cost+ Surface_cost +
Pipe_cost + Mis_cost + Endhaul_cost);
 /*  printf("\%.1f ", ExcavCost);  */
  return(ConstrucCost);
}
```

2 TEXT.C

```
/*  TEXT.C  by Shenglin Xu   8/18/95   */
/*****************************************************************/
/* This program tests the example in GRID (ESRI, 1992). The results are      */
/* the same as ESRI's. This means this program is correct.                   */
/*****************************************************************/
#include <stdio.h>
#include <math.h>

#define M 6 /* number of rows */
#define N 6 /* number of colums */
#define B 999999  /* represent infinite number */
#define NA 900000  /* After a smallest is chosen,it will be assigned this */
#define SPACE 1   /* DEM space is 30 meters */
#define DS 1

float sum_es[M][N],tem_sum,earth_out[M][N], earth_o = 0;
int direc[M][N],road[M][N], tem_dir, direc_out[M][N], dtm[M][N];   /* direc = direction of the road */

int is = 0, js = 0, ct  = 1;  /* is js are the smallest earth_work index */
void get_data();   /*  get data from ASCII file */
void earth_rd();  /*   calculate earthwork around road cells*/
void earth_ew(int, int);    /*   calculate 8 ways of earthwork around a cell*/
void find_small(); /*  find smallest earthwork cell */
void put_data();    /* final sum_es to ASCII file */

void main()
{
  int flag = 1;
  get_data();
  earth_rd();
  find_small();
   while (flag <= M*N-ct) {    /* ct is the number of road cell */
/*     printf("is = %d js = %d", is, js);   */
    earth_ew(is, js);
    find_small();
    flag++;
  }
  put_data();
}

/* Function to get data from ASCII file and initial earth_w and slope */
void get_data()
{
  int i, j;
  FILE *ifp1, *ifp2;          /* sum_es = sum of the earth_w and slope */
  if ((ifp1 = fopen("dem6_a", "r")) == NULL) {  /* open for reading DEM file */
    printf("\nCannot open dem6_a - by");
    exit(1);
  }
  if ((ifp2 = fopen("road6_a", "r")) == NULL) {  /*  existing road file */
```

```
    printf("\nCannot open road6_a - by");
    exit(1);
  }
   for (i = 0; i < M; ++i)  {
     for (j = 0; j < N; ++j) {
       fscanf(ifp1, "%d", &dtm[i][j]);
       fscanf(ifp2, "%d", &road[i][j]);
       sum_es[i][j] = B;  /* initial all sum_es cells with 999999 */
/*       printf("dtm[%d][%d]    %d  ", i,j, dtm[i][j]);  */
     }
   }
   fclose(ifp1);
   fclose(ifp2);
}

/* function to calculate earth_work */
void earth_rd()
{
  int i, j;
   for (i = 0; i < M; ++i)  {
     for (j = 0; j < N; ++j) {
       if (dtm[i][j] == -9999) {
          earth_out[i][j] = -9999;
          direc_out[i][j] = -9999;
        }

       if (road[i][j] != -9999)  {
          earth_o = 0;
              ct++;
          earth_out[i][j] = 0;          /* out_put cell is 0  */
          earth_ew(i, j);
        }
     }
   }
}

void earth_ew(int i, int j)
{
 if (sum_es[i][j+1] >= NA -DS && sum_es[i][j+1] <= NA +DS)
 ;
 else  {
   if ((j+1 < M) && road[i][j+1] == -9999 && (dtm[i][j+1] != -9999)) {
     if (sum_es[i][j+1] <= B - DS)  {   /* This cell already has been */
           tem_sum = sum_es[i][j+1];  /* calculated, so, save it and */
           tem_dir = direc[i][j+1];   /* compare with new one */
       sum_es[i][j+1] =  earth_o + 0.5*SPACE*(dtm[i][j] + dtm[i][j+1]);
           direc[i][j+1] = 5;
           if (tem_sum < sum_es[i][j+1]) { /* if the saved one is smaller */
         sum_es[i][j+1] = tem_sum;     /* then keep it */
         direc[i][j+1] = tem_dir;
       }
     }
```

```
    else {
      sum_es[i][j+1] = earth_o + 0.5*SPACE*(dtm[i][j] + dtm[i][j+1]);
          direc[i][j+1] = 5;        /* first time calculation of */
    }                        /* earthwork */
  }
}

if (sum_es[i][j-1] >= NA -DS && sum_es[i][j-1] <= NA +DS)
;
else {
  if ((j-1 >= 0) && road[i][j-1] == -9999 && dtm[i][j-1] != -9999) {
    if (sum_es[i][j-1] <= B-DS) {
      tem_sum = sum_es[i][j-1];
          tem_dir = direc[i][j-1];
          sum_es[i][j-1] =  earth_o + 0.5*SPACE*(dtm[i][j] + dtm[i][j-1]);
          direc[i][j-1] = 1;
          if (tem_sum < sum_es[i][j-1]) {
        sum_es[i][j-1] = tem_sum;
        direc[i][j-1] = tem_dir;
      }
    }
    else {
      sum_es[i][j-1] = earth_o + 0.5*SPACE*(dtm[i][j] + dtm[i][j-1]);
      direc[i][j-1] = 1;
    }
  }
}
if (sum_es[i+1][j] >= NA -DS && sum_es[i+1][j] <= NA +DS)
;
else {
 if ((i+1 < M) && road[i+1][j] == -9999 && dtm[i+1][j] != -9999) {
    if (sum_es[i+1][j] <= B-DS) {
          tem_sum = sum_es[i+1][j];
          tem_dir = direc[i+1][j];
          sum_es[i+1][j] =  earth_o + 0.5*SPACE*(dtm[i][j] + dtm[i+1][j]);
          direc[i+1][j] = 7;
          if (tem_sum < sum_es[i+1][j]) {
        sum_es[i+1][j] = tem_sum;
        direc[i+1][j] = tem_dir;
      }
    }
        else {
      sum_es[i+1][j] =  earth_o + 0.5*SPACE*(dtm[i][j] + dtm[i+1][j]);
          direc[i+1][j] = 7;
    }
  }
}
if (sum_es[i-1][j] >= NA -DS && sum_es[i-1][j] <= NA +DS)
;
else {
 if ((i-1 >= 0) && road[i-1][j] == -9999 && dtm[i-1][j] != -9999) {
   if (sum_es[i-1][j] <= B-DS) {
```

```
            tem_sum = sum_es[i-1][j];
            tem_dir = direc[i-1][j];
            sum_es[i-1][j] =  earth_o + 0.5*SPACE*(dtm[i][j] + dtm[i-1][j]);
            direc[i-1][j] = 3;
            if (tem_sum < sum_es[i-1][j]) {
          sum_es[i-1][j] = tem_sum;
          direc[i-1][j] = tem_dir;
        }
      }
      else {
        sum_es[i-1][j] =  earth_o + 0.5*SPACE*(dtm[i][j] + dtm[i-1][j]);
              direc[i-1][j] = 3;
      }
    }
}
if (sum_es[i+1][j+1] >= NA -DS && sum_es[i+1][j+1] <= NA +DS)
;
else {
 if (i+1<M && j+1<N && road[i+1][j+1] == -9999 && dtm[i+1][j+1] != -9999) {
    if (sum_es[i+1][j+1] <= B-DS)  {
            tem_sum = sum_es[i+1][j+1];
            tem_dir = direc[i+1][j+1];
            sum_es[i+1][j+1] =  earth_o + 1.414*0.5*SPACE*(dtm[i][j] + dtm[i+1][j+1]);
            direc[i+1][j+1] = 6;
            if (tem_sum < sum_es[i+1][j+1]) {
          sum_es[i+1][j+1] = tem_sum;
          direc[i+1][j+1] = tem_dir;
        }
          }
      else {
        sum_es[i+1][j+1] =  earth_o + 1.414*0.5*SPACE*(dtm[i][j] + dtm[i+1][j+1]);
            direc[i+1][j+1] = 6;
      }
    }
}
if (sum_es[i+1][j-1] >= NA -DS && sum_es[i+1][j-1] <= NA +DS)
;
else {
  if ((i+1 <N) && j-1 >= 0 && road[i+1][j-1] == -9999 && dtm[i+1][j-1] != -9999) {
    if (sum_es[i+1][j-1] <= B-DS)  {
            tem_sum = sum_es[i+1][j-1];
            tem_dir = direc[i+1][j-1];
            sum_es[i+1][j-1] =  earth_o + 1.414*0.5*SPACE*(dtm[i][j] + dtm[i+1][j-1]);
            direc[i+1][j-1] = 8;
            if (tem_sum < sum_es[i+1][j-1]) {
          sum_es[i+1][j-1] = tem_sum;
          direc[i+1][j-1] = tem_dir;
        }
      }
      else {
        sum_es[i+1][j-1] =  earth_o + 1.414*0.5*SPACE*(dtm[i][j] + dtm[i+1][j-1]);
              direc[i+1][j-1] = 8;
```

```
    }
   }
  }
  if (sum_es[i-1][j-1] >= NA -DS && sum_es[i-1][j-1] <= NA +DS)
  ;
  else {
   if (i-1 >=0 && j-1 >= 0 && road[i-1][j-1] == -9999 && dtm[i-1][j-1] != -9999) {
     if (sum_es[i-1][j-1] <= B-DS) {
           tem_sum = sum_es[i-1][j-1];
           tem_dir = direc[i-1][j-1];
           sum_es[i-1][j-1] = earth_o + 1.414*0.5*SPACE*(dtm[i][j] + dtm[i-1][j-1]);
           direc[i-1][j-1] = 2;
           if (tem_sum < sum_es[i-1][j-1]) {
         sum_es[i-1][j-1] = tem_sum;
         direc[i-1][j-1] = tem_dir;
           }
     }
     else {
       sum_es[i-1][j-1] = earth_o + 1.414*0.5*SPACE*(dtm[i][j] + dtm[i-1][j-1]);
           direc[i-1][j-1] = 2;
     }
   }
  }
 if (sum_es[i-1][j+1] >= NA -DS && sum_es[i-1][j+1] <= NA +DS)
 ;
 else {
 if (i-1 >=0 && j+1 < M && road[i-1][j+1] == -9999 && dtm[i-1][j+1] != -9999) {
    if (sum_es[i-1][j+1] <= B-DS) {
           tem_sum = sum_es[i-1][j+1];
           tem_dir = direc[i-1][j+1];
           sum_es[i-1][j+1] = earth_o + 1.414*0.5*SPACE*(dtm[i][j] + dtm[i-1][j+1]);
           direc[i-1][j+1] = 4;
           if (tem_sum < sum_es[i-1][j+1]) {
             sum_es[i-1][j+1] = tem_sum;
             direc[i-1][j+1] = tem_dir;
           }
         }
         else {
           sum_es[i-1][j+1] = earth_o + 1.414*0.5*SPACE*(dtm[i][j] + dtm[i-1][j+1]);
           direc[i-1][j+1] = 4;
         }
   }
 }
}

/* Function to find the smallest in sum_es grid */
void find_small()
{
  /* put sum_es cell value to a one dimesion array */
  int i,j;
  tem_sum = B;earth_o = 0;
  for (i = 0; i < M; ++i) {
```

```
    for (j = 0; j < N; ++j) {

          if (sum_es[i][j] >= NA -DS && sum_es[i][j] <= NA +DS)
          ;
          else {
            if ((sum_es[i][j] <= B-DS) && sum_es[i][j] > 1) {
                  if (tem_sum > sum_es[i][j])  {
                    tem_sum = sum_es[i][j];
                    is = i; js = j;
                    tem_dir = direc[i][j];

                  }
            }
          }
    }
  }
  if  (tem_sum >= NA -DS && tem_sum <= NA + DS)
  ;
  else  {
    if (tem_sum <= B-DS && tem_sum > 1 ) {
          earth_out[is][js] = tem_sum;
          direc_out[is][js] = tem_dir;
          earth_o = tem_sum;
          sum_es[is][js] = NA;      /* the smallest can not be rechosen */
    }
  }
}
/* Function to put data as ASCII files */
void put_data()
{
  int i,j;
  FILE *ofp1, *ofp2;
  ofp1 = fopen("ear_out", "w");     /* open for writing DEM file */
  ofp2 = fopen("dir_out", "w");    /* open for writing back link  file */
    for (i = 0; i < M; ++i)  {
        for (j = 0; j < N; ++j) {
          fprintf(ofp1, "%.1f ", earth_out[i][j]);
          fprintf(ofp2, "%d ", direc_out[i][j]);
    }
    fprintf(ofp1, "\n");
    fprintf(ofp2, "\n");
  }
  fclose(ofp1);
  fclose(ofp2);
}
```

APPENDIX C RESULTS

1 Results made by TEST.C

/* Source grid NODATA -9999 (input file)

-9999 1 1 -9999 -9999 -9999
-9999 -9999 1 -9999 -9999 -9999
-9999 -9999 -9999 -9999 -9999 -9999
-9999 -9999 -9999 -9999 -9999 -9999
-9999 -9999 -9999 -9999 -9999 -9999
2 -9999 -9999 -9999 -9999 -9999 -9999

/* Cost grid (input file)

1 3 4 4 3 2
4 6 2 3 7 6
5 8 7 5 6 6
1 4 5 -9999 5 1
4 7 5 -9999 2 6
1 2 2 1 3 4

/* Least cumulative cost (output file)

2.0 0.0 0.0 4.0 6.7 9.2
4.5 4.0 0.0 2.5 7.5 13.1
8.0 7.1 4.5 4.9 8.9 12.7
5.0 7.5 10.5 -9999.0 10.6 9.2
2.5 5.7 6.4 -9999.0 7.1 11.1
0.0 1.5 3.5 5.0 7.0 10.5

/* Direction (output file)

1 0 0 5 4 5
7 1 0 5 5 6
3 8 7 6 6 3
3 5 7 -9999 3 4
3 4 4 -9999 4 5
0 5 5 5 5 5

ROAD PLANNING PROJECT

Road Map

This map shows the selected roads in the study area in grid form.

Legend

Existing roads

Selected roads

SCALE 1:20,000

0 1250

ft

3. VER_TABLE USED IN PATHDISTANCE FUNCTION

```
/* This ver_table is made according to the variable cost formula.
/* The first colume is slope in degree, the second column is the
/* related cost. Above statement is not part of VER_TABLE.

-90 9999
-10.2 4.7
-9.65 4.5
-9.09 4.3
-8.53 4.0
-7.97 3.8
-7.41 3.6
-6.84 3.4
-6.28 3.2
-5.71 3.0
-5.14 2.8
-4.57 2.7
0 2.7
1.15 2.7
1.72 3.1
2.29 3.4
2.86 3.8
3.43 4.1
4.0 4.5
4.57 4.8
5.14 5.2
5.71 5.7
6.28 6.2
6.84 6.6
7.41 7.1
7.97 7.6
8.53 8.1
9.09 8.6
9.65 9.0
10.2 9.5
90 9999
```

4. TEMP.DAT used in ROAD.C

```
/* This ASCII file is used to provide road template etc. data
/* for ROAD.C. User can communicate with ROAD.C
/* by this file. The above statement is not part of  TEMP.DAT.

100000          TotalVolume
12          RunningWidth
8           SurfaceDepth
3           Ditch
5           Turnouts
100          TurnoutLength
2           CutSlope(.5:1)
0.67         FillSlope(1.5:1)
.5          ExcavationCost(common)$/mile
2           ExcavationCost(rock)$/mile
6           Pipes_Per_mile
300          Pipes_cost_(per_pipe)$
60000         BridgeCost_$
1500         ClearGrub_$/arce
200         Seeding_$/arce
10000         Miscellaneous(survey_and_design)_$/mile
51.34        TruckCost_$/h
25          TruckLoad
6               RockCode(____-9999_for_no_soil_data_file)
.1          Surfacing_cost/sy
5           Surfacing_cost/cy
1           EndhaulCost_/cy
```