Timber Transportation Optimization and Vehicle Routing: A GIS application in South Africa

By

S. Francois Oberholzer

A PAPER

submitted to

Department of Forest Engineering
Oregon State University

In partial fulfillment of
the requirements for the
degree of

Master of Forestry

Presented September 6, 2001
Commencement December 2001

AN ABSTRACT OF THE PAPER OF

S. Francois Oberholzer for the degree Master of Forestry in Forest Operations presented on September 6, 2001.

Title: Timber Transport Optimization and Vehicle Routing: A GIS application in South Africa

Abstract Approved: J Sessions

Dr. John Sessions

Evaluating different timber transport configurations for a large road network is a challenging task. Road curves are often too sharp to allow access to certain configurations. This paper evaluates the feasibility of using a geographical information system (GIS) road data set as a basis for determining the accessibility of plantation roads for several truck configurations and to determine the optimal road system for three given vehicle configurations by minimizing total cost. The main constraint considered is the ability of a vehicle configuration to move through the curves found in the road network of the Elandshoogte plantation, Mpumalanga, South Africa. There are two objectives in this project: 1) to evaluate the feasibility of using geographical information system (GIS) road data as a basis for determining the accessibility of plantation roads to several truck configurations and 2) to determine the optimal road system for the three given vehicle configuration by minimizing the total cost. Constraints are generated by calculating the amount of offtracking for each vehicle configuration for all the roads and road connections found on the plantation by using the OFFTRACK program (Erkert, 1989). The minimum cost network is solved by using NETWORK 2000 (Chung and Sessions, 2001).

TABLE OF CONTENTS

LIST OF TABLES

LIST OF ILLUSTRATIONS

1. INTRODUCTION

The transportation of timber from the harvesting area to the final processing facility is one of the largest parts of the mill delivered cost of timber in South Africa (Brink[1], *pers. comm.*). Opportunities exist for reducing transportation costs by optimizing various components of the transportation process. These include optimization of vehicle routing and truck scheduling, improving loading times, reducing miscellaneous waiting times at the mill and selection of appropriate transport configurations for a given area. The optimization of vehicle routing has received much attention (Wong, 1981). This paper will explore the feasibility of using a geographic information system (GIS) data set and the OFFTRACK program (Erkert, 1989) to determine the accessibility of plantation roads to several truck configurations, and to determine the optimal road system for the given vehicle combinations by using NETWORK 2000 (Chung and Sessions, 2001).

Sappi Forests Ltd embarked on a project in 2000 to evaluate their current transportation systems, and to compare them to new systems. There are many different vehicle configurations and route combinations that need to be considered. Comparing all possible configurations for a large area could be costly if each configuration is to be tested physically on the ground. It would be more efficient to verify the feasibility of each of these configurations by determining the behavior of the vehicle on plantation roads through the use of a computer program, thus reducing the number of possible configurations. To be a feasible road link, the vehicle configuration must be able to traverse all curves and road connections. All the roads in the network are existing roads. This paper will not consider building new roads or widening existing roads.

Three steps will be followed to convert the GIS data to information that can be used to determine vehicle accessibility and vehicle routing:

1. Extraction of information from the road coverage;
2. Determining the vehicle accessibility to the road network;
3. Determining the optimum route for the vehicle to take subject to the constraints identified in step two.

This paper is organized as follows. The forest operations and vehicle configurations used are discussed in Chapter 2. The methods for determining vehicle accessibility are discussed in Chapter 3 and the route optimization is discussed in Chapter 4. The results of the network analysis are presented in Chapter 5

[1] Andre Brink, Logistics Manager, Sappi Forests, Pietermaritzburg, South Africa

2. BACKGROUND

Elandshoogte plantation in the Mpumalanga province in South Africa was selected for this project. The plantation has 408.25 km of roads on 4596.41 ha, which results in a road density of 88.8m/ha. Roads occupy approximately 3% of the total area.

2.1 ROADS AND HARVESTING

There are three access roads leading to the plantation (Figure 2-4). Three types of roads are found on the plantation: 1) double lane five-meter wide paved surface roads, 2) double lane five-meter wide crowned gravel road and 3) single lane un-surfaced dirt roads that range from three to four meters in width. The centerlines and widths of all the roads on the plantation have been surveyed and incorporated into a GIS data file. The accuracy of the GIS data files will not be considered in this paper.

The main timber species on this plantation are *Pinus patula*, *P. taeda* and *P. elliottii*, which are planted at approximately 1200 trees per hectare. All the trees will be used for making pulp, and therefore no thinning takes place in the life of these stands. Trees are harvested at a rotation age of approximately 20 years, at which point the average volume per tree ranges between $0.2m^3$ and $0.4m^3$. Approximately 1.4 million tons will be harvested from 159 harvest units over the next 14 years (Table 2-1).

Table 2-1: Harvest volumes

Year	2001	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	**TOTAL**
Vol.(m^3)	9,126	54,376	73,743	184,760	157,026	197,376	261,111	165,301	65,051	100,984	127,954	1,396,808

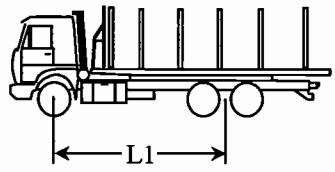
Trees are felled and limbed infield motor-manually. Tree length logs are extracted with cable skidders on level ground and with cable yarders on steep ground. On the landing trees are cut to 2.4m (8 ft) lengths, and stacked on the roadside to be transported. Loading takes place with a mobile tri-wheel loader.

2.2 TRANSPORTATION SYSTEMS

This paper considers three vehicle configurations used by Sappi Forests: Shorthaul, Longhaul and Rigid-and-drawbar

Shorthaul

The first configuration is the short rigid timber truck (Shorthaul, no articulation points) that have a highway payload of 12 tons and an off highway payload of 17 tons (Figure 2-1). This configuration is mainly used in short haul applications (haul distances less than ten kilometers). The timber is hauled from the harvest unit to a central depot where it is offloaded and stored. It will then be reloaded on the Longhaul configuration truck for transport to the mill. If haul distances to the mill are short (less than ten kilometers), the Shorthaul trucks can transport the timber directly to the mill.

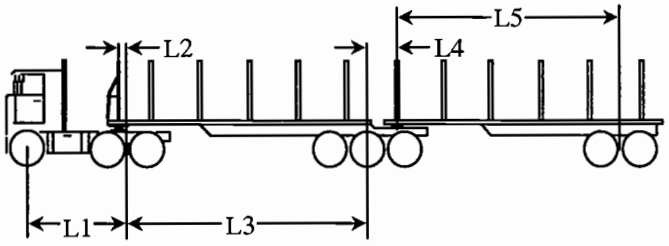


L1 = Vehicle length (Front axle to centerline drivers)

Figure 2-1: Shorthaul configuration

Longhaul

The Longhaul configuration works in conjunction with the Shorthaul system. This configuration consist of a tractor and two semi trailers, and can take a payload of 38 tons (Figure 2-2). The Longhaul configuration is used to move timber from depots to the mill, with haul distances ranging from a few kilometers to several hundred kilometers.

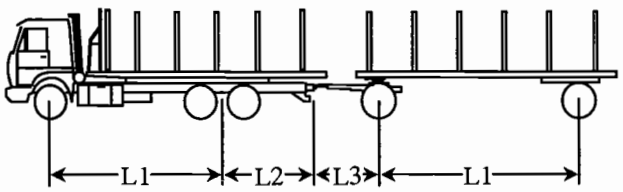


L1 = Vehicle length (Front axle to centerline drivers)
L2 = Tractor firth wheel position (ahead of centerline drivers)
L3 = First Trailer length (fifth wheel to first trailer tandems)
L4 = Trailer fifth wheel position (ahead of tandems)
L5 = Second trailer length (first trailer fifth wheel to second trailer tandems)

Figure 2-2: Longhaul configuration

Rigid-and-drawbar

The third configuration consists of a short rigid timber truck and a trailer (Rigid-and-drawbar). Timber is hauled from the harvest unit directly to the mill, and can take a payload of 32 tons (Figure 2-3). Haul distances vary from a few kilometers to several hundred kilometers.



L1 = Vehicle length (Front axle to centerline drivers)
L2 = Stinger length (centerline drivers to hitch)
L3 = Trailer reach (hitch to trailer fifth wheel)
L4 = Trailer length (trailer fifth wheel to last trailer axle)

Figure 2-3: Rigid-and-drawbar configuration

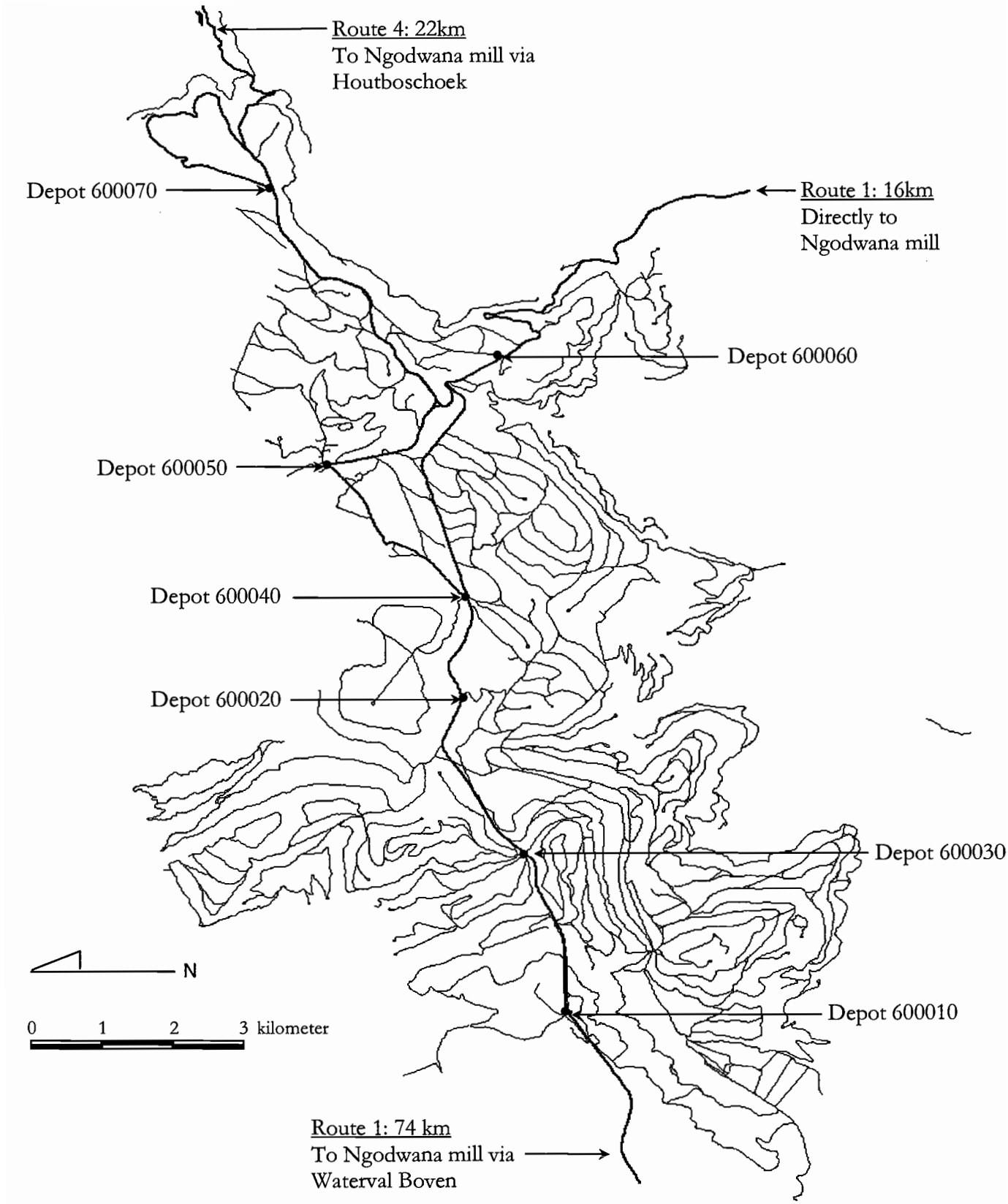


Figure 2-4: Elandshoogte map showing main routes and location of depots

3. METHOD AND DATA FOR VEHICLE ACCESS

Geographic Information Systems (GIS) have become commonplace in the forest industry. Sappi Forests has most of their geographical information such as roads, harvest units and soils available on GIS. These GIS databases will serve as the basis for the information required for the analysis in this paper.

Many of the plantation roads were built to provide access for silvicultural and fire crews, therefore hauling operations were not considered when the roads were constructed. Road grade, curve centerline radius and road width in the curve play a role in determining the ability of a given truck configuration to access a road. The radius of many of the curves in the plantations are too small for certain truck configurations when considering offtracking. Offtracking of vehicles occurs when the rear wheels of a vehicle do not follow the same path as the front wheels (Figure 3-1). On plantation roads with tight curves the amount of offtracking can be substantial, requiring wider roads. The amount of offtracking will depend upon the geometry of the vehicle, the radius of the curve, the length of the curve, and the location of the vehicle (Erkert, 1989). Thus, for a vehicle to have access to a road link it must be able to pass all the curves in the link with all of its wheels remaining on the road surface. Widening of roads is beyond the scope of this paper.

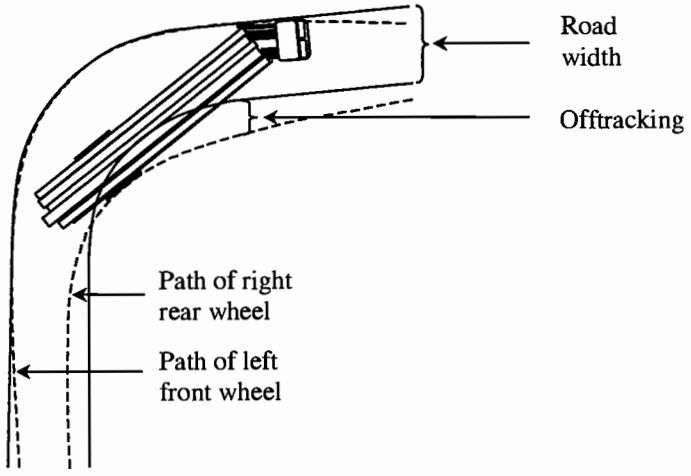


Figure 3-1: Vehicle offtracking

The following steps was used to convert the GIS data to information that can be used to determine vehicle accessibility and vehicle routing:

1. extraction of information from the road coverage and
2. determining the amount of offtracking from each road and road section.

Programs were compiled in Microsoft Visual Basic 6.0 to extract data from files, to edit files, and to do calculations.

3.1 EXTRACTION OF INFORMATION FROM THE ROAD COVERAGE

The road coverage is used as the source of all the road data. It contains information regarding the road width and length, as well as From Node and To Node for each road arc. Each road is made up of between two to several hundred nodes of which the X- and Y-coordinates are known. The nodes are connected by vectors known as road arcs.

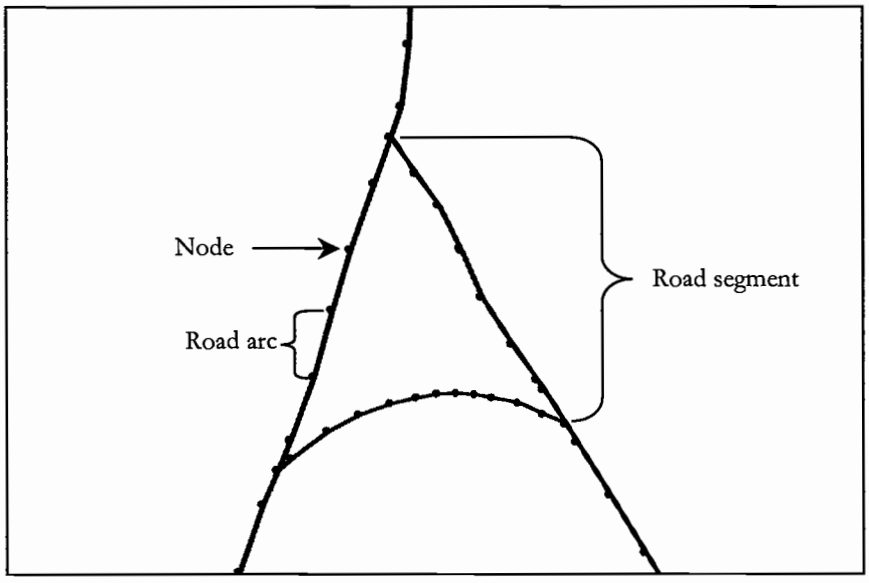


Figure 3-2: Road section, nodes and road arcs

The coordinates for each node can be derived by converting the ARC/INFO roads coverage into an AutoCAD Drawing Interchange File (DXF) by using the ARC command ARCDXF. The total road network has 1177 road segments. The coordinates were extracted from the DXF file (Appendix A: Step 1), and the coordinates for each road segment were written to a separate file (called "fw_line") for each road segment. The azimuth and distance between consecutive nodes were calculated for each road, and the information was written to a separate file (called "fw_ang") for each road (Appendix A: Step 2).

Offtracking can occur anywhere on a road segment, but it can also occur between road segments (Figure 3-3). The length of the truck could be longer than two connected road segments, and it is therefore necessary to connect three road sections in a row should the middle section be shorter than the length of the longest truck. The reason for this is that the truck still develops offtracking while in the curve. The longer the truck is in the curve, the more offtracking develops up to the point where maximum offtracking is reached in a sharp curve, or steady offtracking is reached otherwise. It is therefore necessary to create a new set of files (called "combcurv") that contain the information of the road segment connections (Appendix A: Step 5). Wherever a road is connected to another, a file was created. The last 50 meters of a road and the first 50 meters of the adjoining road are used in the new file. In the situation where a road connected to an adjoining road, and the adjoining road is less than 50 meters long, a third consecutive road was connected to the second. Figure 3-3 gives an indication of this process. If road number 708 is considered, the program will connect road number 672 with it. Since number 672 is shorter than 50 meters, a third road is considered. Two files will therefore be created in this process: a road file containing segments 708 – 672 – 668, and a road file containing 708 – 672 – 696. The azimuth and distance between consecutive nodes were calculated again for the combined road

segments, and the information was written to a separate file (comcvang) for each road file (Appendix A: Step 7).

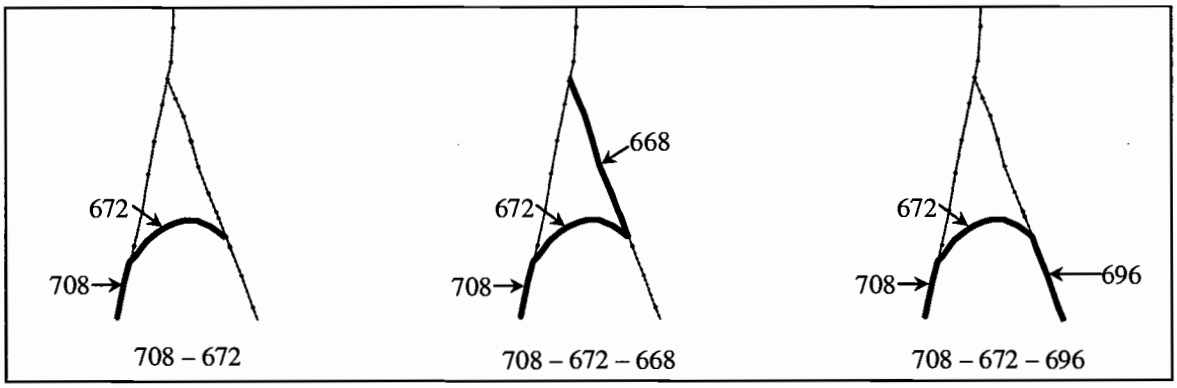


Figure 3-3: Combined road sections

The radii between consecutive nodes were determined with a formula that calculates the radius of a circle with any three points on the edge of the circle (Appendix A: Step 10) (Figure 3-4).

- $x1$ = X coordinate of node 1 $y1$ = Y coordinate of node 1
- $x2$ = X coordinate of node 2 $y2$ = Y coordinate of node 2
- $x3$ = X coordinate of node 3 $y3$ = Y coordinate of node 3
- *xcircle* = X coordinate of the midpoint of the circle
- *ycircle* = Y coordinate of the midpoint of the circle

Formulas:

- $A1 = x1^2 + y1^2 - x2^2 - y2^2$ $A2 = x1^2 + y1^2 - x3^2 - y3^2$
- $X1 = x1 - x2$ $Y1 = y1 - y2$
- $X2 = x1 - x3$ $Y2 = y1 - y3$
- $xcircle = \frac{A2 * Y1 - A1 * Y2}{2 * X2 * Y1 - 2 * X1 * Y2}$ $ycircle = \frac{A1 - 2 * X1 * xcircle}{2 * Y1}$

The radius is therefore:

- $radius = \sqrt{(x2 - xcircle)^2 + (y2 - ycircle)^2}$

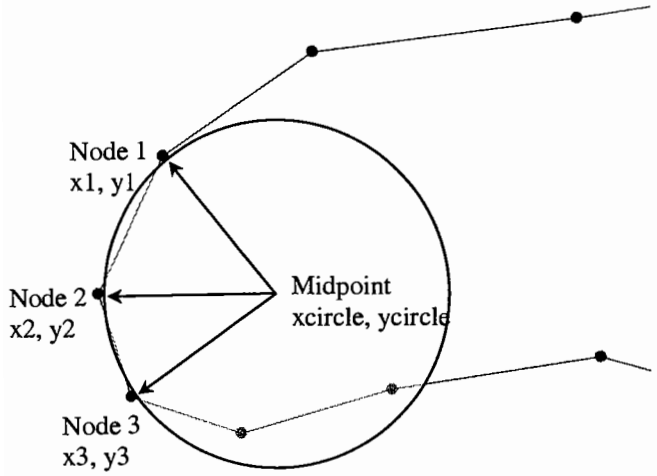


Figure 3-4: Radius calculation

All the information associated with each road segment was written to a separate file for each road file (called "fulldata"). These files contain the X and Y coordinates, azimuths and distances between consecutive nodes, radius at each node and an identification number for each road in that file. With this information, files can be compiled in such a way that it can be read in OFFTRACK (Erkert, 1989). The OFFTRACK program, however, is limited to road files with less than 100 nodes, and it is therefore necessary to break the roads into sections less than 100 nodes (Appendix B). The road files that were longer than 100 nodes were broken up by allowing overlap between the new road files so that offtracking occurring at the end of a road could be captured in the new files. If a road file contained between 100 and 199 nodes, it was split into two new files. If it contained between 200 and 299 nodes it was split into three new files, and 300 to 399 nodes were split into four files. No road files had more than 400 nodes.

3.2 DETERMINING THE AMOUNT OF OFFTRACKING

Offtrack

Several methods are available to calculate the amount of road width that the vehicle will need. These include numerical or algebraic solution of the equations of vehicle motion through the curve, vehicle simulators (draft model or field size), empirical equations based upon field observations, and deep curve equilibrium formulas (Erkert, 1989; Sessions 1985). The majority of roads to be considered were made up of compound curves, and many analytical techniques are not capable of analyzing the off tracking of truck and trailer combinations through a set of compound curves (Cain and Langdon, 1982; SAE, 1964). Physical vehicle simulators would be too cumbersome to use. A computer simulation program was considered to be the most appropriate method to use in this application.

The OFFTRACK computer simulation program was developed by Erkert (1989) specifically for the forest road situation. The Tractrix equations are used to describe how a vehicle negotiates a given road (Erkert, 1989). In order to run OFFTRACK, three input screens must first be completed. These are road design, vehicle combination and road segments. The idea is to put each road segment and all the combined road files created (3.1 Extraction of information from the road coverage) through OFFTRACK in order to determine if it would be feasible for a given truck configuration to move through the curves.

Road design:

The design parameters consist of the minimum travel way width of the road and the minimum steering correction. A 0.6m (2.0 ft) minimum steering correction is customary to account for driver error, and was used as default in this application. The road width varies between three and five meters, depending on the type of road.

Vehicle combination:

The input for any vehicle are the wheelbases and hinge point offsets for each unit in the combination as denoted in Figure 2-1, Figure 2-2 and Figure 2-3.

Road segments:

The road segment input screen requires three pieces of information for each road segment:

- the radii at each node in a segment,
- the beginning distance (distance from the beginning of the road to the beginning of the segment), and
- ending distance (distance from the beginning of the road to the end of the segment).

The files generated from the road coverage in Chapter 3 duplicates the road segments file used in OFFTRACK. The files were named RD* (* = 1 to 7951).

Since OFFTRACK was written to operate in MSDOS, the program relies on manual input of commands, which makes it impractical to enter all the files that were created manually. In order to overcome this problem, a procedure was compiled to determine the offtracking for each road

file, using the OFFTRACK program in a macro. This macro was compiled by Sessions[2] (2001). It uses a program called STUFFKEY that records keystrokes on the keyboard, and feeds these keystrokes to OFFTRACK (Appendix C). This allowed the input process of 7951 files to be automated. The results generated by the OFFTRACK program were printed to a separate file for each road called O-RD* (* = 1 to 7951). Each of these files is 640 kilobytes in size. Due to size of the print-files, the OFFTRACK macro had to run off a network drive that can accommodate the 4.5-gigabyte directory.

These output files were analyzed with a Visual Basic program, and the maximum offtracking for each road segment and road-connection segment was recorded (Appendix D). Only the Rigid-and-drawbar configuration were analyzed with OFFTRACK. Certain assumptions are made concerning the Shorthaul and Longhaul system, and these are discussed under Assumptions, p. 20. Even though the process is automated, it is still slow because of the interaction between the execution of the macro and OFFTRACK.

The results from OFFTRACK can be used to determine which critical curves in the road network would limit the use of a certain configuration. The "fulldata" files were used to link the offtracking for each file back to specific roads. The road network for a given truck configuration can consequently be compiled to eliminate the critical curves (see section 4.2). The OFFTRACK program has difficulty calculating offtracking when the radii involved are less than the length of a truck. The files with radii less than 20 meters were written to a separate directory where they could be manually manipulated and entered into OFFTRACK.

[2] Bren Sessions, Beaverton, Oregon.

Assumptions

- All the vehicles considered have adequate power and traction to perform their tasks.
- Steering and trailing axles are fixed perpendicular to the longitudinal axis of the vehicle.
- The Shorthaul trucks can negotiate the entire road network.
- The Longhaul trucks can only use the main hauling routes, since these vehicles require a large area to turn around.
- Road construction and road widening are not considered.

4. DETERMINING THE OPTIMUM ROUTE

4.1 NETWORK ANALYSIS

Network analysis techniques are frequently used in forest applications to determine the most economical network of roads for hauling timber, considering fixed and variable costs. Several programs have been developed to facilitate this analysis, e.g. MINCOST, TIMBER TRANSPORT, and NETWORK (Moore, 1987). In a network problem the objective is to determine the optimal route to use given a set of road segments, a set of harvest units (entry points into the network), and a destination (exit point) from the network (Sessions and Sessions, 1988). In a study done by Moore (1987), he found that the NETWORK model consistently gave the best solutions for five planning areas that were investigated. Both the MINCOST and TIMBER TRANSPORT models generated sub optimal solutions for large data sets (Moore, 1987; Kirby, 1981). NETWORK was revised in 2000 and is now called NETWORK 2000. The program uses three different heuristic algorithms with intelligent neighborhood search rules. It calculates the minimum cost network by using a shortest path based algorithm to solve the fixed and variable cost problem (Chung and Sessions, 2000). Since a heuristic approach is used to solve the problem, an optimal solution is not guaranteed.

The theory of network analysis involves determining the least impedance path given generators and attractors, more often called sales and mills respectively, where as the paths of impedance are called links or arcs (Moore, 1987; Shen, 1988). For the remainder of this study, sales refer to roads that have any amount of timber volume assigned to them, and the actual compartments where the timber will be harvested are referred to as harvest units. The links provide the access from the harvest units to the mill. The points of intersection of the links are called nodes. Nodes

may be located at road intersections, abrupt changes between fixed and variable costs, or strategic locations to gain more accuracy in the network solution. In this application, the nodes were taken as the middle points of each road (Figure 4-1). Associated with each link is a variable cost component, which is the cost per unit volume to haul one unit across the link, and a fixed cost factor, which is the cost to construct or maintain the link. Each segment could be a one-way or two-way link, depending on whether transport is allowed in both directions or not. Each harvest unit is described by its entry nodes, its destination node (mill), its volume, and the year in which it is scheduled to be transported.

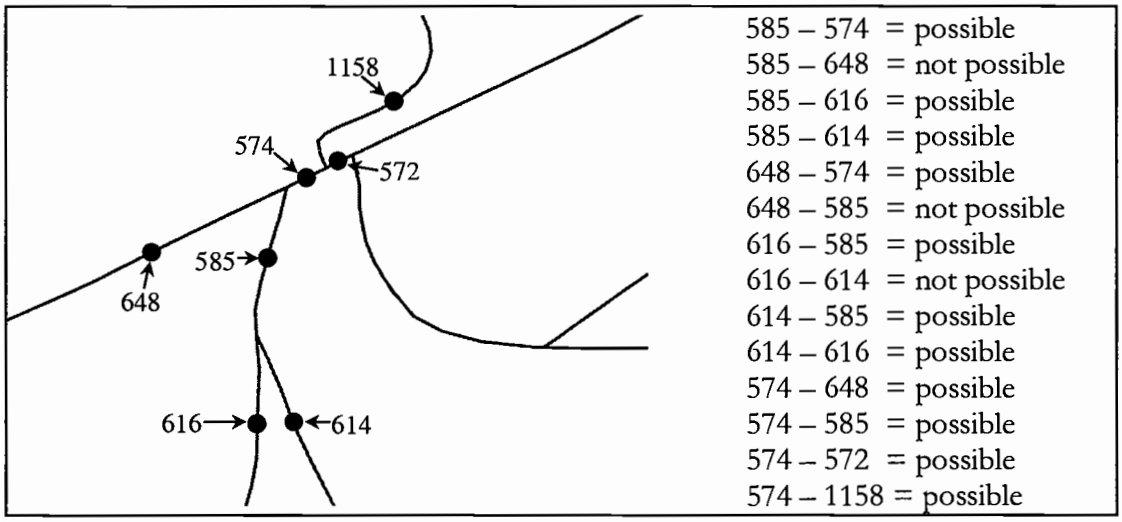


Figure 4-1: Network nodes – route midpoints

4.2 NETWORK FORMULATION

In this paper the route optimization problem is a cost minimization problem. The problem is to solve the minimum cost route, taking into account the following factors: 1) constraints in the road network, 2) the volume of timber to transport, 3) the variable transport cost of each configuration, 4) the variable road maintenance cost associated with each configuration, 5) the variable transfer cost at depots, and 6) the fixed construction cost of depots. The problem can be stated as:

$$\text{Minimize } Z = \sum_{t=1}^{Q}\sum_{i=1}^{M}\sum_{j=1}^{N} x_{ijt} c_{ijt} + \sum_{t=1}^{Q}\sum_{i=1}^{M} y_{jt} p_{jt}$$

where:

x_{ijt} = flow of timber from source i over link j in period t

c_{ijt} = discounted variable cost from source i over link j in period t

y_{jt} = 0,1 integer for link j in period t, where $y = 1$ when depot is constructed at link j in period t

p_{jt} = discounted fixed cost of constructing a depot at link j in period t

S_{it} = Source number i in period t

D_t = total volume to be delivered at the destination in period t

B = an arbitrary large constant greater than the total volume of timber to be transported to the mill in any period

subject to:

i. $x_{ijt} \geq 0$

ii. $t \in Q$

iii. $i \in M$

iv. $j \in N$

v. All timber must be removed from source i in period t. Therefore,

$$\sum_{j=1}^{N} x_{ijt} = S_{it}$$

vi. All the timber from all the sources must be taken to the mill. Therefore,

$$\sum_{i=1}^{M} S_{it} = D_t$$

vii. If a depot is to be used, it has to be constructed, therefore

Period 1: $$By_{j1} \geq \sum_{i=1}^{M}\sum_{j=1}^{N} x_{ij1}$$

Period 2: $$By_{j1} + By_{j2} \geq \sum_{i=1}^{M}\sum_{j=1}^{N} x_{ij2}$$

...

Period Q: $$By_{j1} + By_{j2} + \ \ldots \ + By_{jq} \geq \sum_{i=1}^{M}\sum_{j=1}^{N} x_{ijQ}$$

viii. Conservation of flow: the volume of timber on incoming links to node r $\{K_r\}$ must be equal to the outgoing volume from node r, where $\{L_r\}$ is the set of outgoing links from node r.

$$\sum_{k \in K_r} x_{ikt} - \sum_{l \in L_r} x_{ilt} = 0$$

The variable transport cost is the cost of hauling the timber, and the variable maintenance cost is the cost of grading the road surface while the road is used for hauling (Table 4-1). The fixed cost of constructing a depot and the variable cost of transferring the timber at the depot is shown in Table 4-2.

Table 4-1: Transport and maintenance variable costs

System	Variable cost: Transport (Rand/truck/ton-km)	Variable cost: Maintenance (Rand/truck/ton-km)
Shorthaul	1.76	0.40
Longhaul	0.84	0.20
Rigid-and-drawbar	1.28	0.29

Table 4-2: Depot costs: fixed and variable

System	Fixed construction cost (Rand/link)	Variable transfer cost (Rand/ton)
Depots	100000	2.35

Link files and sale files

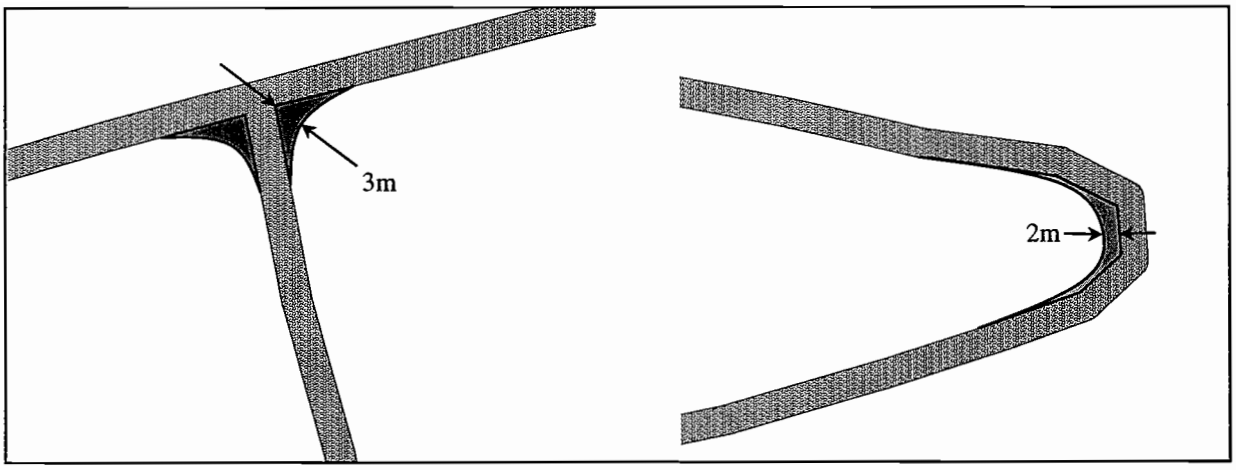
It is assumed that each road adjacent to a harvest unit will be used for hauling. The volume from a specific harvest unit is proportionally allocated to each road according the length of each road. In the situation where a road is common to two or more harvest units, the volume for the road is the sum of the proportional volumes from each harvest unit.

To prepare the data for entry into NETWORK 2000, the *Link Data* and *Sale Data* files have to be compiled. The link data contains the road segments and the sale data contains the harvest schedules. The link data file has to be modeled to recognize the constraints generated by OFFTRACK.

Conventionally, the network would be set up to use the begin- and end-points of the roads as the "from nodes" and "to nodes". This method of data entry makes it difficult to allow for constraints. This problem was overcome by taking road midpoints as the "from nodes" and "to nodes" (Figure 4-1, Figure 4-4). This made it possible to allow network links between roads that did not exceed the allowable offtracking. Roads or links that had offtracking exceeding the allowable amount were omitted from the network. The allowable offtracking depends on the type of road file. A road file can consist of one, two, or three road segments. Approximately two meters of road widening occur on the plantation roads at Elandshoogte (Schoombee[3], *pers. comm.*), and the allowable offtracking on these roads were considered to be two meters (Figure 4-2). Wherever road segments connect, there is a substantial amount of curve widening that is the result of vehicle movement from one segment to the next (Figure 4-3). The maximum allowable offtracking for the curve widening was considered to be three meters.

Three networks were created, one for each of the configurations. The Shorthaul road network is the complete set of links encompassing all the roads in the network. The Longhaul configuration had only the road links associated with the main hauling roads. The network for the Rigid-and-drawbar configuration is a subset of the complete set of links, and this subset includes only roads and road connections that did not exceed the allowable offtracking.

[3] Piet Schoombee, Harvesting Manager, Sappi Forests, Ngodwana, Mpumalanga, South Africa.



* Not drawn to scale

Figure 4-2: Allowable offtracking in curve widening

Figure 4-3: Allowable offtracking in normal curve

Example

The entire network is too large to be graphically represented, and an example of the network is used to illustrate the network formulation. The input and results from a single period network (Figure 4-4) are shown in Table 4-3, Table 4-4 and Table 4-5. The example network has one sale, one mill, and 24 nodes of which one is a depot. Instead of describing the harvest in terms of the harvest units, the harvest is described in terms of road sections and the volume assigned to each section. Each node in the network represents a road. Four types of nodes can be assigned to a road. If any timber volume is assigned to the road, the road will serve as a sale, an entry into the network, and will be assigned a six digit number starting with 1, and ending with the number of the road (e.g. 585 = 100585), as can be seen in Figure 4-4. Every road in the network can be used for Shorthaul transportation, and is assigned a number starting with 7 (e.g. 585 = 700585). Only the main routes can be used for Longhaul transportation, and only roads that lie on the main routes can be assigned a number starting with 8 (e.g. 574 = 800574). The roads that did not exceed the allowable offtracking can be used by the Rigid-and-drawbar system, and is assigned a

number starting with 9 (e.g. 585 = 900585). Timber is transferred from the Shorthaul system to the Longhaul system via a depot which is assigned a number starting with 6 (e.g. 600010). Timber has to be brought in with a Shorthaul system to the depot where it is offloaded and stored. The timber is then later loaded onto the Longhaul trucks to be taken to the mill, which has the number 999999 assigned to it. The timber would therefore enter the network through a road that has any volume assigned it. In the example network, road number 585 was selected. The timber can take one of three routes. The first route is directly to the mill with the Rigid-and-drawbar system, illustrated on the right in Figure 4-4 (100585 → 900585 → ...). The second route is illustrated on the far left of Figure 4-4; the Shorthaul system taking timber directly to the mill (100585 → 700585 → 700648 → 999999). The third route is illustrated on the left of Figure 4-4; the Longhaul system. Timber is taken with a Shorthaul truck to a depot where it is transshipped to a Longhaul truck for transport to the mill (100585 → 700585 → 600010 → ...). The minimum total cost found is R5181.42.

Table 4-3: Network input example (link data)

Link Identifier		Length	Road cost:	Haul cost: Total
(From)	(To)	(meter)	Total (Rand/ton)	(Rand/km/ton)
100585	700585	0	0	0
100585	900585	0	0	0
700585	700648	902	4510	1.08
700585	700574	60	300	0.07
700585	700616	116	580	0.14
700585	700614	112	560	0.13
700648	700585	902	4510	1.08
700648	700574	869	4345	1.04
700648	600010	855	4275	1.03
700648	999999	74171	370855	89.01
700574	700585	60	300	0.07
700574	700648	869	4345	1.04
700574	600010	14	70	0.02
700574	700572	23	115	0.03
700574	701158	255	1275	0.31
700572	700574	23	115	0.03
700572	700571	539	2695	0.65
700572	700587	104	520	0.12

700572	701158	251	1255	0.30
701158	700574	255	1275	0.31
701158	700572	251	1255	0.30
700616	700585	116	580	0.14
700614	700585	112	560	0.13
700571	700572	539	2695	0.65
700571	999999	15531	77655	18.64
700587	700572	104	520	0.12
600010	600011	0	100000	2.35
600011	800648	855	4275	0.49
600011	800574	14	70	0.01
800648	800574	869	4345	0.50
800648	999999	74171	370855	42.28
800574	800648	869	4345	0.50
800574	800572	23	115	0.01
800572	800574	12	60	0.01
800572	800571	539	2695	0.31
800571	800572	539	2695	0.31
800571	999999	15531	77655	8.85
900585	900574	60	300	0.05
900585	900616	116	580	0.09
900585	900614	112	560	0.09
900574	900585	60	300	0.05
900574	901158	255	1275	0.20
900574	900572	23	115	0.02
900574	900648	869	4345	0.70
900648	999999	74171	370855	59.34
900648	900574	869	4345	0.70
900572	900574	23	115	0.02
900572	900571	539	2695	0.43
900572	900587	104	520	0.08
901158	900574	255	1275	0.20
900616	900585	116	580	0.09
900614	900585	112	560	0.09
900571	900572	539	2695	0.43
900571	999999	15531	77655	12.42
900587	900572	104	520	0.08

Table 4-4: Network input example (sale data)

Origin node	Destination node	Harvest Volume	Year
100585	999999	248.58	2006

Table 4-5: Solution path for example.

Solution path
100585 → 700585 → 600010 → 600011 → 800574 → 800572 → 800571 → 999999

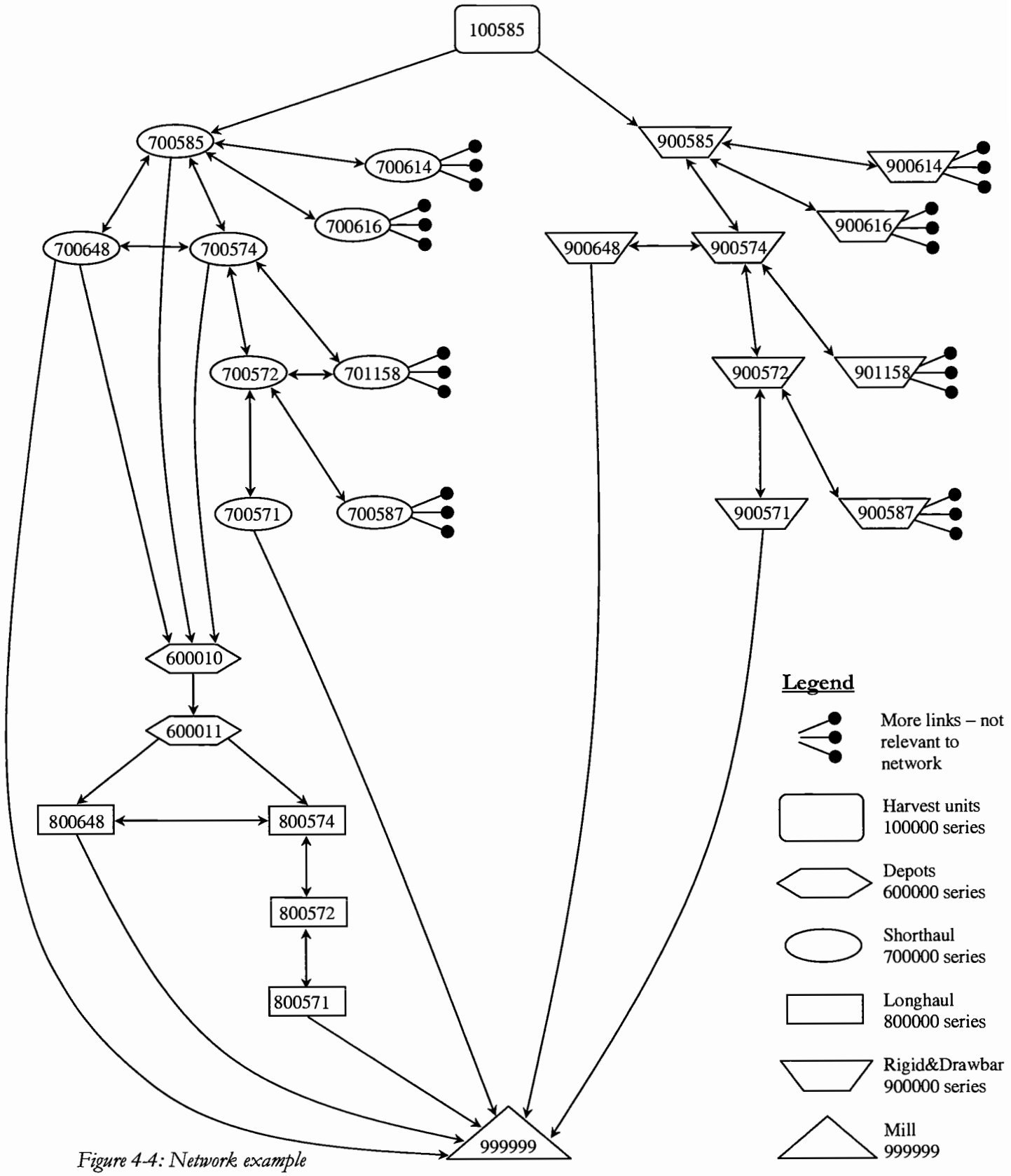


Figure 4-4: Network example

Elandshoogte Network

The total network has 1060 sales, 7 depots, and one mill. Of the total 9261 nodes in the network, 232 are dedicated to the Longhaul configuration, 4162 to the Rigid-and-drawbar configuration, and 5099 to the Shorthaul configuration.

The current inflation rate in South Africa is approximately 7%, and the nominal interest rate is about 11%. For the purpose of this study a 4% real rate of return was therefore used.

Several iterations were done with the systems on their own, and in conjunction with other systems.

Iteration 1: Shorthaul system

The Shorthaul configuration was entered into the network on its own, assuming the trucks can take the timber directly from the harvesting units to the mill. The depots were omitted from this network. The result of this iteration is compared against the Rigid-and-drawbar configuration in iteration 2.

Iteration 2: Rigid-and-drawbar system

The Rigid-and-drawbar configuration was entered into the network on its own. The result of this iteration is compared against the Shorthaul configuration in iteration 1.

Iteration 3: Shorthaul and Longhaul system

The Shorthaul and Longhaul configurations were entered into the network together. The result of this iteration is compared against the result of iteration 4, using all three systems together.

Iteration 4: Shorthaul, Longhaul and Rigid-and-drawbar systems

All three configurations were entered into the network. The result of this iteration is compared against the result of iteration 3.

Iterations 5 – 11: Shorthaul, Longhaul, and Rigid-and-drawbar – closing one depot at a time

One depot was removed in each iteration to determine how the removal of each depot would affect the total cost.

5. RESULTS

The best solution is defined as the network path providing the lowest cost solution. It is not known how close the best solution is to the exact lowest solution. In order to get an optimal solution, the problem would have to be analyzed using a mixed integer linear programming model to verify that the optimal solution has been found.

Four network configurations were analyzed separately with Network 2000. The first network configuration consists of only the Shorthaul system and the second network configuration consists of the Rigid-and-drawbar system. The third network configuration consists of the Shorthaul and the Longhaul system, and the fourth network configuration of all three transport systems.

5.1 NETWORK ANALYSIS

Table 5-1: Results of network analysis – comparing systems (iterations 1 through 4)

	Total discounted variable cost		Total discounted fixed cost		Total discounted variable and fixed cost	
	Rand	Rand/ton	Rand	Rand/ton	Rand	Rand/ton
Iteration 1: Shorthaul configuration	58,427,860.76	41.88	0.00	0.00	58,427,860.76	41.88
Iteration 2: Rigid-and-drawbar configuration	43,259,417.39	31.35	0.00	0.00	43,259,417.39	31.35
Iteration 3: Shorthaul and Longhaul configuration[1]	34,328,847.51	24.61	540,742.15	0.39	34,869,589.66	25.00
Iteration 4: Shorthaul, Longhaul and Rigid-and-drawbar configurations	34,225,578.81	24.53	619,773.60	0.44	34,845,352.41	24.98

[1]Eleven harvesting units could not be accessed with the Rigid-and-drawbar configuration, and was therefore omitted from the sale file.

Table 5-2: Results of network analysis – comparing removal of depots (iterations 5 through 11)

Depot removal	Total discounted variable cost	Total discounted fixed cost	TOTAL COST
Iteration 5: Depot 600010	24.84	0.34	25.17
Iteration 6: Depot 600020	25.08	0.26	25.34
Iteration 7: Depot 600030	24.89	0.32	25.21
Iteration 8: Depot 600040	25.07	0.26	25.33
Iteration 9: Depot 600050	24.59	0.39	24.98
Iteration 10: Depot 600060	25.06	0.39	25.45
Iteration 11: Depot 600070	24.59	0.39	24.98

Iteration 1: Shorthaul system

The paths in (Figure 5-1) are for all harvesting units during all periods, even though the routes from only six harvesting units are shown. The Shorthaul system was the most expensive of all the systems to operate, as is to be expected.

Iteration 2: Rigid-and-drawbar system

The paths in (Figure 5-1) are for all harvesting units during all periods, even though the routes from only six harvesting units are shown. The Rigid-and-drawbar configuration has a relatively low cost, but it still have to be used in conjunction with the Shorthaul system, since eleven of the harvesting units could not be reached with this configuration.

Iteration 3: Shorthaul and Longhaul system

The paths in (Figure 5-2) are for all harvesting units during all periods, even though the routes from only six harvesting units are shown. Six of the seven depots were activated in iteration 3, with depot 60070 excluded.

Iteration 4: Shorthaul, Longhaul and Rigid-and-drawbar systems

The paths in (Figure 5-3) are for all harvesting units during all periods, even though the routes from only six harvesting units are shown. All seven depots were activated in iteration 4. Where the Rigid-and-drawbar configuration was used in conjunction with all the other configurations, volumes for this configuration was concentrated in one area (Figure 5-4).

Iterations 5 – 11: Shorthaul, Longhaul, and Rigid-and-drawbar – closing one depot at a time

The results of removing one depot in each iteration to determine the effect on total cost can be seen in Table 5-2.

5.2 DISCUSSION:

Iterations 1 and 2 are compared to determine the effect of removing the links that exceed the allowable offtracking. Since both iterations attempt to take the timber to the mill at the least cost, any change in the path can be ascribed to the differences in the links. Figure 5-1 shows the routes taken from six harvesting units. It can be seen that The Rigid-and-drawbar configuration takes different routes than the Shorthaul configuration. This is the result of removing links from the Rigid-and-drawbar network that exceed the allowable offtracking.

Iterations 3 and 4 are compared to determine the effect of adding the rigid-and-drawbar configuration to the Shorthaul-Longhaul system used. There is a marginal improvement in cost from R25.00 to R24.98 (Table 5-1). It can be seen in Figure 5-4 that only a small section of the total harvest area is serviced by the Rigid-and-drawbar system. Only 23,189 tons out of the total 1,396,808 tons were transported with the Rigid-and-drawbar system.

Iterations 5 through 11 are compared to determine how the removal of each depot would affect the total cost (Table 5-2). Closing depot 600050 or 600070 resulted in the largest reduction in costs. Both costs were close to the lowest solution found with all depots open.

Even though curve widening was not considered in this study, it could have an effect on the selection of vehicle configurations. By widening all the roads that are currently three and four meters wide by one meter, 330 links are opened up to the Rigid-and-drawbar system. Seven more harvesting units are also accessible to the Rigid-and-drawbar system, leaving only four that cannot be reached with this system. There was no cost improvement when all three systems were considered simultaneously after road widening. In an iteration done with only the Rigid-and-drawbar system, the cost did go down from R31.35/ton to R30.66/ton. In an iteration done with all systems, the cost remained the same.

The lowest costs were obtained with all or most of the depots open. This can be attributed to the comparatively low rate of the Longhaul configuration. Widening the roads to allow the Longhaul configuration to move deeper into the plantation would lower the distances traveled with the Shorthaul configuration and increase the distances traveled with the Longhaul configurations. This could lead to significant cost reductions, and could be the topic of further studies.

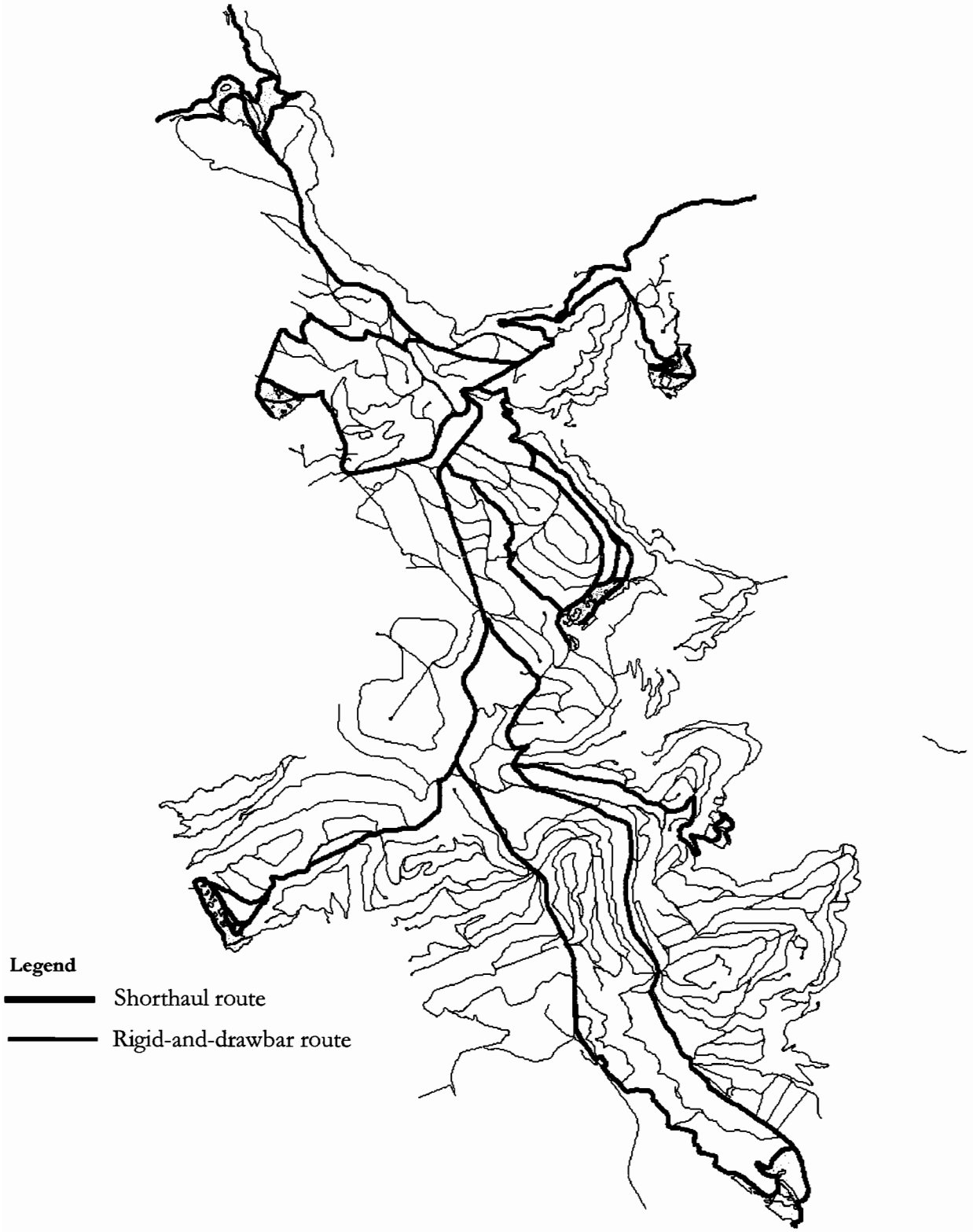


Figure 5-1: Results of iteration 1 and 2 – Routes taken by Shorthaul and Rigid-and-drawbar configurations (direct to mill) (paths from six selected harvest units shown)

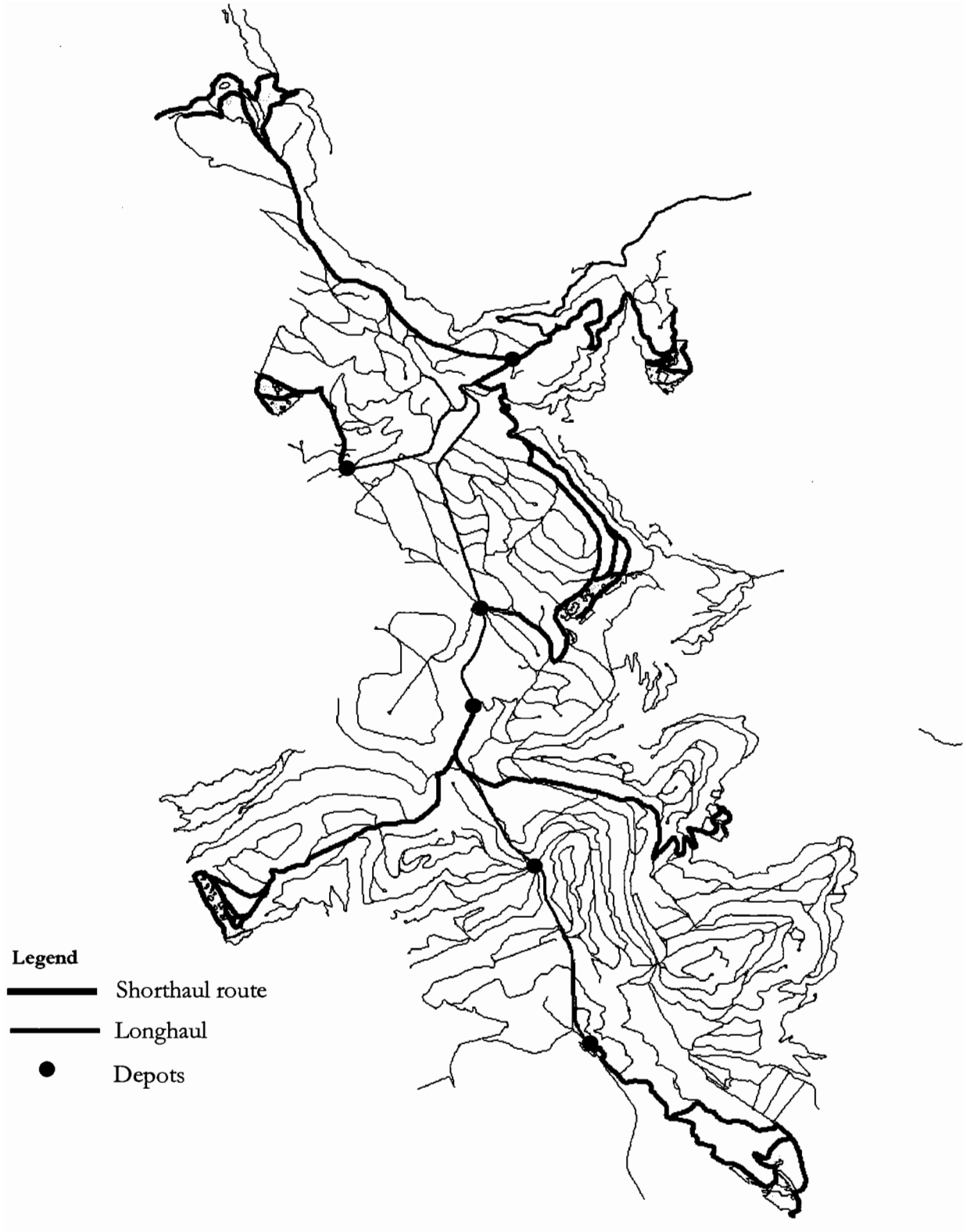


Figure 5-2: Results of iteration 3 – Path taken by Shorthaul and Longhaul configurations (paths from six selected harvest units shown)

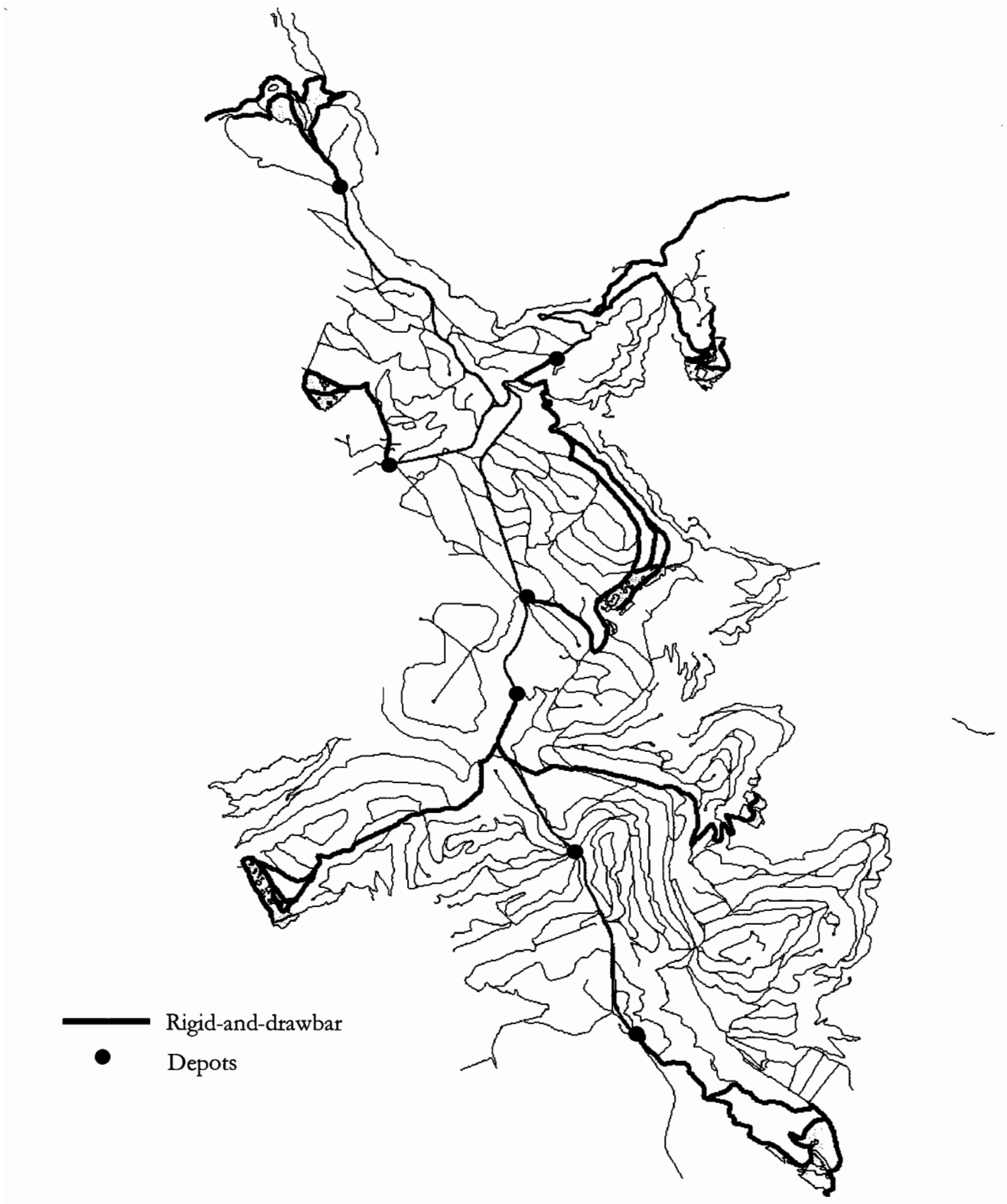


Figure 5-3: Results of iteration 4 – Path taken by Shorthaul, Longhaul and Rigid-and-drawbar configurations (paths from six selected harvest units shown)

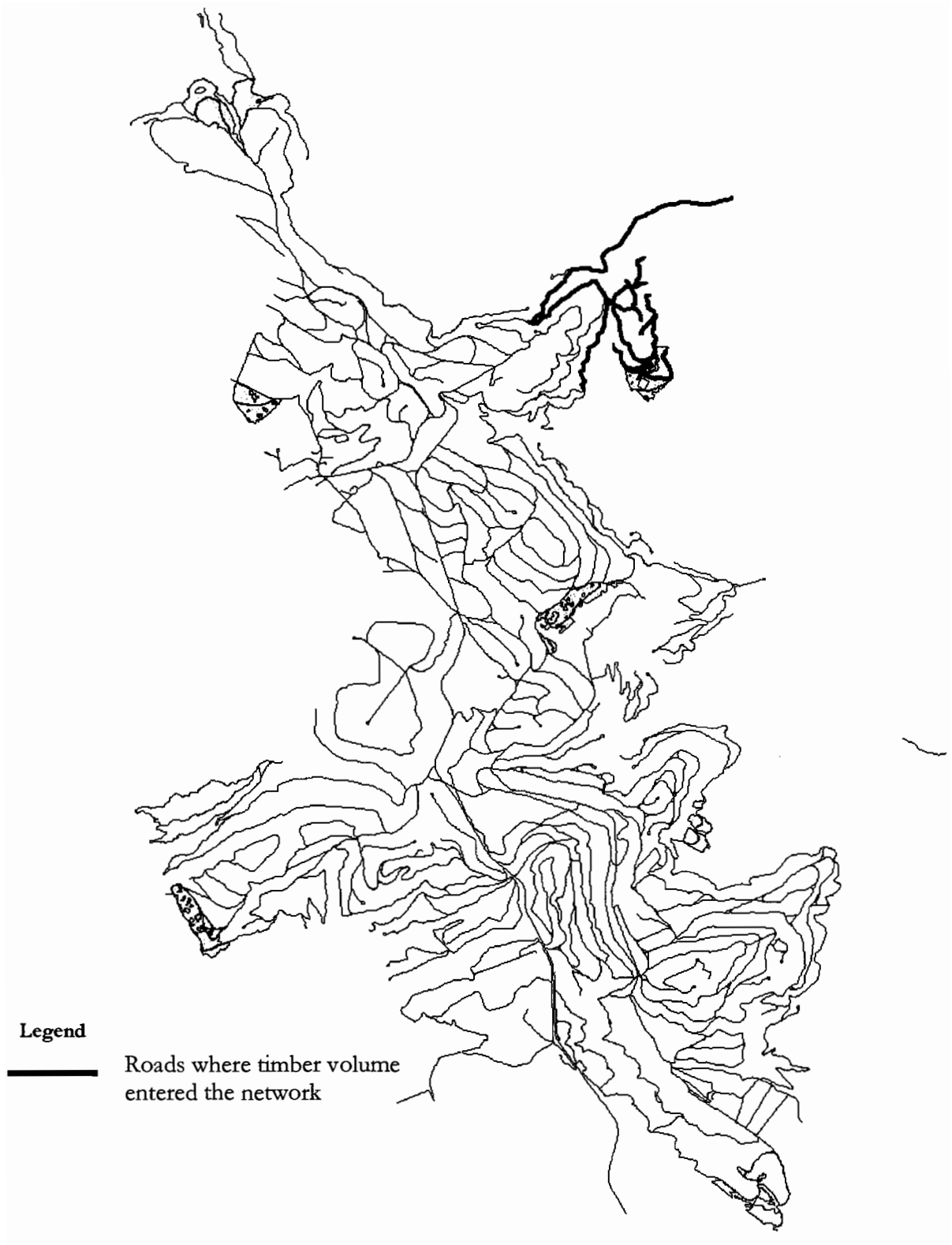


Figure 5-4: Rigid-and-drawbar harvesting sources (al paths shown)

6. CONCLUSIONS

The use of a geographical information system (GIS) road data set from the Elandshoogte plantation as a basis for determining the accessibility of plantation roads proved to be successful in this application. The GIS data set provided sufficient information that could be analyzed to determine vehicle accessibility. The computer simulation program OFFTRACK proved to be a useful method for this application.

Using NETWORK 2000 to determine the optimal road system for the three given vehicle configuration by minimizing the total cost was also successful. Repeated runs gave consistent results that provided both costs and routes consistent with actual transportation costs and routes. A sensitivity analysis was done by removing depots manually from the network. None of these analyses could improve on the NETWORK 2000 solution.

The accuracy of the GIS data sets were not considered in this study, and could be the focus of further studies. Field verification of the results also falls outside the context of this paper.

7. REFERENCE

CAIN, C. and LANGDON, J.A., 1982. "A guide for Determining Minimum Road Width on Curves for Single-Lane Forest Roads". Engineering Field Notes, Volume 14, USDA Forest Service, April-June.

CHUNG, W. and SESSIONS, J., 2000. NETWORK 2000, a program for optimizing large fixed and variable cost transportation problems. In Proceedings of the Eighth Symposium on Systems Analysis in Forest Resources. Edited by Arthaud, G. J. Society of American Foresters. Aspen, Colorado. (in press)

ERKERT, T.W., 1989. Computer simulation of offtracking of struck and trailer combinations using forest roads. Master of Science paper. Department of Civil Engineering, Oregon State University, Corvallis, Oregon 97331. 89 pp.

KIRBY, M., 1981. Timber Transport Examples, USDA Forest Service, Pacific Southwest Forest Range experiment station, Berkley, California.

MOORE, T.L., 1987. An Empirical Evaluation of Three Network Analysis Programs used in the USDA Forest Service. Master of Science paper. Department of Civil Engineering, Oregon State University, Corvallis, Oregon. 102 pp.

SESSIONS, J., 1985. A Heuristic Algorithm for the Solution of the Variable and Fixed Cost Transportation Problem. In Proceedings of the 1985 Symposium on Systems Analysis in Forest Resources. Edited by Dress and Field. Society of American Foresters. Athens, Georgia. December 9 – 11. pp. 324 – 336.

SESSIONS, J. and SESSIONS, J.B., 1988. Network Analysis Using Microcomputers For Logging Planning. College of Forestry, Oregon State University, Corvallis, Oregon.

SHEN, Z., 1988. Log Truck Scheduling by Network programming. Master of Forestry paper. Department of Forest Engineering, Oregon State University, Corvallis, Oregon 97331. 117 pp.

"Turning Ability and Off Tracking", SAE Recommended Practice, Society of Automotive Engineers. SAE J695b, SAE Handbook, 1964, pp 875 – 880.

WONG, P., 1981. An Empirical evaluation of the Proration Option of MINCOST Network Program. USDA Forest Service Engineering Field notes. Washington D.C.

8. APPENDIXES

APPENDIX A

```
Option Base 1
Private Sub cmdCALCCURV_Click()

' COORDINATES.VBP recorded 6/7/2000 by Francois Oberholzer

'mmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmm
' | PROGRAM TO CALCULATE ROAD CURVES AND FLEET ANGLES
' | Francois Oberholzer
'mmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmm

Dim DATA1 As String     ' road_cr2.dxf
Dim DATA2 As String     ' fw_line*.txt (forward line input)
Dim DATA3 As String     ' fw_ang*.txt (forward line angle)
Dim DATA4 As String     ' Weed*.txt (backward line input)
Dim DATA5 As String     ' fulldata
Dim DATA6 As String     ' road* file (to be read in Offtrack)
Dim DATA7 As String     ' ENDS file: record first and last coordinates of each file
Dim DATA8 As String     ' COMBCURV* file: X and Y coordinates of intersection
Dim DATA9 As String     ' COMCVANG* file: The angle difference at intersections
Dim DATA10 As String    ' Tempcurv: contain all the coordinates of the intersections in a format that can be     ' ' generated
                        into a shapefile (Arc/Info)
Dim DATA11 As String    ' XScurves: file that contains the id of the roads in each COBCURV file
Dim DATA12 As String    ' Tempcurv2
Dim DATA13 As String    ' Problem_R (Problem Roads)
Dim DATA14 As String    ' FULLDATA Back_up
Dim DATA15 As String    ' RD backup

Dim AA As String
Dim BB As String
Dim FIRSTX As Double
Dim FIRSTY As Double
Dim countvertex(10000) As Integer
Dim X_first(10000) As Double
Dim X_last(10000) As Double
Dim Y_first(10000) As Double
Dim Y_last(10000) As Double
Dim cc(10000) As Integer
Dim link_counter(10000) As Integer
Dim critangle As Double
Dim ccounter(10000) As Integer
Dim DATA(200000, 14) As Double
Dim vec_count(10000) As Integer
Dim vector_count(10000) As Integer
Dim new_vector_count As Integer
Dim vector_in_road(10000) As Integer
Dim ENDS(4000, 4) As Double
Dim countintersect(10000) As Integer
Dim afterweed_vector(10000) As Integer
Dim TEMPORARRY(30) As Integer
Dim XScurv(10000, 4) As Integer

'KILL all previous files first
'Kill "k:\Oberholf\HSU\mf\output\curves_output\rd*"
'Kill "k:\Oberholf\HSU\mf\output\curves_output\fulldata*"
'Kill "k:\Oberholf\HSU\mf\output\curves_output_backup\rd*"
'Kill "k:\Oberholf\HSU\mf\output\curves_output_backup\fulldata*"

'=======================================================================
' STEP 1: WRITING COORDINATES TO ARC-INDIVIDUAL FILES
' This is where I read in the coordinates. The file ROAD_CR2.DXF was
' "Ungenerated" from the ROADS coverage using the function ARCDXF. In this
' section the coordinates for each road arc are written to an individual road
' file (1177 files in total for Elandshoogte).
' These files are called FW_LINE1177, which means the coordinates when looking
' forward from the start of the arc.
'=======================================================================
DATA1 = "k:\oberholf\hsu\mf\vb_programs\curves\road_cr2.dxf" 'road_cr2.DXF, dxf_3.dxf
```

```
Open DATA1 For Input As #1
Do While Not EOF(1)
   Input #1, AA
   If RTrim$(LTrim$(AA)) = "POLYLINE" Then
      If count_roads > 0 Then
         Close #2
         VCount = o
      End If
      count_roads = count_roads + 1
      DATA2 = "k:\Oberholf\HSU\mf\output\curves_output\fw_line" + LTrim$(RTrim$(Str$(count_roads)))
      Open DATA2 For Output As #2
   ElseIf AA = "VERTEX" Then
      For M = 1 To 3
         Input #1, BB
      Next M
      Input #1, X
      Input #1, AA
      Input #1, Y
      Print #2, X, Y
      VCount = VCount + 1
   End If
Loop
Close #1
Close #2

'=============================================================================
' STEP 2: CALCULATING DISTANCES AND ANGLES - FORWARD (ROADS)
' Calculate the distances and angles in a forward direction
'=============================================================================
For i = 1 To count_roads
   DATA2 = "k:\Oberholf\HSU\mf\output\curves_output\fw_line" + LTrim$(RTrim$(Str$(i)))
   DATA3 = "k:\Oberholf\HSU\mf\output\curves_output\fw_ang" + LTrim$(RTrim$(Str$(i)))
   FIRSTX = 0
   FIRSTY = 0
   SECONDX = 0
   SECONDY = 0
   countvertex(i) = 0
   LENGTHOFLINE = 0
   SQRTLENGTHOFLINE = 0
   COSALPHA = 0
   ANGLES = 0
   AZIMUTH = 0
   HYP = 0
   ADJ = 0
   X_first(i) = 0: X_last(i) = 0
   Y_first(i) = 0: Y_last(i) = 0

   Open DATA2 For Input As #2
   Open DATA3 For Output As #3
   Do While Not EOF(2)
      SECONDX = FIRSTX
      SECONDY = FIRSTY
      Input #2, FIRSTX, FIRSTY
      countvertex(i) = countvertex(i) + 1
      If countvertex(i) = 1 Then
         Print #3, FIRSTX, FIRSTY, LENGTHOFLINE, ANGLES
         SECONDX = FIRSTX
         SECONDY = FIRSTY
         Input #2, FIRSTX, FIRSTY
         countvertex(i) = countvertex(i) + 1
         X_first(i) = SECONDX 'Use this to check for circular route
         Y_first(i) = SECONDY 'Use this to check for circular route
      End If

      ' The angles start at the second node and look back at the first.
      SQRTLENGTHOFLINE = Abs(SECONDX - FIRSTX) ^ 2 + Abs(SECONDY - FIRSTY) ^ 2
      LENGTHOFLINE = (SQRTLENGTHOFLINE) ^ 0.5
      If (FIRSTX = SECONDX) And (SECONDY > FIRSTY) Then
         AZIMUTH = 180
      End If
```

```
    If (FIRSTY = SECONDY) And (SECONDX > FIRSTX) Then
      AZIMUTH = 270
    End If
    If (FIRSTX = SECONDX) And (SECONDY < FIRSTY) Then
      AZIMUTH = 0
    End If
    If (FIRSTY = SECONDY) And (SECONDX < FIRSTX) Then
      AZIMUTH = 90
    End If
    If (SECONDX < FIRSTX) And (SECONDY < FIRSTY) Then
      SQRHYP = Abs(SECONDX - FIRSTX) ^ 2 + Abs(SECONDY - FIRSTY) ^ 2
      HYP = Sqr(SQRHYP)
      ADJ = FIRSTX - SECONDX
      COSALPHA = ADJ / HYP
      ANGLES = (1.570796 - Atn(COSALPHA / Sqr(1 - COSALPHA * COSALPHA))) * 180 / 3.1415926538979
      AZIMUTH = 90 - ANGLES
    End If
    If (SECONDX < FIRSTX) And (SECONDY > FIRSTY) Then
      SQRHYP = (Abs(SECONDX - FIRSTX)) ^ 2 + (Abs(SECONDY - FIRSTY)) ^ 2
      HYP = Sqr(SQRHYP)
      ADJ = FIRSTX - SECONDX
      COSALPHA = ADJ / HYP
      ANGLES = (1.570796 - Atn(COSALPHA / Sqr(1 - COSALPHA * COSALPHA))) * 180 / 3.1415926538979
      AZIMUTH = 90 + ANGLES
    End If
    If (SECONDX > FIRSTX) And (SECONDY > FIRSTY) Then
      SQRHYP = Abs(SECONDX - FIRSTX) ^ 2 + Abs(SECONDY - FIRSTY) ^ 2
      HYP = Sqr(SQRHYP)
      ADJ = SECONDX - FIRSTX
      COSALPHA = ADJ / HYP
      ANGLES = (1.570796 - Atn(COSALPHA / Sqr(1 - COSALPHA * COSALPHA))) * 180 / 3.1415926538979
      AZIMUTH = 270 - ANGLES
    End If
    If (SECONDX > FIRSTX) And (SECONDY < FIRSTY) Then
      SQRHYP = Abs(SECONDX - FIRSTX) ^ 2 + Abs(SECONDY - FIRSTY) ^ 2
      HYP = Sqr(SQRHYP)
      ADJ = SECONDX - FIRSTX
      COSALPHA = ADJ / HYP
      ANGLES = (1.570796 - Atn(COSALPHA / Sqr(1 - COSALPHA * COSALPHA))) * 180 / 3.1415926538979
      AZIMUTH = 270 + ANGLES
    End If
    Print #3, FIRSTX, FIRSTY, LENGTHOFLINE, AZIMUTH
  Loop
  Close #2
  Close #3
Next i

'=============================================================================
' STEP 3: WEED OUT ALL THE SHORT VECTORS
' In this section I will weed out all the short vectors that throw the data of.
' All the sections < 5 meters will be weeded out
'=============================================================================
' Put all the data in an array first
' ----------------------------------
count_vectors = 0
For i = 1 To count_roads
  DATA3 = "k:\Oberholf\HSU\mf\output\curves_output\fw_ang" + LTrim$(RTrim$(Str$(i)))
  Open DATA3 For Input As #3
  Do While Not EOF(3)
    Input #3, X, Y, dist, angl
    vec_count(i) = vec_count(i) + 1
    count_vectors = count_vectors + 1
    DATA(count_vectors, 1) = i
    DATA(count_vectors, 2) = X
    DATA(count_vectors, 3) = Y
    DATA(count_vectors, 4) = dist
    DATA(count_vectors, 5) = angl
  Loop
  Close #3
Next i
```

```
' Now look for all the road vectors that are shorter than 2 meters. Then look at
' the angle difference with the previous vector, and the next vector. Choose the
' angle difference with the least change, and the connect the current vector with
' the other associated vector.
' If the road vectors are between 2 and 5 meters, look for an angle difference of
' less than 2 degrees, and repeat the process described above.
' ------------------------------------------------------------------------------
DATA7 = "k:\Oberholf\HSU\mf\output\curves_output\ENDS.txt"
Open DATA7 For Output As #7
For i = 1 To count_roads
   For j = 1 To count_vectors
      If DATA(j, 1) = i Then
         vector_count(i) = vector_count(i) + 1
         'Record the first and the last COORDINATES of each arc here.
         'Each arc START with 10 and END with 20
         If vector_count(i) = 1 Then
            Write #7, i, 10, DATA(j, 2), DATA(j, 3)
         ElseIf vector_count(i) = vec_count(i) Then
            Write #7, i, 20, DATA(j, 2), DATA(j, 3)
         End If
         'Now get back to the weeding
         If vector_count(i) >= 2 And vector_count(i) < vec_count(i) Then
            If DATA(j, 4) < 2 Then                    'Check here for VECTORS < 2m
               If DATA((j - 1), 2) <> 0 Then
                  If Abs((DATA(j, 5)) - DATA((j - 1), 5)) < 300 Then
                     back_difference = Abs(DATA(j, 5) - DATA((j - 1), 5))
                  End If
                  If Abs(DATA(j, 5) - DATA((j + 1), 5)) < 300 Then
                     front_difference = Abs(DATA(j, 5) - DATA((j + 1), 5))
                  End If
               ElseIf DATA((j - 1), 2) = 0 Then
                  If DATA((j - 2), 2) <> 0 Then
                     If Abs((DATA(j, 5)) - DATA((j - 2), 5)) < 300 Then
                        back_difference = Abs(DATA(j, 5) - DATA((j - 2), 5))
                     End If
                     If Abs(DATA(j, 5) - DATA((j + 1), 5)) < 300 Then
                        front_difference = Abs(DATA(j, 5) - DATA((j + 1), 5))
                     End If
                  ElseIf DATA((j - 2), 2) = 0 Then
                     If Abs((DATA(j, 5)) - DATA((j - 3), 5)) < 300 Then
                        back_difference = Abs(DATA(j, 5) - DATA((j - 3), 5))
                     End If
                     If Abs(DATA(j, 5) - DATA((j + 1), 5)) < 300 Then
                        front_difference = Abs(DATA(j, 5) - DATA((j + 1), 5))
                     End If
                  End If
               End If

               If DATA((j - 1), 2) <> 0 Then
                  If Abs((DATA(j, 5)) - DATA((j - 1), 5)) > 300 Then
                     If DATA(j, 5) > 300 Then
                        back_difference = -((360 - DATA(j, 5)) + DATA((j - 1), 5))
                     ElseIf DATA((j - 1), 5) > 300 Then
                        back_difference = DATA(j, 5) + (360 - DATA(j - 1, 5))
                     End If
                  End If
               ElseIf DATA((j - 1), 2) = 0 Then
                  If DATA((j - 2), 2) <> 0 Then
                     If Abs((DATA(j, 5)) - DATA((j - 2), 5)) > 300 Then
                        If DATA(j, 5) > 300 Then
                           back_difference = -((360 - DATA(j, 5)) + DATA((j - 2), 5))
                        ElseIf DATA((j - 2), 5) > 300 Then
                           back_difference = DATA(j, 5) + (360 - DATA((j - 2), 5))
                        End If
                     End If
                  ElseIf DATA((j - 2), 2) = 0 Then
                     If Abs((DATA(j, 5)) - DATA((j - 3), 5)) > 300 Then
                        If DATA(j, 5) > 300 Then
                           back_difference = -((360 - DATA(j, 5)) + DATA((j - 3), 5))
```

```
        ElseIf DATA((j - 3), 5) > 300 Then
          back_difference = DATA(j, 5) + (360 - DATA((j - 3), 5))
        End If
      End If
    End If
  End If

  If Abs((DATA(j, 5)) - DATA((j + 1), 5)) > 300 Then
    If DATA(j, 5) > 300 Then
      front_difference = -((360 - DATA(j, 5)) + DATA((j + 1), 5))
    ElseIf DATA((j - 1), 5) > 300 Then
      front_difference = DATA(j, 5) + (360 - DATA(j + 1, 5))
    End If
  End If

  If Abs(back_difference) < Abs(front_difference) Then
    ' Use points (j-2) and (j)
    ' The angles start at the second node and look back at the first.
    If DATA((j - 2), 2) <> 0 Then
      FIRSTX = DATA(j, 2):      FIRSTY = DATA(j, 3)
      SECONDX = DATA((j - 2), 2): SECONDY = DATA((j - 2), 3)
    ElseIf DATA((j - 2), 2) = 0 Then
      If DATA((j - 3), 2) <> 0 Then
        FIRSTX = DATA(j, 2):      FIRSTY = DATA(j, 3)
        SECONDX = DATA((j - 3), 2): SECONDY = DATA((j - 3), 3)
      ElseIf DATA((j - 3), 2) = 0 Then
        FIRSTX = DATA(j, 2):      FIRSTY = DATA(j, 3)
        SECONDX = DATA((j - 4), 2): SECONDY = DATA((j - 4), 3)
      End If
    End If
    SQRTLENGTHOFLINE = Abs(SECONDX - FIRSTX) ^ 2 + Abs(SECONDY - FIRSTY) ^ 2
    LENGTHOFLINE = (SQRTLENGTHOFLINE) ^ 0.5
    If (FIRSTX = SECONDX) And (SECONDY > FIRSTY) Then
      AZIMUTH = 180
    End If
    If (FIRSTY = SECONDY) And (SECONDX > FIRSTX) Then
      AZIMUTH = 270
    End If
    If (FIRSTX = SECONDX) And (SECONDY < FIRSTY) Then
      AZIMUTH = 0
    End If
    If (FIRSTY = SECONDY) And (SECONDX < FIRSTX) Then
      AZIMUTH = 90
    End If
    If (SECONDX < FIRSTX) And (SECONDY < FIRSTY) Then
      SQRHYP = Abs(SECONDX - FIRSTX) ^ 2 + Abs(SECONDY - FIRSTY) ^ 2
      HYP = Sqr(SQRHYP)
      ADJ = FIRSTX - SECONDX
      COSALPHA = ADJ / HYP
      ANGLES = (1.570796 - Atn(COSALPHA / Sqr(1 - COSALPHA * COSALPHA))) * 180 / 3.1415926538979
      AZIMUTH = 90 - ANGLES
    End If
    If (SECONDX < FIRSTX) And (SECONDY > FIRSTY) Then
      SQRHYP = (Abs(SECONDX - FIRSTX)) ^ 2 + (Abs(SECONDY - FIRSTY)) ^ 2
      HYP = Sqr(SQRHYP)
      ADJ = FIRSTX - SECONDX
      COSALPHA = ADJ / HYP
      ANGLES = (1.570796 - Atn(COSALPHA / Sqr(1 - COSALPHA * COSALPHA))) * 180 / 3.1415926538979
      AZIMUTH = 90 + ANGLES
    End If
    If (SECONDX > FIRSTX) And (SECONDY > FIRSTY) Then
      SQRHYP = Abs(SECONDX - FIRSTX) ^ 2 + Abs(SECONDY - FIRSTY) ^ 2
      HYP = Sqr(SQRHYP)
      ADJ = SECONDX - FIRSTX
      COSALPHA = ADJ / HYP
      ANGLES = (1.570796 - Atn(COSALPHA / Sqr(1 - COSALPHA * COSALPHA))) * 180 / 3.1415926538979
      AZIMUTH = 270 - ANGLES
    End If
    If (SECONDX > FIRSTX) And (SECONDY < FIRSTY) Then
      SQRHYP = Abs(SECONDX - FIRSTX) ^ 2 + Abs(SECONDY - FIRSTY) ^ 2
```

```
    HYP = Sqr(SQRHYP)
    ADJ = SECONDX - FIRSTX
    COSALPHA = ADJ / HYP
    ANGLES = (1.570796 - Atn(COSALPHA / Sqr(1 - COSALPHA * COSALPHA))) * 180 / 3.1415926538979
    AZIMUTH = 270 + ANGLES
  End If
  DATA(j, 4) = LENGTHOFLINE
  DATA(j, 5) = AZIMUTH
  For pp = 1 To 7
    DATA((j - 1), pp) = 0
  Next pp
ElseIf Abs(back_difference) >= Abs(front_difference) And j < count_vectors Then
  ' Use points (j-1) and (j+1)
  ' The angles start at the second node and look back at the first.
  If DATA((j - 1), 2) <> 0 Then
    FIRSTX = DATA((j + 1), 2):  FIRSTY = DATA((j + 1), 3)
    SECONDX = DATA((j - 1), 2): SECONDY = DATA((j - 1), 3)
  ElseIf DATA((j - 1), 2) = 0 Then
    If DATA((j - 2), 2) <> 0 Then
      FIRSTX = DATA((j + 1), 2):    FIRSTY = DATA((j + 1), 3)
      SECONDX = DATA((j - 2), 2): SECONDY = DATA((j - 2), 3)
    ElseIf DATA((j - 2), 2) = 0 Then
      FIRSTX = DATA((j + 1), 2):    FIRSTY = DATA((j + 1), 3)
      SECONDX = DATA((j - 3), 2): SECONDY = DATA((j - 3), 3)
    End If
  End If
  SQRTLENGTHOFLINE = Abs(SECONDX - FIRSTX) ^ 2 + Abs(SECONDY - FIRSTY) ^ 2
  LENGTHOFLINE = (SQRTLENGTHOFLINE) ^ 0.5
  If (FIRSTX = SECONDX) And (SECONDY > FIRSTY) Then
    AZIMUTH = 180
  End If
  If (FIRSTY = SECONDY) And (SECONDX > FIRSTX) Then
    AZIMUTH = 270
  End If
  If (FIRSTX = SECONDX) And (SECONDY < FIRSTY) Then
    AZIMUTH = 0
  End If
  If (FIRSTY = SECONDY) And (SECONDX < FIRSTX) Then
    AZIMUTH = 90
  End If
  If (SECONDX < FIRSTX) And (SECONDY < FIRSTY) Then
    SQRHYP = Abs(SECONDX - FIRSTX) ^ 2 + Abs(SECONDY - FIRSTY) ^ 2
    HYP = Sqr(SQRHYP)
    ADJ = FIRSTX - SECONDX
    COSALPHA = ADJ / HYP
    ANGLES = (1.570796 - Atn(COSALPHA / Sqr(1 - COSALPHA * COSALPHA))) * 180 / 3.1415926538979
    AZIMUTH = 90 - ANGLES
  End If
  If (SECONDX < FIRSTX) And (SECONDY > FIRSTY) Then
    SQRHYP = (Abs(SECONDX - FIRSTX)) ^ 2 + (Abs(SECONDY - FIRSTY)) ^ 2
    HYP = Sqr(SQRHYP)
    ADJ = FIRSTX - SECONDX
    COSALPHA = ADJ / HYP
    ANGLES = (1.570796 - Atn(COSALPHA / Sqr(1 - COSALPHA * COSALPHA))) * 180 / 3.1415926538979
    AZIMUTH = 90 + ANGLES
  End If
  If (SECONDX > FIRSTX) And (SECONDY > FIRSTY) Then
    SQRHYP = Abs(SECONDX - FIRSTX) ^ 2 + Abs(SECONDY - FIRSTY) ^ 2
    HYP = Sqr(SQRHYP)
    ADJ = SECONDX - FIRSTX
    COSALPHA = ADJ / HYP
    ANGLES = (1.570796 - Atn(COSALPHA / Sqr(1 - COSALPHA * COSALPHA))) * 180 / 3.1415926538979
    AZIMUTH = 270 - ANGLES
  End If
  If (SECONDX > FIRSTX) And (SECONDY < FIRSTY) Then
    SQRHYP = Abs(SECONDX - FIRSTX) ^ 2 + Abs(SECONDY - FIRSTY) ^ 2
    HYP = Sqr(SQRHYP)
    ADJ = SECONDX - FIRSTX
    COSALPHA = ADJ / HYP
    ANGLES = (1.570796 - Atn(COSALPHA / Sqr(1 - COSALPHA * COSALPHA))) * 180 / 3.1415926538979
```

```
        AZIMUTH = 270 + ANGLES
      End If
      DATA((j + 1), 4) = LENGTHOFLINE
      DATA((j + 1), 5) = AZIMUTH
      For pp = 1 To 7
        DATA(j, pp) = 0
      Next pp
    End If
End If
If DATA(j, 4) >= 2 And DATA(j, 4) < 7 Then    'Check here for VECTORS > 2m and < 5m
  If DATA((j - 1), 2) <> 0 Then
    If Abs((DATA(j, 5)) - DATA((j - 1), 5)) < 300 Then
      back_difference = Abs(DATA(j, 5) - DATA((j - 1), 5))
    End If
    If Abs(DATA(j, 5) - DATA((j + 1), 5)) < 300 Then
      front_difference = Abs(DATA(j, 5) - DATA((j + 1), 5))
    End If
  ElseIf DATA((j - 1), 2) = 0 Then
    If DATA((j - 2), 2) <> 0 Then
      If Abs((DATA(j, 5)) - DATA((j - 2), 5)) < 300 Then
        back_difference = Abs(DATA(j, 5) - DATA((j - 2), 5))
      End If
      If Abs(DATA(j, 5) - DATA((j + 1), 5)) < 300 Then
        front_difference = Abs(DATA(j, 5) - DATA((j + 1), 5))
      End If
    ElseIf DATA((j - 2), 2) = 0 Then
      If Abs((DATA(j, 5)) - DATA((j - 3), 5)) < 300 Then
        back_difference = Abs(DATA(j, 5) - DATA((j - 3), 5))
      End If
      If Abs(DATA(j, 5) - DATA((j + 1), 5)) < 300 Then
        front_difference = Abs(DATA(j, 5) - DATA((j + 1), 5))
      End If
    End If
  End If
  If DATA((j - 1), 2) <> 0 Then
    If Abs((DATA(j, 5)) - DATA((j - 1), 5)) > 300 Then
      If DATA(j, 5) > 300 Then
        back_difference = -((360 - DATA(j, 5)) + DATA((j - 1), 5))
      ElseIf DATA((j - 1), 5) > 300 Then
        back_difference = DATA(j, 5) + (360 - DATA(j - 1, 5))
      End If
    End If
  ElseIf DATA((j - 1), 2) = 0 Then
    If DATA((j - 2), 2) <> 0 Then
      If Abs((DATA(j, 5)) - DATA((j - 2), 5)) > 300 Then
        If DATA(j, 5) > 300 Then
          back_difference = -((360 - DATA(j, 5)) + DATA((j - 2), 5))
        ElseIf DATA((j - 2), 5) > 300 Then
          back_difference = DATA(j, 5) + (360 - DATA((j - 2), 5))
        End If
      End If
    ElseIf DATA((j - 2), 2) = 0 Then
      If Abs((DATA(j, 5)) - DATA((j - 3), 5)) > 300 Then
        If DATA(j, 5) > 300 Then
          back_difference = -((360 - DATA(j, 5)) + DATA((j - 3), 5))
        ElseIf DATA((j - 3), 5) > 300 Then
          back_difference = DATA(j, 5) + (360 - DATA((j - 3), 5))
        End If
      End If
    End If
  End If
  If Abs((DATA(j, 5)) - DATA((j + 1), 5)) > 300 Then
    If DATA(j, 5) > 300 Then
      front_difference = -((360 - DATA(j, 5)) + DATA((j + 1), 5))
    ElseIf DATA((j - 1), 5) > 300 Then
      front_difference = DATA(j, 5) + (360 - DATA(j + 1, 5))
    End If
  End If
  If Abs(back_difference) < 2 Or Abs(front_difference) < 2 Then
    If Abs(back_difference) < Abs(front_difference) Then
```

```
    ' Use points (j-2) and (j)
    ' The angles start at the second node and look back at the first.
    If DATA((j - 2), 2) <> 0 Then
       FIRSTX = DATA(j, 2):       FIRSTY = DATA(j, 3)
       SECONDX = DATA((j - 2), 2): SECONDY = DATA((j - 2), 3)
    ElseIf DATA((j - 2), 2) = 0 Then
       If DATA((j - 3), 2) <> 0 Then
          FIRSTX = DATA(j, 2):       FIRSTY = DATA(j, 3)
          SECONDX = DATA((j - 3), 2): SECONDY = DATA((j - 3), 3)
       ElseIf DATA((j - 3), 2) = 0 Then
          FIRSTX = DATA(j, 2):       FIRSTY = DATA(j, 3)
          SECONDX = DATA((j - 4), 2): SECONDY = DATA((j - 4), 3)
       End If
    End If
    SQRTLENGTHOFLINE = Abs(SECONDX - FIRSTX) ^ 2 + Abs(SECONDY - FIRSTY) ^ 2
    LENGTHOFLINE = (SQRTLENGTHOFLINE) ^ 0.5
    If (FIRSTX = SECONDX) And (SECONDY > FIRSTY) Then
       AZIMUTH = 180
    End If
    If (FIRSTY = SECONDY) And (SECONDX > FIRSTX) Then
       AZIMUTH = 270
    End If
    If (FIRSTX = SECONDX) And (SECONDY < FIRSTY) Then
       AZIMUTH = 0
    End If
    If (FIRSTY = SECONDY) And (SECONDX < FIRSTX) Then
       AZIMUTH = 90
    End If
    If (SECONDX < FIRSTX) And (SECONDY < FIRSTY) Then
       SQRHYP = Abs(SECONDX - FIRSTX) ^ 2 + Abs(SECONDY - FIRSTY) ^ 2
       HYP = Sqr(SQRHYP)
       ADJ = FIRSTX - SECONDX
       COSALPHA = ADJ / HYP
       ANGLES = (1.570796 - Atn(COSALPHA / Sqr(1 - COSALPHA * COSALPHA))) * 180 / 3.1415926538979
       AZIMUTH = 90 - ANGLES
    End If
    If (SECONDX < FIRSTX) And (SECONDY > FIRSTY) Then
       SQRHYP = (Abs(SECONDX - FIRSTX)) ^ 2 + (Abs(SECONDY - FIRSTY)) ^ 2
       HYP = Sqr(SQRHYP)
       ADJ = FIRSTX - SECONDX
       COSALPHA = ADJ / HYP
       ANGLES = (1.570796 - Atn(COSALPHA / Sqr(1 - COSALPHA * COSALPHA))) * 180 / 3.1415926538979
       AZIMUTH = 90 + ANGLES
    End If
    If (SECONDX > FIRSTX) And (SECONDY > FIRSTY) Then
       SQRHYP = Abs(SECONDX - FIRSTX) ^ 2 + Abs(SECONDY - FIRSTY) ^ 2
       HYP = Sqr(SQRHYP)
       ADJ = SECONDX - FIRSTX
       COSALPHA = ADJ / HYP
       ANGLES = (1.570796 - Atn(COSALPHA / Sqr(1 - COSALPHA * COSALPHA))) * 180 / 3.1415926538979
       AZIMUTH = 270 - ANGLES
    End If
    If (SECONDX > FIRSTX) And (SECONDY < FIRSTY) Then
       SQRHYP = Abs(SECONDX - FIRSTX) ^ 2 + Abs(SECONDY - FIRSTY) ^ 2
       HYP = Sqr(SQRHYP)
       ADJ = SECONDX - FIRSTX
       COSALPHA = ADJ / HYP
       ANGLES = (1.570796 - Atn(COSALPHA / Sqr(1 - COSALPHA * COSALPHA))) * 180 / 3.1415926538979
       AZIMUTH = 270 + ANGLES
    End If
    DATA(j, 4) = LENGTHOFLINE
    DATA(j, 5) = AZIMUTH
    For pp = 1 To 7
       DATA((j - 1), pp) = 0
    Next pp
ElseIf Abs(back_difference) >= Abs(front_difference) And j < count_vectors Then
    ' Use points (j-1) and (j+1)
    ' The angles start at the second node and look back at the first.
    If DATA((j - 1), 2) <> 0 Then
       FIRSTX = DATA((j + 1), 2):  FIRSTY = DATA((j + 1), 3)
```

```
                    SECONDX = DATA((j - 1), 2): SECONDY = DATA((j - 1), 3)
                  ElseIf DATA((j - 1), 2) = 0 Then
                    If DATA((j - 2), 2) <> 0 Then
                      FIRSTX = DATA((j + 1), 2):     FIRSTY = DATA((j + 1), 3)
                      SECONDX = DATA((j - 2), 2): SECONDY = DATA((j - 2), 3)
                    ElseIf DATA((j - 2), 2) = 0 Then
                      FIRSTX = DATA((j + 1), 2):     FIRSTY = DATA((j + 1), 3)
                      SECONDX = DATA((j - 3), 2): SECONDY = DATA((j - 3), 3)
                    End If
                  End If
                  SQRTLENGTHOFLINE = Abs(SECONDX - FIRSTX) ^ 2 + Abs(SECONDY - FIRSTY) ^ 2
                  LENGTHOFLINE = (SQRTLENGTHOFLINE) ^ 0.5
                  If (FIRSTX = SECONDX) And (SECONDY > FIRSTY) Then
                    AZIMUTH = 180
                  End If
                  If (FIRSTY = SECONDY) And (SECONDX > FIRSTX) Then
                    AZIMUTH = 270
                  End If
                  If (FIRSTX = SECONDX) And (SECONDY < FIRSTY) Then
                    AZIMUTH = 0
                  End If
                  If (FIRSTY = SECONDY) And (SECONDX < FIRSTX) Then
                    AZIMUTH = 90
                  End If
                  If (SECONDX < FIRSTX) And (SECONDY < FIRSTY) Then
                    SQRHYP = Abs(SECONDX - FIRSTX) ^ 2 + Abs(SECONDY - FIRSTY) ^ 2
                    HYP = Sqr(SQRHYP)
                    ADJ = FIRSTX - SECONDX
                    COSALPHA = ADJ / HYP
                    ANGLES = (1.570796 - Atn(COSALPHA / Sqr(1 - COSALPHA * COSALPHA))) * 180 / 3.1415926538979
                    AZIMUTH = 90 - ANGLES
                  End If
                  If (SECONDX < FIRSTX) And (SECONDY > FIRSTY) Then
                    SQRHYP = (Abs(SECONDX - FIRSTX)) ^ 2 + (Abs(SECONDY - FIRSTY)) ^ 2
                    HYP = Sqr(SQRHYP)
                    ADJ = FIRSTX - SECONDX
                    COSALPHA = ADJ / HYP
                    ANGLES = (1.570796 - Atn(COSALPHA / Sqr(1 - COSALPHA * COSALPHA))) * 180 / 3.1415926538979
                    AZIMUTH = 90 + ANGLES
                  End If
                  If (SECONDX > FIRSTX) And (SECONDY > FIRSTY) Then
                    SQRHYP = Abs(SECONDX - FIRSTX) ^ 2 + Abs(SECONDY - FIRSTY) ^ 2
                    HYP = Sqr(SQRHYP)
                    ADJ = SECONDX - FIRSTX
                    COSALPHA = ADJ / HYP
                    ANGLES = (1.570796 - Atn(COSALPHA / Sqr(1 - COSALPHA * COSALPHA))) * 180 / 3.1415926538979
                    AZIMUTH = 270 - ANGLES
                  End If
                  If (SECONDX > FIRSTX) And (SECONDY < FIRSTY) Then
                    SQRHYP = Abs(SECONDX - FIRSTX) ^ 2 + Abs(SECONDY - FIRSTY) ^ 2
                    HYP = Sqr(SQRHYP)
                    ADJ = SECONDX - FIRSTX
                    COSALPHA = ADJ / HYP
                    ANGLES = (1.570796 - Atn(COSALPHA / Sqr(1 - COSALPHA * COSALPHA))) * 180 / 3.1415926538979
                    AZIMUTH = 270 + ANGLES
                  End If
                  DATA((j + 1), 4) = LENGTHOFLINE
                  DATA((j + 1), 5) = AZIMUTH
                  For pp = 1 To 7
                    DATA(j, pp) = 0
                  Next pp
                End If
              End If
            End If
          End If
        End If
      Next j
Next i
Close #7
```

```
For i = 1 To count_roads
   DATA4 = "k:\Oberholf\HSU\mf\output\curves_output\WEED" + LTrim$(RTrim$(Str$(i)))
   Open DATA4 For Output As #4
   For j = 1 To count_vectors
      If DATA(j, 1) = i Then
         If DATA(j, 1) <> 0 Then
            Print #4, i, DATA(j, 2), DATA(j, 3), DATA(j, 4), DATA(j, 5)
            afterweed_vector(i) = afterweed_vector(i) + 1
            'Print #4, DATA(j, 2) & "," & DATA(j, 3)
         End If
      End If
   Next j
   Close #4
Next i
Close #4

'=============================================================================
' STEP 4: PUTTING X, Y, DISTANCE AND ANGLE IN AN ARRAY (ROADS)
'=============================================================================
' Clear the Array to make space for the new data
Erase DATA

' Now create the new Array
new_vector_count = 0
For i = 1 To count_roads
   DATA4 = "k:\Oberholf\HSU\mf\output\curves_output\WEED" + LTrim$(RTrim$(Str$(i)))
   Open DATA4 For Input As #4
   Do While Not EOF(4)
      Input #4, i, X, Y, dist, angl
      vector_in_road(i) = vector_in_road(i) + 1
      new_vector_count = new_vector_count + 1
      DATA(new_vector_count, 1) = i
      DATA(new_vector_count, 2) = X
      DATA(new_vector_count, 3) = Y
      DATA(new_vector_count, 4) = dist
      DATA(new_vector_count, 5) = angl
      DATA(new_vector_count, 7) = 5      '5 refer to SINGLE ROAD
   Loop
   Close #4
Next i

'GoTo jump_start

'=============================================================================
' STEP 5: CONNECT ROADS WITH OTHER ROADS, AND CHECK FOR CURVES
'=============================================================================
Open DATA7 For Input As #7
For i = 1 To count_roads
   Do While Not EOF(7)
      Input #7, roadid, einde, X, Y
      ends_count = ends_count + 1
      ENDS(ends_count, 1) = roadid
      ENDS(ends_count, 2) = einde
      ENDS(ends_count, 3) = X
      ENDS(ends_count, 4) = Y
   Loop
Next i
Close #7

For i = 1 To count_roads
  For j = 1 To new_vector_count 'THIS IS FOR THE FIRST ROAD SECTION
   If DATA(j, 1) = i Then
    count_001 = count_001 + 1
'ROADS THAT CONNECT AT THE START OF THE CURRENT ROAD
    If count_001 = 1 Then 'Check for roads that join at BEGINNING of first road
     If (DATA(j, 2) <> DATA((j + afterweed_vector(i)) - 1, 2) Or _
     DATA(j, 3) <> DATA((j + afterweed_vector(i)) - 1, 3)) Then 'Check if road is STRAIGHT
       For t = 1 To ends_count 'SECTION BETWEEN THE FIRST ROAD AND SECOND ROAD
         'The DATA-road is the BEGINNING of the data(j,1) road
         'The ENDS-road is the END/BEGINNING of the road joining the data(j,1) road
```

```
        If DATA(j, 2) = ENDS(t, 3) And _
        DATA(j, 3) = ENDS(t, 4) And _
        ENDS(t, 1) <> i Then
          pad = DATA(j, 1)
          en_kyk_na = ENDS(t, 1)
          oneside = ENDS(t, 2)
          otherroad = ENDS(t, 1)
'-------------------------------------------------------------------------------------------
           'Print all the data in the first road, since it is connected
           If afterweed_vector(i) > 10 Then
            firstroadcount = 9
           ElseIf afterweed_vector(i) <= 10 Then
            firstroadcount = afterweed_vector(i) - 1
           End If
'-------------------------------------------------------------------------------------------
           count_003 = 0
           For u = 1 To new_vector_count 'THIS IS FOR THE SECOND ROAD SECTION
            If DATA(u, 1) = ENDS(t, 1) Then
             count_003 = count_003 + 1
             midroad_id = DATA(u, 1)
             If DATA(u, 2) = ENDS(t, 3) And DATA(u, 3) = ENDS(t, 4) Then
              If afterweed_vector(midroad_id) > 10 Then
               midroad_count = 9
              ElseIf afterweed_vector(midroad_id) <= 10 Then
               midroad_count = afterweed_vector(midroad_id) - 1
              End If
'-------------------------------------------------------------------------------------------
              If afterweed_vector(midroad_id) >= 10 Then
               newcurves = newcurves + 1
               DATA8 = "k:\Oberholf\HSU\mf\output\curves_output\combcurv" + LTrim$(RTrim$(Str$(newcurves)))
               Open DATA8 For Output As #8
               If count_003 = 1 Then
                For v = 0 To firstroadcount
                 Write #8, i, DATA(j + firstroadcount - v, 2), _
                 DATA(j + firstroadcount - v, 3)
                Next v
                For v = 1 To midroad_count
                 If DATA(u + v, 1) = ENDS(t, 1) Then
                  Write #8, midroad_id, DATA(u + v, 2), DATA(u + v, 3)
                 End If
                Next v
                Close #8
               ElseIf count_003 = afterweed_vector(ENDS(t, 1)) Then
                For v = 0 To firstroadcount
                 Write #8, i, DATA(j + firstroadcount - v, 2), _
                 DATA(j + firstroadcount - v, 3)
                Next v
                For v = 1 To midroad_count
                 If DATA(u - v, 1) = ENDS(t, 1) Then
                  Write #8, midroad_id, DATA(u - v, 2), DATA(u - v, 3)
                 End If
                Next v
                Close #8
               End If
 GOTO Next
'-------------------------------------------------------------------------------------------
              'If there are a short road connecting to the current road,
              'then I will select the next road connecting with the short road.
'-------------------------------------------------------------------------------------------
              ElseIf afterweed_vector(midroad_id) < 10 Then
               If count_003 = 1 Then 'First road connect to START of MIDDLE ROAD
                midroad_thirdroad_X = DATA(u + afterweed_vector(midroad_id) - 1, 2)
                midroad_thirdroad_Y = DATA(u + afterweed_vector(midroad_id) - 1, 3)
               ElseIf count_003 = afterweed_vector(ENDS(t, 1)) Then 'First road connect to
                                                    'END of MIDDLE ROAD
                midroad_thirdroad_X = DATA(u - afterweed_vector(midroad_id) + 1, 2)
                midroad_thirdroad_Y = DATA(u - afterweed_vector(midroad_id) + 1, 3)
               End If                                                SECTION
'-------------------------------------------------------------------------------------------
```

```
' This is where the end of the second road connect to third section
'-------------------------------------------------------------------------------------------------------------------------------
count_004 = 0 'count_004 is the count for the THIRD ROAD
For tt = 1 To ends_count
 For jj = 1 To new_vector_count
  If DATA(jj, 1) = ENDS(tt, 1) Then 'If the ROAD ID = PREVIOUS ROAD
   count_004 = count_004 + 1
   If ENDS(tt, 3) = midroad_thirdroad_X And _
   ENDS(tt, 4) = midroad_thirdroad_Y And _
   ENDS(tt, 1) <> midroad_id Then  'If the current END = END of the previous road
    thirdroad_id = DATA(jj, 1)
    If DATA(jj, 2) = midroad_thirdroad_X And _
    DATA(jj, 3) = midroad_thirdroad_Y And _
    midroad_id <> thirdroad_id Then 'If the CURRENT ROAD end = PREVIOUS ROAD end
     If afterweed_vector(thirdroad_id) > 10 Then 'Decide how far to go back into each road
      thirdroad_count = 9
     ElseIf afterweed_vector(thirdroad_id) <= 10 Then
      thirdroad_count = afterweed_vector(thirdroad_id) - 1
     End If
     newcurves = newcurves + 1
     DATA8 = "k:\Oberholf\HSU\mf\output\curves_output\combcurv" + LTrim$(RTrim$(Str$(newcurves)))
     Open DATA8 For Output As #8
     If count_004 = 1 Then 'If the THIRD road START at the BEGINNING
      For v = 0 To firstroadcount 'If the FIRST road START at the BEGINNING
       Write #8, i, DATA(j + firstroadcount - v, 2), _
       DATA(j + firstroadcount - v, 3)
      Next v
     If count_003 = 1 Then 'If the SECOND road START at the BEGINNING
      For v = 1 To midroad_count
       If DATA(u + v, 1) = ENDS(t, 1) Then
        Write #8, midroad_id, DATA(u + v, 2), DATA(u + v, 3)
       End If
      Next v
     ElseIf count_003 = afterweed_vector(ENDS(t, 1)) Then 'If the SECOND road START at the END
      For v = midroad_count To 1 Step -1
       If DATA(u - afterweed_vector(midroad_id) + v, 1) = ENDS(t, 1) Then
        Write #8, midroad_id, DATA(u - afterweed_vector(midroad_id) + v, 2), _
        DATA(u - afterweed_vector(midroad_id) + v, 3)
       End If
      Next v
     End If
     For v = 1 To thirdroad_count
      Write #8, thirdroad_id, DATA(jj + v, 2), DATA(jj + v, 3)
     Next v
     Close #8
     ElseIf count_004 > 1 Then 'If the THIRD road START at the END
      For v = 0 To firstroadcount 'If the FIRST road START at the BEGINNING
       Write #8, i, DATA(j + firstroadcount - v, 2), _
       DATA(j + firstroadcount - v, 3)
      Next v
      If count_003 = 1 Then  'If the SECOND road START at the BEGINNING
       For v = 1 To midroad_count
        If DATA(u + v, 1) = ENDS(t, 1) Then
         Write #8, midroad_id, DATA(u + v, 2), DATA(u + v, 3)
        End If
       Next v
      ElseIf count_003 = afterweed_vector(ENDS(t, 1)) Then 'If the SECOND road START at the END
       For v = midroad_count To 1 Step -1
        If DATA(u - afterweed_vector(midroad_id) + v, 1) = ENDS(t, 1) Then
         Write #8, midroad_id, DATA(u - afterweed_vector(midroad_id) + v, 2), _
         DATA(u - afterweed_vector(midroad_id) + v, 3)
        End If
       Next v
      End If
      For v = 1 To thirdroad_count
       Write #8, thirdroad_id, DATA(jj - v, 2), DATA(jj - v, 3)
      Next v
      Close #8
     End If
```

```
                    End If
                   End If
                  End If
                 Next jj
                 count_004 = 0
                Next tt
'-------------------------------------------------------------------------------------------------------------------------------
              End If 'ElseIf afterweed_vector(midroad_id) < 10
            End If 'If DATA(u, 2) = ENDS(t, 3) And DATA(u, 3) = ENDS(t, 4)
          End If 'If DATA(u, 1) = ENDS(t, 1) Then
        Next u
       'End If 'If SECOND ROAD is not circular
      End If 'If the ROAD has an end that match with END
    Next t

'CIRCULAR ROADS START HERE(Road connect at BEGINNING)
    ElseIf (DATA(j, 2) = DATA((j + afterweed_vector(i)) - 1, 2) And _
    DATA(j, 3) = DATA((j + afterweed_vector(i)) - 1, 3)) Then 'Check if road is CIRCULAR
'Circular roads at the start
     If count_001 = 1 And (DATA(j, 2) = DATA((j + afterweed_vector(i) - 1), 2) And _
     DATA(j, 3) = DATA((j + afterweed_vector(i) - 1), 3)) Then
      For t = 1 To ends_count
       If DATA(j, 2) = ENDS(t, 3) And _
       DATA(j, 3) = ENDS(t, 4) And ENDS(t, 1) <> i Then
        'Approach the CIRCLE ROAD from the ONE side
        newcurves = newcurves + 1
        DATA8 = "k:\Oberholf\HSU\mf\output\curves_output\combcurv" + LTrim$(RTrim$(Str$(newcurves)))
        Open DATA8 For Output As #8
        count_003 = 0
        If afterweed_vector(i) > 10 Then
         firstroadcount = 9
        ElseIf afterweed_vector(i) <= 10 Then
         firstroadcount = afterweed_vector(i) - 1
        End If
'-------------------------------------------------------------------------------------------------------------------------------
        For u = 1 To new_vector_count
         If DATA(u, 1) = i Then
          count_003 = count_003 + 1
          If count_003 = 1 Then
           For v = 0 To firstroadcount
            Write #8, i, DATA(u + firstroadcount - v, 2), DATA(u + firstroadcount - v, 3)
           Next v
          End If
         End If
        Next u
        count_003 = 0

        If ENDS(t, 2) = 10 Then 'This road START at this point
         For u = 1 To new_vector_count
          If DATA(u, 1) = ENDS(t, 1) Then
           midroad_id = DATA(u, 1)
           If afterweed_vector(midroad_id) > 10 Then 'Decide how far to go back into each road
            midroad_count = 9
           ElseIf afterweed_vector(midroad_id) <= 10 Then
            midroad_count = afterweed_vector(midroad_id) - 1
           End If
           count_003 = count_003 + 1
           If count_003 = 1 Then
            For v = 1 To midroad_count
             Write #8, DATA(u, 1), DATA(u + v, 2), DATA(u + v, 3)
            Next v
           End If
          End If
         Next u
        ElseIf ENDS(t, 2) = 20 Then 'This road ENDS at this point
         For u = new_vector_count To 1 Step -1
          If DATA(u, 1) = ENDS(t, 1) Then
           midroad_id = DATA(u, 1)
           If afterweed_vector(midroad_id) > 10 Then 'Decide how far to go back into each road
            midroad_count = 9
```

```
                ElseIf afterweed_vector(midroad_id) <= 10 Then
                  midroad_count = afterweed_vector(midroad_id) - 1
                End If
                count_003 = count_003 + 1
                If count_003 = 1 Then
                  For v = 1 To midroad_count
                    Write #8, DATA(u, 1), DATA(u - v, 2), DATA(u - v, 3)
                  Next v
                End If
              End If
            Next u
          End If
          Close #8
'-------------------------------------------------------------------------------------------------------------
          'Now Approach the CIRCLE ROAD from the OTHER side
          newcurves = newcurves + 1
          DATA8 = "k:\Oberholf\HSU\mf\output\curves_output\combcurv" + LTrim$(RTrim$(Str$(newcurves)))
          Open DATA8 For Output As #8
          count_003 = 0
          For u = new_vector_count To 1 Step -1
            If DATA(u, 1) = i Then
              count_003 = count_003 + 1
              If count_003 = 1 Then
                For v = 1 To firstroadcount
                  Write #8, i, DATA(u - firstroadcount + v, 2), DATA(u - firstroadcount + v, 3)
                Next v
              End If
            End If
          Next u
          count_003 = 0
          If ENDS(t, 2) = 10 Then 'This road START at this point
            For u = 1 To new_vector_count
              If DATA(u, 1) = ENDS(t, 1) Then
                count_003 = count_003 + 1
                If count_003 = 1 Then
                  'Hier is waar die KAK is: road 267
                  For v = 1 To midroad_count
                    Write #8, DATA(u, 1), DATA(u + v, 2), DATA(u + v, 3)
                  Next v
                End If
              End If
            Next u
          ElseIf ENDS(t, 2) = 20 Then 'This road ENDS at this point
            For u = new_vector_count To 1 Step -1
              If DATA(u, 1) = ENDS(t, 1) Then
              count_003 = count_003 + 1
                If count_003 = 1 Then
                  For v = 1 To midroad_count
                    Write #8, DATA(u, 1), DATA(u - v, 2), DATA(u - v, 3)
                  Next v
                End If
              End If
            Next u
          End If
          Close #8
'-------------------------------------------------------------------------------------------------------------
        End If
        count_003 = 0
      Next t
    End If
  End If 'If FIRST ROAD is not circular

'ROADS THAT CONNECT AT THE END OF THE CURRENT ROAD
  ElseIf count_001 > 1 Then 'Check for roads that join at END of first road
    If (DATA(j, 2) <> DATA((j + afterweed_vector(i)) - 1, 2) Or _
    DATA(j, 3) <> DATA((j + afterweed_vector(i)) - 1, 3)) Then 'Check if road is STRAIGHT
      For t = 1 To ends_count 'SECTION BETWEEN THE FIRST ROAD AND SECOND ROAD
        'The DATA-road is the BEGINNING of the data(j,1) road
        'The ENDS-road is the END/BEGINNING of the road joining the data(j,1) road
        If DATA(j, 2) = ENDS(t, 3) And _
```

```
        DATA(j, 3) = ENDS(t, 4) And _
        ENDS(t, 1) <> i Then
          pad = DATA(j, 1)
          en_kyk_na = ENDS(t, 1)
          oneside = ENDS(t, 2)
          otherroad = ENDS(t, 1)
          If ENDS(t, 1) = ENDS(t + 1, 1) And _
          ENDS(t, 3) = ENDS(t + 1, 3) And _
          ENDS(t, 4) = ENDS(t + 1, 4) Then
            GoTo nextroad 'This was a circular route. I will deal with it later
          Else
'-------------------------------------------------------------------------------------------
            'Print all the data in the first road, since it is connected
            If afterweed_vector(i) > 10 Then
              firstroadcount = 9
            ElseIf afterweed_vector(i) <= 10 Then
              firstroadcount = afterweed_vector(i) - 1
            End If
'-------------------------------------------------------------------------------------------
            count_003 = 0
            For u = 1 To new_vector_count 'THIS IS FOR THE SECOND ROAD SECTION
              If DATA(u, 1) = ENDS(t, 1) Then
                count_003 = count_003 + 1
                midroad_id = DATA(u, 1)
                If DATA(u, 2) = ENDS(t, 3) And DATA(u, 3) = ENDS(t, 4) Then
'-------------------------------------------------------------------------------------------
                  If afterweed_vector(midroad_id) > 10 Then 'Decide how far to go back into each road
                    midroad_count = 9
                  ElseIf afterweed_vector(midroad_id) <= 10 Then
                    midroad_count = afterweed_vector(midroad_id) - 1
                  End If
                  If afterweed_vector(midroad_id) >= 10 Then
                    newcurves = newcurves + 1
                    DATA8 = "k:\Oberholf\HSU\mf\output\curves_output\combcurv" + LTrim$(RTrim$(Str$(newcurves)))
                    Open DATA8 For Output As #8
                    If count_003 = 1 Then
                      For v = firstroadcount To 0 Step -1
                        Write #8, i, DATA(j - v, 2), DATA(j - v, 3)
                      Next v
                      For v = 1 To midroad_count
                        If DATA(u + v, 1) = ENDS(t, 1) Then
                          Write #8, midroad_id, DATA(u + v, 2), DATA(u + v, 3)
                        End If
                      Next v
                      Close #8
                    ElseIf count_003 = afterweed_vector(ENDS(t, 1)) Then
                      For v = firstroadcount To 0 Step -1
                        Write #8, i, DATA(j - v, 2), DATA(j - v, 3)
                      Next v
                      For v = 1 To midroad_count
                        If DATA(u - v, 1) = ENDS(t, 1) Then
                          Write #8, midroad_id, DATA(u - v, 2), DATA(u - v, 3)
                        End If
                      Next v
                      Close #8
                    End If
GOTO Next
'-------------------------------------------------------------------------------------------
                  'If there are a short road connecting to the current road,
                  'then I will select the next road connecting with the short road.
'-------------------------------------------------------------------------------------------
                  ElseIf afterweed_vector(midroad_id) < 10 Then
                    If count_003 = 1 Then 'First road connect to START of MIDDLE ROAD
                      midroad_thirdroad_X = DATA(u + afterweed_vector(midroad_id) - 1, 2)
                      midroad_thirdroad_Y = DATA(u + afterweed_vector(midroad_id) - 1, 3)
                    ElseIf count_003 = afterweed_vector(ENDS(t, 1)) Then 'First road connect to
                                                        'END of MIDDLE ROAD
                      midroad_thirdroad_X = DATA(u - afterweed_vector(midroad_id) + 1, 2)
                      midroad_thirdroad_Y = DATA(u - afterweed_vector(midroad_id) + 1, 3)
                    End If                                              SECTION
```

```
'-------------------------------------------------------------------------------------------------

        ' This is where the end of the second road connect to third section
'-------------------------------------------------------------------------------------------------------------------
        count_004 = 0 'count_004 is the count for the THIRD ROAD
        For tt = 1 To ends_count
          For jj = 1 To new_vector_count
            If DATA(jj, 1) = ENDS(tt, 1) Then 'If the ROAD ID = PREVIOUS ROAD
              count_004 = count_004 + 1
              If ENDS(tt, 3) = midroad_thirdroad_X And _
              ENDS(tt, 4) = midroad_thirdroad_Y And _
              ENDS(tt, 1) <> midroad_id Then  'If the current END = END of the previous road
                thirdroad_id = DATA(jj, 1)
                If afterweed_vector(midroad_id) > 10 Then 'Decide how far to go back into each road
                  midroad_count = 9
                ElseIf afterweed_vector(midroad_id) <= 10 Then
                  midroad_count = afterweed_vector(midroad_id) - 1
                End If
                If DATA(jj, 2) = midroad_thirdroad_X And _
                DATA(jj, 3) = midroad_thirdroad_Y And _
                midroad_id <> thirdroad_id Then 'If the CURRENT ROAD end = PREVIOUS ROAD end
                  If afterweed_vector(thirdroad_id) > 10 Then 'Decide how far to go back into each road
                    thirdroad_count = 9
                  ElseIf afterweed_vector(thirdroad_id) <= 10 Then
                    thirdroad_count = afterweed_vector(thirdroad_id) - 1
                  End If
                  newcurves = newcurves + 1
                  DATA8 = "k:\Oberholf\HSU\mf\output\curves_output\combcurv" + LTrim$(RTrim$(Str$(newcurves)))
                  Open DATA8 For Output As #8
                  If count_004 = 1 Then 'If the THIRD road START at the BEGINNING
                    For v = firstroadcount To 0 Step -1  'If the FIRST road START at the BEGINNING
                      Write #8, i, DATA(j - v, 2), DATA(j - v, 3)
                    Next v
                  If count_003 = 1 Then 'If the SECOND road START at the BEGINNING
                    For v = 1 To midroad_count
                      If DATA(u + v, 1) = ENDS(t, 1) Then
                        Write #8, midroad_id, DATA(u + v, 2), DATA(u + v, 3)
                      End If
                    Next v
                  ElseIf count_003 = afterweed_vector(ENDS(t, 1)) Then 'If the SECOND road START at the END
                    For v = midroad_count To 1 Step -1
                      If DATA(u - afterweed_vector(midroad_id) + v, 1) = ENDS(t, 1) Then
                        Write #8, midroad_id, DATA(u - afterweed_vector(midroad_id) + v, 2), _
                        DATA(u - afterweed_vector(midroad_id) + v, 3)
                      End If
                    Next v
                  End If
                  For v = 1 To thirdroad_count
                    Write #8, thirdroad_id, DATA(jj + v, 2), DATA(jj + v, 3)
                  Next v
                  Close #8
                  ElseIf count_004 > 1 Then 'If the THIRD road START at the END
                    For v = firstroadcount To 0 Step -1 'If the FIRST road START at the BEGINNING
                      Write #8, i, DATA(j - v, 2), DATA(j - v, 3)
                    Next v
                    If count_003 = 1 Then  'If the SECOND road START at the BEGINNING
                      For v = 1 To midroad_count
                        If DATA(u + v, 1) = ENDS(t, 1) Then
                          Write #8, midroad_id, DATA(u + v, 2), DATA(u + v, 3)
                        End If
                      Next v
                    ElseIf count_003 = afterweed_vector(ENDS(t, 1)) Then 'If the SECOND road START at the END
                      For v = midroad_count To 1 Step -1
                        If DATA(u - afterweed_vector(midroad_id) + v, 1) = ENDS(t, 1) Then
                          Write #8, midroad_id, DATA(u - afterweed_vector(midroad_id) + v, 2), _
                          DATA(u - afterweed_vector(midroad_id) + v, 3)
                        End If
                      Next v
                    End If
```

```
                        For v = 1 To thirdroad_count
                          Write #8, thirdroad_id, DATA(jj - v, 2), DATA(jj - v, 3)
                        Next v
                        Close #8
                      End If
                    End If
                  End If
                End If
              Next jj
              count_004 = 0
            Next tt
'-------------------------------------------------------------------------------------------------------------------------------
            End If 'ElseIf afterweed_vector(midroad_id) < 10
          End If 'If DATA(u, 2) = ENDS(t, 3) And DATA(u, 3) = ENDS(t, 4)
        End If 'If DATA(u, 1) = ENDS(t, 1) Then
      Next u
    End If 'If SECOND ROAD is not circular
  End If 'If the ROAD has an end that match with END
Next t
End If 'If FIRST ROAD is not circular
End If 'If count_001 = 1
Close #8
End If 'If DATA(j, 1) = i
nextroad:
Close #8
Next j
count_001 = 0
Next i

' Print all the Data to one file. Then take this file and convert it back to
' coordinates to get an idea if the process worked.
GoTo skip_kill
DATA10 = "k:\Oberholf\HSU\mf\output\curves_output\tempcurv"
DATA11 = "k:\Oberholf\HSU\mf\output\curves_output\XScurves"
DATA12 = "k:\Oberholf\HSU\mf\output\curves_output\tempcurv2"
Open DATA10 For Output As #10
Open DATA11 For Output As #11
For i = 1 To newcurves
  DATA8 = "k:\Oberholf\HSU\mf\output\curves_output\combcurv" + LTrim$(RTrim$(Str$(i)))
  Open DATA8 For Input As #8
  Print #10, "ROAD ID", i
  id_counter = 0
  Do While Not EOF(8)
    id_counter = id_counter + 1
    Input #8, id, X, Y
    Print #10, id, X, Y
    TEMPORARRY(id_counter) = id
  Loop
  XScount = 0
  For j = 1 To (id_counter - 1)
    If TEMPORARRY(j) - TEMPORARRY(j + 1) <> 0 Then
      XScount = XScount + 1
      XScurv(i, 1) = i
      If XScount = 1 Then
        XScurv(i, 2) = TEMPORARRY(j)
        XScurv(i, 3) = TEMPORARRY(j + 1)
      ElseIf XScount = 2 Then
        XScurv(i, 4) = TEMPORARRY(j + 1)
      End If
    End If
  Next j
    Print #11, XScurv(i, 1), XScurv(i, 2), XScurv(i, 3), XScurv(i, 4) 'i, thirdid, secondid, firstid
  Print #10, "end"
  Close #8
  For j = 1 To 30
    TEMPORARRY(j) = 0
  Next j
Next i
Print #10, "end"
Close #10
```

```
Close #11
kk = 1

'=========================================================================
' STEP 6: DELETE THE "COMBCURV"FILES THAT ARE DUPLICATED
'=========================================================================
' It is not necessary to follow this procedure. it was written merely to delete
' some of the files that were duplicated. It was not used in the final run in
' order to save on computational time.

'First check for segments that contain TWO vectors
For i = 1 To newcurves
   last_sector = XScurv(i, 4): mid_sector = XScurv(i, 3): first_sector = XScurv(i, 2)
   For j = 1 To newcurves
      If last_sector = 0 And XScurv(j, 4) = 0 And XScurv(j, 3) <> 0 Then
         If XScurv(j, 1) <> XScurv(i, 1) And XScurv(j, 2) = mid_sector And _
         XScurv(j, 3) = first_sector Then
            Kill "k:\Oberholf\HSU\mf\output\curves_output\combcurv" + LTrim$(RTrim$(Str$(j)))
            For k = (j + 1) To newcurves
               Name "k:\Oberholf\HSU\mf\output\curves_output\combcurv" + LTrim$(RTrim$(Str$(k))) As _
               "k:\Oberholf\HSU\mf\output\curves_output\combcurv" + LTrim$(RTrim$(Str$(k - 1)))
            Next k
            newcurves = newcurves - 1
            XScurv(j, 2) = 0: XScurv(j, 3) = 0: XScurv(j, 4) = 0
         End If
      End If
   Next j
Next i

'Then check for the segments that contain THREE vectors
For i = 1 To newcurves
   last_sector = XScurv(i, 4): mid_sector = XScurv(i, 3): first_sector = XScurv(i, 2)
   If last_sector = 0 Then
      GoTo next_newcurves
   Else
      For j = 1 To newcurves
         If XScurv(j, 1) <> XScurv(i, 1) And XScurv(j, 2) = last_sector And _
         XScurv(j, 3) = mid_sector And XScurv(j, 4) = first_sector Then
            Kill "k:\Oberholf\HSU\mf\output\curves_output\combcurv" + LTrim$(RTrim$(Str$(j)))
            For k = (j + 1) To newcurves
               Name "k:\Oberholf\HSU\mf\output\curves_output\combcurv" + LTrim$(RTrim$(Str$(k))) As _
               "k:\Oberholf\HSU\mf\output\curves_output\combcurv" + LTrim$(RTrim$(Str$(k - 1)))
            Next k
            newcurves = newcurves - 1
            XScurv(j, 2) = 0: XScurv(j, 3) = 0: XScurv(j, 4) = 0
         End If
      Next j
   End If
next_newcurves:
Next i

' Print all the Data to one file. Then take this file and convert it back to
' coordinates to get an idea if the process worked.
Open DATA12 For Output As #12
For i = 1 To newcurves
   newcount = newcount + 1
   ' "k:\Oberholf\HSU\mf\output\curves_output\combcurv" + LTrim$(RTrim$(Str$(j)))
   DATA8 = "k:\Oberholf\HSU\mf\output\curves_output\combcurv" + LTrim$(RTrim$(Str$(i)))
   Open DATA8 For Input As #8
   Print #12, i
   id_counter = 0
   Do While Not EOF(8)
      id_counter = id_counter + 1
      Input #8, id, X, Y
      Print #12, X, Y
      TEMPORARRY(id_counter) = id
   Loop
   Print #12, "end"
   Close #8
Next i
```

```
Close #12
skip_kill:

'=============================================================================
' STEP 7: CALCULATING DISTANCES AND ANGLES - FORWARD (ROAD INTERSECTIONS)
' This is where I Calculate the distances and angles in a forward direction
'=============================================================================
For i = 1 To newcurves
   DATA8 = "k:\Oberholf\HSU\mf\output\curves_output\combcurv" + LTrim$(RTrim$(Str$(i)))
   DATA9 = "k:\Oberholf\HSU\mf\output\curves_output\comcvang" + LTrim$(RTrim$(Str$(i)))
   FIRSTX = 0
   FIRSTY = 0
   SECONDX = 0
   SECONDY = 0
   countintersect(i) = 0
   LENGTHOFLINE = 0
   SQRTLENGTHOFLINE = 0
   COSALPHA = 0
   ANGLES = 0
   AZIMUTH = 0
   HYP = 0
   ADJ = 0
   X_first(i) = 0: X_last(i) = 0
   Y_first(i) = 0: Y_last(i) = 0

   Open DATA8 For Input As #8
   Open DATA9 For Output As #9
   Do While Not EOF(8)
      SECONDX = FIRSTX
      SECONDY = FIRSTY
      Input #8, id, FIRSTX, FIRSTY
      countintersect(i) = countintersect(i) + 1
      If countintersect(i) = 1 Then
         Print #9, FIRSTX, FIRSTY, LENGTHOFLINE, ANGLES, id
         SECONDX = FIRSTX
         SECONDY = FIRSTY
         Input #8, id, FIRSTX, FIRSTY
         countintersect(i) = countintersect(i) + 1
         X_first(i) = SECONDX 'Use this to check for circular route
         Y_first(i) = SECONDY 'Use this to check for circular route
      End If

      ' The angles start at the second node and look back at the first.
      SQRTLENGTHOFLINE = Abs(SECONDX - FIRSTX) ^ 2 + Abs(SECONDY - FIRSTY) ^ 2
      LENGTHOFLINE = (SQRTLENGTHOFLINE) ^ 0.5
      If (FIRSTX = SECONDX) And (SECONDY > FIRSTY) Then
         AZIMUTH = 180
      End If
      If (FIRSTY = SECONDY) And (SECONDX > FIRSTX) Then
         AZIMUTH = 270
      End If
      If (FIRSTX = SECONDX) And (SECONDY < FIRSTY) Then
         AZIMUTH = 0
      End If
      If (FIRSTY = SECONDY) And (SECONDX < FIRSTX) Then
         AZIMUTH = 90
      End If
      If (SECONDX < FIRSTX) And (SECONDY < FIRSTY) Then
         SQRHYP = Abs(SECONDX - FIRSTX) ^ 2 + Abs(SECONDY - FIRSTY) ^ 2
         HYP = Sqr(SQRHYP)
         ADJ = FIRSTX - SECONDX
         COSALPHA = ADJ / HYP
         ANGLES = (1.570796 - Atn(COSALPHA / Sqr(1 - COSALPHA * COSALPHA))) * 180 / 3.1415926538979
         AZIMUTH = 90 - ANGLES
      End If
      If (SECONDX < FIRSTX) And (SECONDY > FIRSTY) Then
         SQRHYP = (Abs(SECONDX - FIRSTX)) ^ 2 + (Abs(SECONDY - FIRSTY)) ^ 2
         HYP = Sqr(SQRHYP)
         ADJ = FIRSTX - SECONDX
         COSALPHA = ADJ / HYP
```

```
        ANGLES = (1.570796 - Atn(COSALPHA / Sqr(1 - COSALPHA * COSALPHA))) * 180 / 3.1415926538979
        AZIMUTH = 90 + ANGLES
      End If
      If (SECONDX > FIRSTX) And (SECONDY > FIRSTY) Then
        SQRHYP = Abs(SECONDX - FIRSTX) ^ 2 + Abs(SECONDY - FIRSTY) ^ 2
        HYP = Sqr(SQRHYP)
        ADJ = SECONDX - FIRSTX
        COSALPHA = ADJ / HYP
        ANGLES = (1.570796 - Atn(COSALPHA / Sqr(1 - COSALPHA * COSALPHA))) * 180 / 3.1415926538979
        AZIMUTH = 270 - ANGLES
      End If
      If (SECONDX > FIRSTX) And (SECONDY < FIRSTY) Then
        SQRHYP = Abs(SECONDX - FIRSTX) ^ 2 + Abs(SECONDY - FIRSTY) ^ 2
        HYP = Sqr(SQRHYP)
        ADJ = SECONDX - FIRSTX
        COSALPHA = ADJ / HYP
        ANGLES = (1.570796 - Atn(COSALPHA / Sqr(1 - COSALPHA * COSALPHA))) * 180 / 3.1415926538979
        AZIMUTH = 270 + ANGLES
      End If
      Print #9, FIRSTX, FIRSTY, LENGTHOFLINE, AZIMUTH, id
    Loop

    Close #8
    Close #9

Next i

'=============================================================================
' STEP 8: PUTTING X, Y, DISTANCE AND ANGLE IN AN ARRAY (INTERSECTIONS)      =
'=============================================================================
' Now create the new Array
total_vector_count = new_vector_count
totalroads = count_roads + newcurves
'ReDim DATA(200000, 14) As Double 'new_vector_count = 0
For i = 1 To newcurves
    DATA9 = "k:\Oberholf\HSU\mf\output\curves_output\comcvang" + LTrim$(RTrim$(Str$(i)))
    Open DATA9 For Input As #9
    Do While Not EOF(9)
      Input #9, X, Y, dist, angl, id
      vector_in_road(i + count_roads) = vector_in_road(i + count_roads) + 1
      total_vector_count = total_vector_count + 1
      DATA(total_vector_count, 1) = i + count_roads
      DATA(total_vector_count, 2) = X
      DATA(total_vector_count, 3) = Y
      DATA(total_vector_count, 4) = dist
      DATA(total_vector_count, 5) = angl
      DATA(total_vector_count, 7) = 9      '9 refer to JUNCTION BETWEEN 2 ROADS
      DATA(total_vector_count, 13) = id
      'DATA(total_vector_count, 14) = secondR
    Loop
    Close #9
Next i

'=============================================================================
' STEP 9: CALCULATE THE DIFFERENCE BETWEEN CONSECTUVE ANGLES AND FLAG THEM
' In this section we will calculate the difference between consecutive angles
' and flag the "Positive" and "Negative" angles.
'=============================================================================
tel = 0
For i = 1 To totalroads
    cc(i) = 0
    For j = 1 To total_vector_count
      If DATA(j, 1) = i Then
        cc(i) = cc(i) + 1
      End If
    Next j
    For j = 1 To total_vector_count
    If i = 113 Then
      dsdsdsd = 9
    End If
```

```
'New method to determine sign
    If DATA(j, 1) = i Then
      tel = tel + 1
      link_counter(i) = 1
      If tel > 2 And j > 2 Then
        'QUADRANT 1
        If DATA(j, 5) > 0 And DATA(j, 5) <= 90 Then
          'QUADRANT 1: Quadrant 1
          If DATA(j - 1, 5) > 0 And DATA(j - 1, 5) <= 90 Then
            DATA(j - 1, 6) = DATA(j, 5) - DATA(j - 1, 5)
          'QUADRANT 1: Quadrant 2
          ElseIf DATA(j - 1, 5) > 90 And DATA(j - 1, 5) <= 180 Then
            DATA(j - 1, 6) = DATA(j, 5) - DATA(j - 1, 5)
          'QUADRANT 1: Quadrant 3
          ElseIf DATA(j - 1, 5) > 180 And DATA(j - 1, 5) <= 270 Then
            If Abs(DATA(j, 5) - DATA(j - 1, 5)) > 180 Then
              DATA(j - 1, 6) = 360 - Abs(DATA(j, 5) - DATA(j - 1, 5))
            ElseIf Abs(DATA(j, 5) - DATA(j - 1, 5)) < 180 Then
              DATA(j - 1, 6) = DATA(j, 5) - DATA(j - 1, 5)
            End If
          'QUADRANT 1: Quadrant 4
          ElseIf DATA(j - 1, 5) > 270 And DATA(j - 1, 5) <= 360 Then
            DATA(j - 1, 6) = DATA(j, 5) + (360 - DATA(j - 1, 5))
          End If

        'QUADRANT 2
        ElseIf DATA(j, 5) > 90 And DATA(j, 5) <= 180 Then
          'QUADRANT 2: Quadrant 1
          If DATA(j - 1, 5) > 0 And DATA(j - 1, 5) <= 90 Then
            DATA(j - 1, 6) = DATA(j, 5) - DATA(j - 1, 5)
          'QUADRANT 2: Quadrant 2
          ElseIf DATA(j - 1, 5) > 90 And DATA(j - 1, 5) <= 180 Then
            DATA(j - 1, 6) = DATA(j, 5) - DATA(j - 1, 5)
          'QUADRANT 2: Quadrant 3
          ElseIf DATA(j - 1, 5) > 180 And DATA(j - 1, 5) <= 270 Then
            DATA(j - 1, 6) = DATA(j, 5) - DATA(j - 1, 5)
          'QUADRANT 2: Quadrant 4
          ElseIf DATA(j - 1, 5) > 270 And DATA(j - 1, 5) <= 360 Then
            If Abs(DATA(j, 5) - DATA(j - 1, 5)) > 180 Then
              DATA(j - 1, 6) = 360 - Abs(DATA(j, 5) - DATA(j - 1, 5))
            ElseIf Abs(DATA(j, 5) - DATA(j - 1, 5)) < 180 Then
              DATA(j - 1, 6) = DATA(j, 5) - DATA(j - 1, 5)
            End If
          End If

        'QUADRANT 3
        ElseIf DATA(j, 5) > 180 And DATA(j, 5) <= 270 Then
          'QUADRANT 3: Quadrant 1
          If DATA(j - 1, 5) > 0 And DATA(j - 1, 5) <= 90 Then
            If DATA(j, 5) - DATA(j - 1, 5) > 180 Then
              DATA(j - 1, 6) = -((360 - (DATA(j, 5) - DATA(j - 1, 5))))
            ElseIf DATA(j, 5) - DATA(j - 1, 5) < 180 Then
              DATA(j - 1, 6) = DATA(j, 5) - DATA(j - 1, 5)
            End If
          'QUADRANT 3: Quadrant 2
          ElseIf DATA(j - 1, 5) > 90 And DATA(j - 1, 5) <= 180 Then
            DATA(j - 1, 6) = DATA(j, 5) - DATA(j - 1, 5)
          'QUADRANT 3: Quadrant 3
          ElseIf DATA(j - 1, 5) > 180 And DATA(j - 1, 5) <= 270 Then
            DATA(j - 1, 6) = DATA(j, 5) - DATA(j - 1, 5)
          'QUADRANT 3: Quadrant 4
          ElseIf DATA(j - 1, 5) > 270 And DATA(j - 1, 5) <= 360 Then
            DATA(j - 1, 6) = DATA(j, 5) - DATA(j - 1, 5)
          End If

        'QUADRANT 4
        ElseIf DATA(j, 5) > 270 And DATA(j, 5) <= 360 Then
          'QUADRANT 4: Quadrant 1
          If DATA(j - 1, 5) > 0 And DATA(j - 1, 5) <= 90 Then
            DATA(j - 1, 6) = -((360 - DATA(j, 5)) + DATA(j - 1, 5))
```

```
            'QUADRANT 4: Quadrant 2
            ElseIf DATA(j - 1, 5) > 90 And DATA(j - 1, 5) <= 180 Then
               If DATA(j, 5) - DATA(j - 1, 5) > 180 Then
                  DATA(j - 1, 6) = -((360 - (DATA(j, 5) - DATA(j - 1, 5))))
               ElseIf DATA(j, 5) - DATA(j - 1, 5) < 180 Then
                  DATA(j - 1, 6) = DATA(j, 5) - DATA(j - 1, 5)
               End If
            'QUADRANT 4: Quadrant 3
            ElseIf DATA(j - 1, 5) > 180 And DATA(j - 1, 5) <= 270 Then
               DATA(j - 1, 6) = DATA(j, 5) - DATA(j - 1, 5)
            'QUADRANT 4: Quadrant 4
            ElseIf DATA(j - 1, 5) > 270 And DATA(j - 1, 5) <= 360 Then
               DATA(j - 1, 6) = DATA(j, 5) - DATA(j - 1, 5)
            End If
         End If
         link_counter(i) = link_counter(i) + 1
         If link_counter(i) = cc(i) Then GoTo get_out
      End If
   End If
 Next j
 tel = 0
get_out:
Next i
countcurv = 0

 '=============================================================================
' STEP 10: CALCULATE THE RADIUS                                          =
' Radius is calculated by putting three consecutive nodes on the edge of a    =
' circle. The radius of third circle is the radius of the curve.          =
'=============================================================================
DATA13 = "k:\Oberholf\HSU\mf\output\curves_output\Problem_R.txt"
Open DATA13 For Output As #13

Pi = 22 / 7

For i = 1 To totalroads
   For j = 1 To total_vector_count
      If DATA(j, 1) = i Then
         If totalroads = 6895 Then
            fferere = 4
         End If
         Count_2 = Count_2 + 1
         If Count_2 = 1 Then
            current_distance = 0
            total_distance = 0
         ElseIf Count_2 = 2 Then
            beginning_distance = 0
         ElseIf Count_2 >= 2 Then
            beginning_distance = 0
         End If
         If Count_2 >= 2 And Count_2 < vector_in_road(i) Then
            x__1 = DATA((j - 1), 2)
            y__1 = DATA((j - 1), 3)
            x__2 = DATA(j, 2)
            y__2 = DATA(j, 3)
            x__3 = DATA((j + 1), 2)
            y__3 = DATA((j + 1), 3)
            A_1 = x__1 ^ 2 + y__1 ^ 2 - x__2 ^ 2 - y__2 ^ 2
            A_2 = x__1 ^ 2 + y__1 ^ 2 - x__3 ^ 2 - y__3 ^ 2
            X_1 = x__1 - x__2
            Y_1 = y__1 - y__2
            X_2 = x__1 - x__3
            Y_2 = y__1 - y__3

            above_line = (A_2 * Y_1 - A_1 * Y_2)
            below_line = (2 * X_2 * Y_1 - 2 * X_1 * Y_2)
            If below_line <> 0 Then
               x_circle = (above_line / below_line)
            ElseIf below_line = 0 Then
               Print #13, DATA(j - 1, 1), DATA(j, 1), DATA(j + 1, 1), DATA(j - 1, 13), DATA(j, 13), DATA(j + 1, 13)
```

```
            DATA(j, 8) = 0.9999        'Choose 0.9999 in order to find it easier later
            DATA(j, 9) = 9             'Not 1 or 2, but 9 to separate it from the others
            DATA(j, 10) = 1.666        '1.666 to find it easier later
            DATA(j, 14) = 99999        '99999 staan vir MISTAKE
            GoTo skip_bogus_road:
          End If
          If Y_1 <> 0 Then
            y_circle = (A_1 - 2 * X_1 * x_circle) / (2 * Y_1)
          ElseIf Y_1 = 0 And Y_2 <> 0 Then
            y_circle = (A_2 - 2 * X_2 * x_circle) / (2 * Y_2)
          ElseIf Y_1 = 0 And Y_2 = 0 Then
            radius = 9999
          End If

          If DATA(j, 6) >= 0 Then
            DATA(j, 8) = ((x__2 - x_circle) ^ 2 + (y__2 - y_circle) ^ 2) ^ 0.5
            If DATA(j, 8) <= 40 Then
              DATA(j, 9) = 1
            Else
              DATA(j, 9) = 0
            End If
          ElseIf DATA(j, 6) < 0 Then
            sign = DATA(j, 6) / Abs(DATA(j, 6))
            DATA(j, 8) = sign * (((x__2 - x_circle) ^ 2 + (y__2 - y_circle) ^ 2) ^ 0.5)
            If DATA(j, 8) >= -40 Then
              DATA(j, 9) = -1
            Else
              DATA(j, 9) = 0
            End If
          End If

          DATA(j, 10) = Abs(DATA(j, 6) * (Pi / 180) * DATA(j, 8))
          current_distance = DATA(j, 10) + beginning_distance
          total_distance = total_distance + current_distance
          DATA(j, 11) = total_distance
        End If
      End If
skip_bogus_road:
    Next j
    Count_2 = 0
Next i

Close 13

Dim count_offtrack_roads(12000) As Integer

' Print this stuff to a file that I can read in OFFTRACK
For i = 1 To totalroads
    count_offtrack_roads(i) = 0
    DATA6 = "k:\Oberholf\HSU\mf\output\curves_output\rd" + LTrim$(RTrim$(Str$(i)))
    DATA15 = "k:\Oberholf\HSU\mf\output\curves_output_backup\rd" + LTrim$(RTrim$(Str$(i)))
    Open DATA6 For Output As #6
    Open DATA15 For Output As #15
    ' Print data in the following format: "13.2","140.00","160.00"
    ' Also: convert the data to FEET
    ' The distance before and after the roadsection = 100 FEET.
    '   This is longer than twice the distance of the truck
    runup = 100
    Print #6, Format(0, "\""0.00\""") & "," & _
     Format(0, "\""0.00\""") & "," & _
     Format(runup, "\""0.00\""")                    'FEET (100)
    Print #15, Format(0, "\""0.00\""") & "," & _
     Format(0, "\""0.00\""") & "," & _
     Format(runup, "\""0.00\""")                    'FEET (100)
    For j = 1 To total_vector_count
      If DATA(j, 1) = i Then
      count_offtrack_roads(i) = count_offtrack_roads(i) + 1
        If count_offtrack_roads(i) = 2 Then
          Write #6, Format(3.28 * DATA(j, 8), "0.00"); _
           Format(runup, "0.00"); _
```

```
            Format(runup + 3.28 * DATA(j, 11), "0.00")        'FEET (100 + radius)
           Write #15, Format(3.28 * DATA(j, 8), "0.00"); _
            Format(runup, "0.00"); _
            Format(runup + 3.28 * DATA(j, 11), "0.00")        'FEET (100 + radius)
            lastsection = j
         End If
         If count_offtrack_roads(i) > 2 And count_offtrack_roads(i) < cc(i) Then
           Write #6, Format(3.28 * DATA(j, 8), "0.00"); _
            Format(runup + 3.28 * DATA(j - 1, 11), "0.00"); _
            Format(runup + 3.28 * DATA(j, 11), "0.00")        'FEET (radius + previous radius)
           Write #15, Format(3.28 * DATA(j, 8), "0.00"); _
            Format(runup + 3.28 * DATA(j - 1, 11), "0.00"); _
            Format(runup + 3.28 * DATA(j, 11), "0.00")        'FEET (radius + previous radius)
            lastsection = j
         End If
      End If
   Next j
   Write #6, Format(0, "0.00"); _
    Format(runup + 3.28 * DATA(lastsection, 11), "0.00"); _
    Format(5000, "0.00")       'FEET (radius + previous radius)
   Write #15, Format(0, "0.00"); _
    Format(runup + 3.28 * DATA(lastsection, 11), "0.00"); _
    Format(5000, "0.00")       'FEET (radius + previous radius)
   Close #6
   Close #15
Next i

'=================================================
' STEP 11: PRINT ALL DATA TO FULLDATA FILE
'=================================================
For i = 1 To totalroads
   DATA5 = "k:\Oberholf\HSU\mf\output\curves_output\fulldata" + LTrim$(RTrim$(Str$(i)))
   DATA14 = "k:\Oberholf\HSU\mf\output\curves_output_backup\fulldata" + LTrim$(RTrim$(Str$(i)))
   Open DATA5 For Output As #5
   Open DATA14 For Output As #14
   For j = 1 To total_vector_count
      If DATA(j, 1) = i Then
         Print #5, DATA(j, 1), DATA(j, 2), DATA(j, 3), DATA(j, 4), _
            DATA(j, 5), DATA(j, 6), DATA(j, 7), DATA(j, 8), _
            DATA(j, 9), DATA(j, 10), DATA(j, 11), DATA(j, 12), _
            DATA(j, 13)
         Print #14, DATA(j, 1), DATA(j, 2), DATA(j, 3), DATA(j, 4), _
            DATA(j, 5), DATA(j, 6), DATA(j, 7), DATA(j, 8), _
            DATA(j, 9), DATA(j, 10), DATA(j, 11), DATA(j, 12), _
            DATA(j, 13)
      End If
   Next j
Close #5
Close #14
Next i

Kill "k:\Oberholf\HSU\mf\output\curves_output\fw_line*"  'DATA2
Kill "k:\Oberholf\HSU\mf\output\curves_output\fw_ang*"   'DATA3
Kill "k:\Oberholf\HSU\mf\output\curves_output\WEED*"     'DATA4
Kill "k:\Oberholf\HSU\mf\output\curves_output\combcurv*" 'DATA8
Kill "k:\Oberholf\HSU\mf\output\curves_output\comcvang*" 'DATA9

End Sub
```

APPENDIX B

```
Option Base 1
Private Sub Form_Load()

'Some of the files are longer than 100 nodes long.
'These files I have to break into sections shorter
'than 100 nodes.
Dim SPLIT_DATA(7000, 3) As Double
Dim SPLIT_FULLDATA(7000, 14) As Double
Dim smallest_number(10000) As Integer
Dim distance(10000) As Integer
Dim number_of_rds(10000) As Integer
Dim number_of_fulldatas(10000) As Integer


Dim count_offtrack_roads(20000) As Double
Dim DATA6 As String
Dim DATA5 As String
Dim DATA7 As String
Dim DATA8 As String
Dim DATA9 As String
Dim DATA10 As String

'Get this value from the "CURVES" program
totalroads = 7873

new_number_of_roads = totalroads

For i = 1 To totalroads
   smallest_number(i) = 999
   distance(i) = 4900
Next i

For i = 1 To totalroads
   count_offtrack_roads(i) = 0
   DATA6 = "k:\oberholf\hsu\mf\output\curves_output\rd" + LTrim$(RTrim$(Str$(i)))
   Open DATA6 For Input As #6
      Do While Not EOF(6)
         count_offtrack_roads(i) = count_offtrack_roads(i) + 1
         Input #6, offt_radius, offt_dist, offt_cum_dist
         little_number = Abs(offt_radius)
         max_distance = offt_dist
         If little_number < smallest_number(i) And little_number <> 0 Then
            smallest_number(i) = little_number
         End If
         If max_distance > distance(i) Then
            distance(i) = max_distance
            count_long_dist = count_long_dist + 1
         End If
      Loop
   Close #6
Next i

For i = 1 To totalroads

'TWO:   If the road is longer than 100 sections, break it into TWO sections
   If count_offtrack_roads(i) >= 100 And count_offtrack_roads(i) < 200 Then

      'Put all the RD and FULLDATA files into an array first
      '=====================================================

      'RD files - array
      count_offt_roads = 0
      DATA6 = "k:\oberholf\hsu\mf\output\curves_output\rd" + LTrim$(RTrim$(Str$(i)))
      Open DATA6 For Input As #6
         Do While Not EOF(6)
            Input #6, offt_radius, offt_dist, offt_cum_dist
            count_offt_roads = count_offt_roads + 1
```

```
      SPLIT_DATA(count_offt_roads, 1) = offt_radius
      SPLIT_DATA(count_offt_roads, 2) = offt_dist
      SPLIT_DATA(count_offt_roads, 3) = offt_cum_dist
   Loop
Close #6

'FULLDATA files - array
count_offt_roads = 0
DATA5 = "k:\oberholf\hsu\mf\output\curves_output\fulldata" + LTrim$(RTrim$(Str$(i)))
Open DATA5 For Input As #5
   Do While Not EOF(5)
      Input #5, FD_id, FD_X, FD_Y, FD_dist, FD_angle, FD_anglediff, _
            FD_junction, FD_radius, FD_column9, FD_curve_length, _
            FD_cloumn11, FD_column12, FD_column13
      count_offt_roads = count_offt_roads + 1
      SPLIT_FULLDATA(count_offt_roads, 1) = FD_id
      SPLIT_FULLDATA(count_offt_roads, 2) = FD_X
      SPLIT_FULLDATA(count_offt_roads, 3) = FD_Y
      SPLIT_FULLDATA(count_offt_roads, 4) = FD_dist
      SPLIT_FULLDATA(count_offt_roads, 5) = FD_angle
      SPLIT_FULLDATA(count_offt_roads, 6) = FD_anglediff
      SPLIT_FULLDATA(count_offt_roads, 7) = FD_junction
      SPLIT_FULLDATA(count_offt_roads, 8) = FD_radius
      SPLIT_FULLDATA(count_offt_roads, 9) = FD_column9
      SPLIT_FULLDATA(count_offt_roads, 10) = FD_curve_length
      SPLIT_FULLDATA(count_offt_roads, 11) = FD_cloumn11
      SPLIT_FULLDATA(count_offt_roads, 12) = FD_column12
      SPLIT_FULLDATA(count_offt_roads, 13) = FD_column13
   Loop
Close #5


'Write new files that would replace the longer files
'================================================
'Make a backup file of the FULLDATA and RD files in another directory
DATA7 = "k:\oberholf\hsu\mf\output\adj_curves_output\rd" + LTrim$(RTrim$(Str$(i)))
Open DATA7 For Output As #7
   For j = 1 To count_offt_roads
      Write #7, SPLIT_DATA(j, 1), SPLIT_DATA(j, 2), SPLIT_DATA(j, 3)
   Next j
Close #7
DATA8 = "k:\oberholf\hsu\mf\output\adj_curves_output\fulldata" + LTrim$(RTrim$(Str$(i)))
Open DATA8 For Output As #8
   For j = 1 To count_offt_roads
      Write #8, SPLIT_FULLDATA(j, 1), SPLIT_FULLDATA(j, 2), SPLIT_FULLDATA(j, 3), _
            SPLIT_FULLDATA(j, 4), SPLIT_FULLDATA(j, 5), SPLIT_FULLDATA(j, 6), _
            SPLIT_FULLDATA(j, 7), SPLIT_FULLDATA(j, 8), SPLIT_FULLDATA(j, 9), _
            SPLIT_FULLDATA(j, 10), SPLIT_FULLDATA(j, 11), SPLIT_FULLDATA(j, 12), _
            SPLIT_FULLDATA(j, 13)
   Next j
Close #8

'FILE 1: 1 - 99
DATA7 = "k:\oberholf\hsu\mf\output\curves_output\rd" + LTrim$(RTrim$(Str$(i)))
Open DATA7 For Output As #7
   For j = 1 To 98
      Write #7, Format(SPLIT_DATA(j, 1), "0.00"), Format(SPLIT_DATA(j, 2), "0.00"), Format(SPLIT_DATA(j, 3), "0.00")
      lastsection = j
   Next j
   Write #7, Format(0, "0.00"), _
         Format(SPLIT_DATA(lastsection, 3), "0.00"), _
         Format(5000, "0.00")
Close #7
DATA8 = "k:\oberholf\hsu\mf\output\curves_output\fulldata" + LTrim$(RTrim$(Str$(i)))
Open DATA8 For Output As #8
   For j = 1 To 99
      Write #8, SPLIT_FULLDATA(j, 1), SPLIT_FULLDATA(j, 2), SPLIT_FULLDATA(j, 3), _
            SPLIT_FULLDATA(j, 4), SPLIT_FULLDATA(j, 5), SPLIT_FULLDATA(j, 6), _
            SPLIT_FULLDATA(j, 7), SPLIT_FULLDATA(j, 8), SPLIT_FULLDATA(j, 9), _
            SPLIT_FULLDATA(j, 10), SPLIT_FULLDATA(j, 11), SPLIT_FULLDATA(j, 12), _
```

```
            SPLIT_FULLDATA(j, 13)
        Next j
    Close #8

    'FILE 2: 100 - 199
    new_number_of_roads = new_number_of_roads + 1
    DATA7 = "k:\oberholf\hsu\mf\output\curves_output\rd" + LTrim$(RTrim$(Str$(new_number_of_roads)))
    Open DATA7 For Output As #7
        number_of_nodes = count_offtrack_roads(i)
        startpoint = count_offt_roads - 98
        Write #7, Format(0, "0.00"), Format(0, "0.00"), Format(100, "0.00")
        For j = startpoint To (number_of_nodes - 1)
            Write #7, Format(SPLIT_DATA(j, 1), "0.00"), _
                Format((SPLIT_DATA(j, 2) - SPLIT_DATA(startpoint, 2) + 100), "0.00"), _
                Format((SPLIT_DATA(j, 3) - SPLIT_DATA(startpoint, 2) + 100), "0.00")
        Next j
        Write #7, Format(0, "0.00"), _
            Format((SPLIT_DATA(number_of_nodes, 2) - SPLIT_DATA(startpoint, 2) + 100), "0.00"), _
            Format(5000, "0.00")
    Close #7
     DATA8 = "k:\oberholf\hsu\mf\output\curves_output\fulldata" + LTrim$(RTrim$(Str$(new_number_of_roads)))
    Open DATA8 For Output As #8
        For j = startpoint To number_of_nodes
            Write #8, SPLIT_FULLDATA(j, 1), SPLIT_FULLDATA(j, 2), SPLIT_FULLDATA(j, 3), _
                SPLIT_FULLDATA(j, 4), SPLIT_FULLDATA(j, 5), SPLIT_FULLDATA(j, 6), _
                SPLIT_FULLDATA(j, 7), SPLIT_FULLDATA(j, 8), SPLIT_FULLDATA(j, 9), _
                SPLIT_FULLDATA(j, 10), SPLIT_FULLDATA(j, 11), SPLIT_FULLDATA(j, 12), _
                SPLIT_FULLDATA(j, 13)
        Next j
    Close #8
  End If

'THREE: If the road is longer than 200 sections, break it into THREE sections
  If count_offtrack_roads(i) >= 200 And count_offtrack_roads(i) < 300 Then

    'Put all the RD and FULLDATA files into an array first
    '=================================================

    'RD files - array
    count_offt_roads = 0
    DATA6 = "k:\oberholf\hsu\mf\output\curves_output\rd" + LTrim$(RTrim$(Str$(i)))
    Open DATA6 For Input As #6
        Do While Not EOF(6)
            Input #6, offt_radius, offt_dist, offt_cum_dist
            count_offt_roads = count_offt_roads + 1
            SPLIT_DATA(count_offt_roads, 1) = offt_radius
            SPLIT_DATA(count_offt_roads, 2) = offt_dist
            SPLIT_DATA(count_offt_roads, 3) = offt_cum_dist
        Loop
    Close #6

    'FULLDATA files - array
    count_offt_roads = 0
    DATA5 = "k:\oberholf\hsu\mf\output\curves_output\fulldata" + LTrim$(RTrim$(Str$(i)))
    Open DATA5 For Input As #5
        Do While Not EOF(5)
            Input #5, FD_id, FD_X, FD_Y, FD_dist, FD_angle, FD_anglediff, _
                FD_junction, FD_radius, FD_column9, FD_curve_length, _
                FD_cloumn11, FD_column12, FD_column13
            count_offt_roads = count_offt_roads + 1
            SPLIT_FULLDATA(count_offt_roads, 1) = FD_id
            SPLIT_FULLDATA(count_offt_roads, 2) = FD_X
            SPLIT_FULLDATA(count_offt_roads, 3) = FD_Y
            SPLIT_FULLDATA(count_offt_roads, 4) = FD_dist
            SPLIT_FULLDATA(count_offt_roads, 5) = FD_angle
            SPLIT_FULLDATA(count_offt_roads, 6) = FD_anglediff
            SPLIT_FULLDATA(count_offt_roads, 7) = FD_junction
            SPLIT_FULLDATA(count_offt_roads, 8) = FD_radius
            SPLIT_FULLDATA(count_offt_roads, 9) = FD_column9
            SPLIT_FULLDATA(count_offt_roads, 10) = FD_curve_length
```

```
      SPLIT_FULLDATA(count_offt_roads, 11) = FD_cloumn11
      SPLIT_FULLDATA(count_offt_roads, 12) = FD_column12
      SPLIT_FULLDATA(count_offt_roads, 13) = FD_column13
   Loop
Close #5

'Write new files that would replace the longer files
'=============================================

'Make a backup file of the FULLDATA and RD files in another directory
DATA7 = "k:\oberholf\hsu\mf\output\adj_curves_output\rd" + LTrim$(RTrim$(Str$(i)))
Open DATA7 For Output As #7
   For j = 1 To count_offt_roads
      Write #7, Format(SPLIT_DATA(j, 1), "0.00"), Format(SPLIT_DATA(j, 2), "0.00"), Format(SPLIT_DATA(j, 3), "0.00")
   Next j
Close #7
DATA8 = "k:\oberholf\hsu\mf\output\adj_curves_output\fulldata" + LTrim$(RTrim$(Str$(i)))
Open DATA8 For Output As #8
   For j = 1 To count_offt_roads
      Write #8, SPLIT_FULLDATA(j, 1), SPLIT_FULLDATA(j, 2), SPLIT_FULLDATA(j, 3), _
            SPLIT_FULLDATA(j, 4), SPLIT_FULLDATA(j, 5), SPLIT_FULLDATA(j, 6), _
            SPLIT_FULLDATA(j, 7), SPLIT_FULLDATA(j, 8), SPLIT_FULLDATA(j, 9), _
            SPLIT_FULLDATA(j, 10), SPLIT_FULLDATA(j, 11), SPLIT_FULLDATA(j, 12), _
            SPLIT_FULLDATA(j, 13)
   Next j
Close #8

'FILE 1: 1 - 99
DATA7 = "k:\oberholf\hsu\mf\output\curves_output\rd" + LTrim$(RTrim$(Str$(i)))
Open DATA7 For Output As #7
   For j = 1 To 98
      Write #7, Format(SPLIT_DATA(j, 1), "0.00"), Format(SPLIT_DATA(j, 2), "0.00"), Format(SPLIT_DATA(j, 3), "0.00")
      lastsection = j
   Next j
   Write #7, Format(0, "0.00"), _
         Format(SPLIT_DATA(lastsection, 3), "0.00"), _
         Format(5000, "0.00")
Close #7
DATA8 = "k:\oberholf\hsu\mf\output\curves_output\fulldata" + LTrim$(RTrim$(Str$(i)))
Open DATA8 For Output As #8
   For j = 1 To 99
      Write #8, SPLIT_FULLDATA(j, 1), SPLIT_FULLDATA(j, 2), SPLIT_FULLDATA(j, 3), _
            SPLIT_FULLDATA(j, 4), SPLIT_FULLDATA(j, 5), SPLIT_FULLDATA(j, 6), _
            SPLIT_FULLDATA(j, 7), SPLIT_FULLDATA(j, 8), SPLIT_FULLDATA(j, 9), _
            SPLIT_FULLDATA(j, 10), SPLIT_FULLDATA(j, 11), SPLIT_FULLDATA(j, 12), _
            SPLIT_FULLDATA(j, 13)
   Next j
Close #8

'FILE 2: 100 - 199
new_number_of_roads = new_number_of_roads + 1
DATA7 = "k:\oberholf\hsu\mf\output\curves_output\rd" + LTrim$(RTrim$(Str$(new_number_of_roads)))
Open DATA7 For Output As #7
   Write #7, Format(0, "0.00"), Format(0, "0.00"), Format(100, "0.00")
   For j = 100 To 198
      Write #7, Format(SPLIT_DATA(j, 1), "0.00"), _
            Format((SPLIT_DATA(j, 2) - SPLIT_DATA(100, 2) + 100), "0.00"), _
            Format((SPLIT_DATA(j, 3) - SPLIT_DATA(100, 2) + 100), "0.00")
      lastsection = j
   Next j
   Write #7, Format(0, "0.00"), _
         Format((SPLIT_DATA(lastsection, 3) - SPLIT_DATA(100, 2) + 100), "0.00"), _
         Format(5000, "0.00")
Close #7
DATA8 = "k:\oberholf\hsu\mf\output\curves_output\fulldata" + LTrim$(RTrim$(Str$(new_number_of_roads)))
Open DATA8 For Output As #8
   For j = 100 To 199
      Write #8, SPLIT_FULLDATA(j, 1), SPLIT_FULLDATA(j, 2), SPLIT_FULLDATA(j, 3), _
            SPLIT_FULLDATA(j, 4), SPLIT_FULLDATA(j, 5), SPLIT_FULLDATA(j, 6), _
            SPLIT_FULLDATA(j, 7), SPLIT_FULLDATA(j, 8), SPLIT_FULLDATA(j, 9), _
```

```
            SPLIT_FULLDATA(j, 10), SPLIT_FULLDATA(j, 11), SPLIT_FULLDATA(j, 12), _
            SPLIT_FULLDATA(j, 13)
      Next j
   Close #8

   'FILE 3: 200 - 299
   new_number_of_roads = new_number_of_roads + 1
   DATA7 = "k:\oberholf\hsu\mf\output\curves_output\rd" + LTrim$(RTrim$(Str$(new_number_of_roads)))
   Open DATA7 For Output As #7
      number_of_nodes = count_offtrack_roads(i)
      startpoint = count_offt_roads - 98
      Write #7, Format(0, "0.00"), Format(0, "0.00"), Format(100, "0.00")
      For j = startpoint To (number_of_nodes - 1)
         Write #7, Format(SPLIT_DATA(j, 1), "0.00"), _
               Format((SPLIT_DATA(j, 2) - SPLIT_DATA(startpoint, 2) + 100), "0.00"), _
               Format((SPLIT_DATA(j, 3) - SPLIT_DATA(startpoint, 2) + 100), "0.00")
      Next j
      Write #7, Format(0, "0.00"), _
            Format((SPLIT_DATA(number_of_nodes, 3) - SPLIT_DATA(startpoint, 2) + 100), "0.00"), _
            Format(5000, "0.00")
   Close #7
    DATA8 = "k:\oberholf\hsu\mf\output\curves_output\fulldata" + LTrim$(RTrim$(Str$(new_number_of_roads)))
   Open DATA8 For Output As #8
      For j = startpoint To number_of_nodes
         Write #8, SPLIT_FULLDATA(j, 1), SPLIT_FULLDATA(j, 2), SPLIT_FULLDATA(j, 3), _
               SPLIT_FULLDATA(j, 4), SPLIT_FULLDATA(j, 5), SPLIT_FULLDATA(j, 6), _
               SPLIT_FULLDATA(j, 7), SPLIT_FULLDATA(j, 8), SPLIT_FULLDATA(j, 9), _
               SPLIT_FULLDATA(j, 10), SPLIT_FULLDATA(j, 11), SPLIT_FULLDATA(j, 12), _
               SPLIT_FULLDATA(j, 13)
      Next j
   Close #8
End If
'-------------------------------------------------------------------------------------------------------------
'FOUR:  If the road is longer than 300 sections, break it into FOUR sections
If count_offtrack_roads(i) >= 300 And count_offtrack_roads(i) < 400 Then

   'Put all the RD and FULLDATA files into an array first
   '=================================================

   'RD files - array
   count_offt_roads = 0
   DATA6 = "k:\oberholf\hsu\mf\output\curves_output\rd" + LTrim$(RTrim$(Str$(i)))
   Open DATA6 For Input As #6
      Do While Not EOF(6)
         Input #6, offt_radius, offt_dist, offt_cum_dist
         count_offt_roads = count_offt_roads + 1
         SPLIT_DATA(count_offt_roads, 1) = offt_radius
         SPLIT_DATA(count_offt_roads, 2) = offt_dist
         SPLIT_DATA(count_offt_roads, 3) = offt_cum_dist
      Loop
   Close #6

   'FULLDATA files - array
   count_offt_roads = 0
   DATA5 = "k:\oberholf\hsu\mf\output\curves_output\fulldata" + LTrim$(RTrim$(Str$(i)))
   Open DATA5 For Input As #5
      Do While Not EOF(5)
         Input #5, FD_id, FD_X, FD_Y, FD_dist, FD_angle, FD_anglediff, _
               FD_junction, FD_radius, FD_column9, FD_curve_length, _
               FD_cloumn11, FD_column12, FD_column13
         count_offt_roads = count_offt_roads + 1
         SPLIT_FULLDATA(count_offt_roads, 1) = FD_id
         SPLIT_FULLDATA(count_offt_roads, 2) = FD_X
         SPLIT_FULLDATA(count_offt_roads, 3) = FD_Y
         SPLIT_FULLDATA(count_offt_roads, 4) = FD_dist
         SPLIT_FULLDATA(count_offt_roads, 5) = FD_angle
         SPLIT_FULLDATA(count_offt_roads, 6) = FD_anglediff
         SPLIT_FULLDATA(count_offt_roads, 7) = FD_junction
         SPLIT_FULLDATA(count_offt_roads, 8) = FD_radius
         SPLIT_FULLDATA(count_offt_roads, 9) = FD_column9
```

```
    SPLIT_FULLDATA(count_offt_roads, 10) = FD_curve_length
    SPLIT_FULLDATA(count_offt_roads, 11) = FD_cloumn11
    SPLIT_FULLDATA(count_offt_roads, 12) = FD_column12
    SPLIT_FULLDATA(count_offt_roads, 13) = FD_column13
  Loop
Close #5

'Write new files that would replace the longer files
'=========================================
'Make a backup file of the FULLDATA and RD files in another directory
DATA7 = "k:\oberholf\hsu\mf\output\adj_curves_output\rd" + LTrim$(RTrim$(Str$(i)))
Open DATA7 For Output As #7
  For j = 1 To count_offt_roads
    Write #7, SPLIT_DATA(j, 1), SPLIT_DATA(j, 2), SPLIT_DATA(j, 3)
  Next j
Close #7
DATA8 = "k:\oberholf\hsu\mf\output\adj_curves_output\fulldata" + LTrim$(RTrim$(Str$(i)))
Open DATA8 For Output As #8
  For j = 1 To count_offt_roads
    Write #8, SPLIT_FULLDATA(j, 1), SPLIT_FULLDATA(j, 2), SPLIT_FULLDATA(j, 3), _
        SPLIT_FULLDATA(j, 4), SPLIT_FULLDATA(j, 5), SPLIT_FULLDATA(j, 6), _
        SPLIT_FULLDATA(j, 7), SPLIT_FULLDATA(j, 8), SPLIT_FULLDATA(j, 9), _
        SPLIT_FULLDATA(j, 10), SPLIT_FULLDATA(j, 11), SPLIT_FULLDATA(j, 12), _
        SPLIT_FULLDATA(j, 13)
  Next j
Close #8

'FILE 1: 1 - 99
DATA7 = "k:\oberholf\hsu\mf\output\curves_output\rd" + LTrim$(RTrim$(Str$(i)))
Open DATA7 For Output As #7
  For j = 1 To 98
    Write #7, Format(SPLIT_DATA(j, 1), "0.00"), Format(SPLIT_DATA(j, 2), "0.00"), Format(SPLIT_DATA(j, 3), "0.00")
    lastsection = j
  Next j
  Write #7, Format(0, "0.00"), _
      Format(SPLIT_DATA(lastsection, 3), "0.00"), _
      Format(5000, "0.00")
Close #7
DATA8 = "k:\oberholf\hsu\mf\output\curves_output\fulldata" + LTrim$(RTrim$(Str$(i)))
Open DATA8 For Output As #8
  For j = 1 To 99
    Write #8, SPLIT_FULLDATA(j, 1), SPLIT_FULLDATA(j, 2), SPLIT_FULLDATA(j, 3), _
        SPLIT_FULLDATA(j, 4), SPLIT_FULLDATA(j, 5), SPLIT_FULLDATA(j, 6), _
        SPLIT_FULLDATA(j, 7), SPLIT_FULLDATA(j, 8), SPLIT_FULLDATA(j, 9), _
        SPLIT_FULLDATA(j, 10), SPLIT_FULLDATA(j, 11), SPLIT_FULLDATA(j, 12), _
        SPLIT_FULLDATA(j, 13)
  Next j
Close #8

'FILE 2: 100 - 199
new_number_of_roads = new_number_of_roads + 1
DATA7 = "k:\oberholf\hsu\mf\output\curves_output\rd" + LTrim$(RTrim$(Str$(new_number_of_roads)))
Open DATA7 For Output As #7
  Write #7, Format(0, "0.00"), Format(0, "0.00"), Format(100, "0.00")
  For j = 100 To 198
    Write #7, Format(SPLIT_DATA(j, 1), "0.00"), _
        Format((SPLIT_DATA(j, 2) - SPLIT_DATA(100, 2) + 100), "0.00"), _
        Format((SPLIT_DATA(j, 3) - SPLIT_DATA(100, 2) + 100), "0.00")
    lastsection = j
  Next j
  Write #7, Format(0, "0.00"), _
      Format((SPLIT_DATA(lastsection, 3) - SPLIT_DATA(100, 2) + 100), "0.00"), _
      Format(5000, "0.00")
Close #7
DATA8 = "k:\oberholf\hsu\mf\output\curves_output\fulldata" + LTrim$(RTrim$(Str$(new_number_of_roads)))
Open DATA8 For Output As #8
  For j = 100 To 199
    Write #8, SPLIT_FULLDATA(j, 1), SPLIT_FULLDATA(j, 2), SPLIT_FULLDATA(j, 3), _
        SPLIT_FULLDATA(j, 4), SPLIT_FULLDATA(j, 5), SPLIT_FULLDATA(j, 6), _
        SPLIT_FULLDATA(j, 7), SPLIT_FULLDATA(j, 8), SPLIT_FULLDATA(j, 9), _
```

```
            SPLIT_FULLDATA(j, 10), SPLIT_FULLDATA(j, 11), SPLIT_FULLDATA(j, 12), _
            SPLIT_FULLDATA(j, 13)
      Next j
   Close #8

   'FILE 3: 200 - 299
   new_number_of_roads = new_number_of_roads + 1
   DATA7 = "k:\oberholf\hsu\mf\output\curves_output\rd" + LTrim$(RTrim$(Str$(new_number_of_roads)))
   Open DATA7 For Output As #7
      Write #7, Format(0, "0.00"), Format(0, "0.00"), Format(100, "0.00")
      For j = 200 To 298
         Write #7, Format(SPLIT_DATA(j, 1), "0.00"), _
               Format((SPLIT_DATA(j, 2) - SPLIT_DATA(200, 2) + 100), "0.00"), _
               Format((SPLIT_DATA(j, 3) - SPLIT_DATA(200, 2) + 100), "0.00")
         lastsection = j
      Next j
      Write #7, Format(0, "0.00"), _
            Format((SPLIT_DATA(lastsection, 3) - SPLIT_DATA(200, 2) + 100), "0.00"), _
            Format(5000, "0.00")
   Close #7
   DATA8 = "k:\oberholf\hsu\mf\output\curves_output\fulldata" + LTrim$(RTrim$(Str$(new_number_of_roads)))
   Open DATA8 For Output As #8
      For j = 200 To 299
         Write #8, SPLIT_FULLDATA(j, 1), SPLIT_FULLDATA(j, 2), SPLIT_FULLDATA(j, 3), _
               SPLIT_FULLDATA(j, 4), SPLIT_FULLDATA(j, 5), SPLIT_FULLDATA(j, 6), _
               SPLIT_FULLDATA(j, 7), SPLIT_FULLDATA(j, 8), SPLIT_FULLDATA(j, 9), _
               SPLIT_FULLDATA(j, 10), SPLIT_FULLDATA(j, 11), SPLIT_FULLDATA(j, 12), _
               SPLIT_FULLDATA(j, 13)
      Next j
   Close #8

   'FILE 3: 300 - 399
   new_number_of_roads = new_number_of_roads + 1
   DATA7 = "k:\oberholf\hsu\mf\output\curves_output\rd" + LTrim$(RTrim$(Str$(new_number_of_roads)))
   Open DATA7 For Output As #7
      number_of_nodes = count_offtrack_roads(i)
      startpoint = count_offt_roads - 98
      Write #7, Format(0, "0.00"), Format(0, "0.00"), Format(100, "0.00")
      For j = startpoint To (number_of_nodes - 1)
         Write #7, Format(SPLIT_DATA(j, 1), "0.00"), _
               Format((SPLIT_DATA(j, 2) - SPLIT_DATA(startpoint, 2) + 100), "0.00"), _
               Format((SPLIT_DATA(j, 3) - SPLIT_DATA(startpoint, 2) + 100), "0.00")
      Next j
      Write #7, Format(0, "0.00"), _
            Format((SPLIT_DATA(number_of_nodes, 3) - SPLIT_DATA(startpoint, 2) + 100), "0.00"), _
            Format(5000, "0.00")
   Close #7
    DATA8 = "k:\oberholf\hsu\mf\output\curves_output\fulldata" + LTrim$(RTrim$(Str$(new_number_of_roads)))
   Open DATA8 For Output As #8
      For j = startpoint To number_of_nodes
         Write #8, SPLIT_FULLDATA(j, 1), SPLIT_FULLDATA(j, 2), SPLIT_FULLDATA(j, 3), _
               SPLIT_FULLDATA(j, 4), SPLIT_FULLDATA(j, 5), SPLIT_FULLDATA(j, 6), _
               SPLIT_FULLDATA(j, 7), SPLIT_FULLDATA(j, 8), SPLIT_FULLDATA(j, 9), _
               SPLIT_FULLDATA(j, 10), SPLIT_FULLDATA(j, 11), SPLIT_FULLDATA(j, 12), _
               SPLIT_FULLDATA(j, 13)
      Next j
   Close #8
End If
'-------------------------------------------------------------------------------------------------
'FIVE:  If the road is longer than 400 sections, break it into FIVE sections
If count_offtrack_roads(i) >= 400 Then
   Stop
End If
Next i

'-------------------------------------------------------------------------------------------------
'SUB routine - those files that have radii < 20 feet should not be used in Offtrack
'The offtrack program sometimes have difficulty with calculating offtracking when
'the radii involved are less than the length of the truck. These files are written
'to a separate directory, where I will feed them into OFFTRACK manually.
```

```
specified_radius = 20

For i = 1 To totalroads
   smallest_number(i) = 999
   distance(i) = 4999
Next i

For i = 1 To new_number_of_roads
   count_offtrack_roads(i) = 0
   DATA7 = "k:\oberholf\hsu\mf\output\curves_output\rd" + LTrim$(RTrim$(Str$(i)))
   Open DATA7 For Input As #7
      Do While Not EOF(7)
         count_offtrack_roads(i) = count_offtrack_roads(i) + 1
         Input #7, offt_radius, offt_dist, offt_cum_dist
         number_of_rds(i) = number_of_rds(i) + 1
         little_number = Abs(offt_radius)
         max_distance = offt_dist
         If little_number < smallest_number(i) And little_number <> 0 Then
            smallest_number(i) = little_number
         End If
         If max_distance > distance(i) Then
            distance(i) = max_distance
            count_long_dist = count_long_dist + 1
         End If
      Loop
   Close #7
Next i

Dim LONG_SECTIONS(10000, 3) As Double
Dim LONG_FULLDATA(10000, 13) As Double
Dim count_long_roads_ind(10000) As Integer
Dim count_long_fulldata_ind(10000) As Integer

For i = 1 To new_number_of_roads
   If distance(i) > 4999 Then
      DATA7 = "k:\oberholf\hsu\mf\output\curves_output\rd" + LTrim$(RTrim$(Str$(i)))
      Open DATA7 For Input As #7
         count_long_roads = 0
         Do While Not EOF(7)
            count_long_roads_ind(i) = count_long_roads_ind(i) + 1
            count_long_roads = count_long_roads + 1
            Input #7, radius, firstsection, secondsection
            LONG_SECTIONS(count_long_roads, 1) = radius
            LONG_SECTIONS(count_long_roads, 2) = firstsection
            LONG_SECTIONS(count_long_roads, 3) = secondsection
         Loop
      Close #7

      DATA8 = "k:\oberholf\hsu\mf\output\curves_output\fulldata" + LTrim$(RTrim$(Str$(i)))
      Open DATA8 For Input As #8
         count_long_fulldata = 0
         Do While Not EOF(8)
            count_long_fulldata_ind(i) = count_long_fulldata_ind(i) + 1
            count_long_fulldata = count_long_fulldata + 1
            Input #8, col1, col2, col3, col4, col5, col6, col7, col8, col9, col10, col11, col12, col13
            LONG_FULLDATA(count_long_fulldata, 1) = col1
            LONG_FULLDATA(count_long_fulldata, 2) = col2
            LONG_FULLDATA(count_long_fulldata, 3) = col3
            LONG_FULLDATA(count_long_fulldata, 4) = col4
            LONG_FULLDATA(count_long_fulldata, 5) = col5
            LONG_FULLDATA(count_long_fulldata, 6) = col6
            LONG_FULLDATA(count_long_fulldata, 7) = col7
            LONG_FULLDATA(count_long_fulldata, 8) = col8
            LONG_FULLDATA(count_long_fulldata, 9) = col9
            LONG_FULLDATA(count_long_fulldata, 10) = co110
            LONG_FULLDATA(count_long_fulldata, 11) = col11
            LONG_FULLDATA(count_long_fulldata, 12) = col12
            LONG_FULLDATA(count_long_fulldata, 13) = col13
         Loop
```

```
Close #8

DATA7 = "k:\oberholf\hsu\mf\output\adj_curves_output\rd" + LTrim$(RTrim$(Str$(i)))
Open DATA7 For Output As #7
   For j = 1 To count_offt_roads
      Write #7, Format(LONG_SECTIONS(j, 1), "0.00"), Format(LONG_SECTIONS(j, 2), "0.00"),
                        Format(LONG_SECTIONS(j, 3), "0.00")
   Next j
Close #7
DATA8 = "k:\oberholf\hsu\mf\output\adj_curves_output\fulldata" + LTrim$(RTrim$(Str$(i)))
Open DATA8 For Output As #8
   For j = 1 To count_offt_roads
      Write #8, LONG_FULLDATA(j, 1), LONG_FULLDATA(j, 2), LONG_FULLDATA(j, 3), _
            LONG_FULLDATA(j, 4), LONG_FULLDATA(j, 5), LONG_FULLDATA(j, 6), _
            LONG_FULLDATA(j, 7), LONG_FULLDATA(j, 8), LONG_FULLDATA(j, 9), _
            LONG_FULLDATA(j, 10), LONG_FULLDATA(j, 11), LONG_FULLDATA(j, 12), _
            LONG_FULLDATA(j, 13)
   Next j
Close #8

'FILE 1: 1 - halfway
DATA7 = "k:\oberholf\hsu\mf\output\curves_output\rd" + LTrim$(RTrim$(Str$(i)))
Open DATA7 For Output As #7
   halfway = Int((count_long_roads_ind(i)) / 2) + 2
   For j = 1 To halfway
      Write #7, Format(LONG_SECTIONS(j, 1), "0.00"), Format(LONG_SECTIONS(j, 2), "0.00"),
                        Format(LONG_SECTIONS(j, 3), "0.00")
      lastsection = j
   Next j
   Write #7, Format(0, "0.00"), _
         Format(LONG_SECTIONS(lastsection, 3), "0.00"), _
         Format(5000, "0.00")
Close #7
DATA8 = "k:\oberholf\hsu\mf\output\curves_output\fulldata" + LTrim$(RTrim$(Str$(i)))
Open DATA8 For Output As #8
   For j = 1 To halfway
      Write #8, LONG_FULLDATA(j, 1), LONG_FULLDATA(j, 2), LONG_FULLDATA(j, 3), _
            LONG_FULLDATA(j, 4), LONG_FULLDATA(j, 5), LONG_FULLDATA(j, 6), _
            LONG_FULLDATA(j, 7), LONG_FULLDATA(j, 8), LONG_FULLDATA(j, 9), _
            LONG_FULLDATA(j, 10), LONG_FULLDATA(j, 11), LONG_FULLDATA(j, 12), _
            LONG_FULLDATA(j, 13)
   Next j
Close #8

'FILE 2: halfway - end
new_number_of_roads = new_number_of_roads + 1
DATA7 = "k:\oberholf\hsu\mf\output\curves_output\rd" + LTrim$(RTrim$(Str$(new_number_of_roads)))
Open DATA7 For Output As #7
   number_of_nodes = count_long_roads_ind(i)
   startpoint = count_long_roads - halfway
   Write #7, Format(0, "0.00"), Format(0, "0.00"), Format(100, "0.00")
   For j = startpoint To (number_of_nodes - 1)
      Write #7, Format(LONG_SECTIONS(j, 1), "0.00"), _
            Format((LONG_SECTIONS(j, 2) - LONG_SECTIONS(startpoint, 2) + 100), "0.00"), _
            Format((LONG_SECTIONS(j, 3) - LONG_SECTIONS(startpoint, 2) + 100), "0.00")
   Next j
   Write #7, Format(0, "0.00"), _
         Format((LONG_SECTIONS(number_of_nodes, 2) - LONG_SECTIONS(startpoint, 2) + 100), "0.00"), _
         Format(5000, "0.00")
Close #7
 DATA8 = "k:\oberholf\hsu\mf\output\curves_output\fulldata" + LTrim$(RTrim$(Str$(new_number_of_roads)))
Open DATA8 For Output As #8
   For j = startpoint To number_of_nodes
      Write #8, LONG_FULLDATA(j, 1), LONG_FULLDATA(j, 2), LONG_FULLDATA(j, 3), _
            LONG_FULLDATA(j, 4), LONG_FULLDATA(j, 5), LONG_FULLDATA(j, 6), _
            LONG_FULLDATA(j, 7), LONG_FULLDATA(j, 8), LONG_FULLDATA(j, 9), _
            LONG_FULLDATA(j, 10), LONG_FULLDATA(j, 11), LONG_FULLDATA(j, 12), _
            LONG_FULLDATA(j, 13)
   Next j
Close #8
```

```
   End If
Next i

For i = 1 To new_number_of_roads
   smallest_number(i) = 999
   distance(i) = 4999
Next i

count_long_dist = 0
For i = 1 To new_number_of_roads
   count_offtrack_roads(i) = 0
   DATA7 = "k:\oberholf\hsu\mf\output\curves_output\rd" + LTrim$(RTrim$(Str$(i)))
   Open DATA7 For Input As #7
      Do While Not EOF(7)
         count_offtrack_roads(i) = count_offtrack_roads(i) + 1
         Input #7, offt_radius, offt_dist, offt_cum_dist
         number_of_rds(i) = number_of_rds(i) + 1
         little_number = Abs(offt_radius)
         max_distance = offt_dist
         If little_number < smallest_number(i) And little_number <> 0 Then
            smallest_number(i) = little_number
         End If
         If max_distance > distance(i) Then
            distance(i) = max_distance
            count_long_dist = count_long_dist + 1
         End If
      Loop
   Close #7
Next i

count_long_dist = 0
For i = 1 To new_number_of_roads
   count_offtrack_roads(i) = 0
   DATA7 = "k:\oberholf\hsu\mf\output\curves_output\rd" + LTrim$(RTrim$(Str$(i)))
   Open DATA7 For Input As #7
      Do While Not EOF(7)
         count_offtrack_roads(i) = count_offtrack_roads(i) + 1
         Input #7, offt_radius, offt_dist, offt_cum_dist
         number_of_rds(i) = number_of_rds(i) + 1
         little_number = Abs(offt_radius)
         max_distance = offt_dist
         If little_number < smallest_number(i) And little_number <> 0 Then
            smallest_number(i) = little_number
         End If
         If max_distance > distance(i) Then
            distance(i) = max_distance
            count_long_dist = count_long_dist + 1
         End If
      Loop
   Close #7
Next i

For i = 1 To new_number_of_roads
   If smallest_number(i) >= specified_radius Then
      DATA7 = "k:\oberholf\hsu\mf\output\curves_output\rd" + LTrim$(RTrim$(Str$(i)))
      DATA9 = "k:\oberholf\hsu\mf\output\rigidadb\offtfile\rd" + LTrim$(RTrim$(Str$(i)))
      Open DATA7 For Input As #7
      Open DATA9 For Output As #9
         Do While Not EOF(7)
            Input #7, radius, firstsection, secondsection
            Write #9, Format(radius, "0.00"), _
                  Format(CInt(firstsection), "0.00"), _
                  Format(CInt(secondsection), "0.00")
         Loop
      Close #7
      Close #9
   ElseIf smallest_number(i) < specified_radius Then
      DATA7 = "k:\oberholf\hsu\mf\output\curves_output\rd" + LTrim$(RTrim$(Str$(i)))
      DATA11 = "k:\oberholf\hsu\mf\output\rigidadb\nogo\rd" + LTrim$(RTrim$(Str$(i)))
      Open DATA7 For Input As #7
```

```
    Open DATA11 For Output As #11
      Do While Not EOF(7)
        Input #7, radius, firstsection, secondsection
        Write #11, Format(radius, "0.00"), _
              Format(CInt(firstsection), "0.00"), _
              Format(CInt(secondsection), "0.00")
      Loop
    Close #7
    Close #11
  End If
Next i

For i = 1 To new_number_of_roads
  If smallest_number(i) >= specified_radius Then
    DATA8 = "k:\oberholf\hsu\mf\output\curves_output\fulldata" + LTrim$(RTrim$(Str$(i)))
    DATA10 = "k:\oberholf\hsu\mf\output\rigidadb\offtfile\fulldata" + LTrim$(RTrim$(Str$(i)))
    Open DATA8 For Input As #8
    Open DATA10 For Output As #10
      Do While Not EOF(8)
        Input #8, col1, col2, col3, col4, col5, col6, col7, col8, col9, col10, col11, col12, col13
        Write #10, col1, col2, col3, col4, col5, col6, col7, col8, col9, col10, col11, col12, col13
      Loop
    Close #8
    Close #10
  ElseIf smallest_number(i) < specified_radius Then
    DATA8 = "k:\oberholf\hsu\mf\output\curves_output\fulldata" + LTrim$(RTrim$(Str$(i)))
    DATA12 = "k:\oberholf\hsu\mf\output\rigidadb\nogo\fulldata" + LTrim$(RTrim$(Str$(i)))
    Open DATA8 For Input As #8
    Open DATA12 For Output As #12
      Do While Not EOF(8)
        Input #8, col1, col2, col3, col4, col5, col6, col7, col8, col9, col10, col11, col12, col13
        Write #12, col1, col2, col3, col4, col5, col6, col7, col8, col9, col10, col11, col12, col13
      Loop
    Close #8
    Close #12
  End If
Next i

'NOW RUN OFFTRACK

End Sub
```

APPENDIX C

Users guide for OFFTRACK macro

1. Create a directory for the program, e.g. C:\OT\
2. Copy all the files from the disk to this directory.
3. Change the directory in the OFFTRACK.INI file to the one created above.
4. The road segment files should be named "RD*", e.g. RD1, RD3489, etc. These files does not necessarily have to follow in sequence.
5. One dummy road segment file should have the following name: "ROAD.SEG"
6. Do not put more than one vehicle file in the directory that was created, e.g. "EXAMPLE.VEH". If there are more, the macro would select the first file in the list.
7. Do not put more than one road design file in the directory that was created, e.g. "EXAMPLE.DES". If there are more, the macro would select the first file in the list.

To execute the program, type RUN.

Each RD* file will be written to O-RD* file with the same extension. The O-RD* files are zipped to the file OUTPUT.ZIP

This program does not need stand alone DOS - it is possible to run this macro on a WindowsNT platform.

Possible errors:

1. Make sure there are less than 100 segments in the road segment files (*.SEG) – if there are more than 100, the program would freeze up.
2. When one of the radii is less than the length of the truck, the program could also freeze up.

APPENDIX D

```
Option Base 1
Private Sub Form_Load()

'RUN THIS PROGRAM AFTER OFFTRACK. This program reads the output generated by
'OFFTRACK, and keep track of the maximum amount of offtrack associated with
'each road.

'UNZIP the "OUTPUT*.ZIP" files to following directory:
' "k:\Oberholf\HSU\mf\output\rigidadb\O-RD\"

'Make a list of all the O-RD files that I have created. This can be done in DOS:
'>dir /s/o-s>ordfiles.txt e.g.: k:\Oberholf\HSU\mf\output\rigidadb\O-RD\dir /s/o-s>ordfiles.txt
'Open the ORDFILES.TXT file in Excel, delete non-related files, split
'the file name with the "text to column" function, and sort.

Dim DATA1 As String
Dim DATA2 As String
Dim DATA3 As String
Dim DATA4 As String
Dim DATA5 As String
Dim DATA6 As String
Dim DATA7 As String
Dim DATA8 As String
Dim DATA9 As String
Dim DATA10 As String
Dim DATA11 As String

Dim OFFT(6000, 7) As Double
Dim Offtrack_value(8000) As Double
Dim DATA(10000, 14) As Double
Dim CONSTRAINTS(100000, 10) As Double
Dim FULLDATA(200000, 16) As Double
Dim count(10000) As Integer
Dim distance(8000) As Double
Dim from_to_sorted(30000, 6) As Double
Dim furthest_distance(8000) As Double
Dim from_to_nodes(10000, 5) As Double

GoTo SKIP_EDITING
DATA5 = "k:\Oberholf\HSU\mf\ordfiles.txt"
DATA6 = "k:\Oberholf\HSU\mf\ordfiles_sorted.txt"
Open DATA5 For Input As #5
   Do While Not EOF(5)
      Input #5, FileNumber
      cc = cc + 1
      If FileNumber = cc Then
         DATA(cc, 1) = FileNumber
         DATA(cc, 2) = 1
DATA(cc, 3) = cc 'TAKE OUT LATER
      ElseIf FileNumber <> cc Then
         DATA(cc, 1) = cc
         DATA(cc, 2) = 0
DATA(cc, 3) = cc 'TAKE OUT LATER
         'Create two files (O-RD and OFFTRACK) for each segment that does not exist
         'I do this so that I can use the "For - next" loop with consecutive files.
         DATA1 = "k:\Oberholf\HSU\mf\output\rigidadb\O-RD\O-RD" + LTrim$(RTrim$(Str$(cc)))
         DATA2 = "k:\Oberholf\HSU\mf\output\rigidadb\OFFT\offtrack" + LTrim$(RTrim$(Str$(cc)))
         Open DATA1 For Output As #1
         Open DATA2 For Output As #2
            Write #1, 9999
            Write #2, 9999
         Close DATA1
         Close DATA2
      End If
   Loop
Close #5
```

```
'Write a new file that contain all the file names
Open DATA6 For Output As #6
   For i = 1 To cc
      Write #6, DATA(cc, 1), DATA(cc, 2), DATA(cc, 3)
   Next i
Close #6

new_number_of_roads = cc
For i = 1 To new_number_of_roads
  Offtrack_value(i) = 0
  distance(i) = 0
  furthest_distance(i) = 0
  count(i) = 1
Next i

For i = 1 To new_number_of_roads
   If DATA(i, 2) = 1 Then ' "IF DATA(i, 2) = 1" is used to see which files contain info
      DATA1 = "k:\Oberholf\HSU\mf\output\rigidadb\O-RD\O-RD" + LTrim$(RTrim$(Str$(i)))
      DATA2 = "k:\Oberholf\HSU\mf\output\rigidadb\OFFT\offtrack" + LTrim$(RTrim$(Str$(i)))
      Counter = -1
      Open DATA1 For Input As #1
      Open DATA2 For Output As #2
         Do While Not EOF(1)
            For m = 1 To 8
               Input #1, dummy
            Next m
            For k = 1 To 333
               For m = 1 To 15
                  Input #1, dist_, delta, widthleft, widthright, totalwidth, widening
                  Counter = Counter + 1
                  furthest_distance(i) = dist_
                  Write #2, Counter, dist_, delta, widthleft, widthright, totalwidth, widening
                  current_value = widening
                  If current_value > Offtrack_value(i) Then
                     Offtrack_value(i) = current_value
                     distance(i) = dist_
                  End If
                  OFFT(Counter, 1) = Counter
                  OFFT(Counter, 2) = dist_
                  OFFT(Counter, 3) = delta
                  OFFT(Counter, 4) = widthleft
                  OFFT(Counter, 5) = widthright
                  OFFT(Counter, 6) = totalwidth
                  OFFT(Counter, 7) = widening
               Next m
               For n = 1 To 11
                  Input #1, dummy
               Next n
            Next k
            For m = 1 To 15
               Input #1, dist_, delta, widthleft, widthright, totalwidth, widening
               Counter = Counter + 1
               furthest_distance(i) = dist_
               Write #2, Counter, dist_, delta, widthleft, widthright, totalwidth, widening
               OFFT(Counter, 1) = Counter
               OFFT(Counter, 2) = dist_
               OFFT(Counter, 3) = delta
               OFFT(Counter, 4) = widthleft
               OFFT(Counter, 5) = widthright
               OFFT(Counter, 6) = totalwidth
               OFFT(Counter, 7) = widening
            Next m
            For p = 1 To 3
               Input #1, dummy
            Next p
         Loop
      Close #1
      Close #2
   End If
Next i
```

```
'Make sure all the new files written to k:\Oberholf\HSU\mf\output\rigidadb\OFFT\offtrack
' did run OK, and did not encounter any problems.
DATA9 = "k:\Oberholf\HSU\mf\output\rigidadb\problemfiles.txt"
Open DATA9 For Output As #9
For i = 1 To new_number_of_roads
   If DATA(i, 2) = 1 Then
      If furthest_distance(i) < 4994 Then
         Write #9, i
      End If
   End If
Next i
Close #9

'Put all the constraints in an array/file that can be accessed later.
'new_number_of_roads = 6501

For i = 1 To new_number_of_roads
   If DATA(i, 2) = 1 Then
      DATA3 = "k:\Oberholf\HSU\mf\output\curves_output\fulldata" + LTrim$(RTrim$(Str$(i)))
      Open DATA3 For Input As #3
      Do While Not EOF(3)
         Input #3, file_id, X_coord, Y_coord, distance_, angle_, cm_6, junction_, radius_, cm_9, curve_length, cm_11, cm_12,
road_id
         count_fulldata_files = count_fulldata_files + 1
         FULLDATA(count_fulldata_files, 1) = file_id
         FULLDATA(count_fulldata_files, 2) = X_coord
         FULLDATA(count_fulldata_files, 3) = Y_coord
         FULLDATA(count_fulldata_files, 4) = distance_
         FULLDATA(count_fulldata_files, 5) = angle_
         FULLDATA(count_fulldata_files, 6) = cm_6
         FULLDATA(count_fulldata_files, 7) = junction_
         FULLDATA(count_fulldata_files, 8) = radius_
         FULLDATA(count_fulldata_files, 9) = cm_9
         FULLDATA(count_fulldata_files, 10) = curve_length
         FULLDATA(count_fulldata_files, 11) = cm_11
         FULLDATA(count_fulldata_files, 12) = cm_12
         FULLDATA(count_fulldata_files, 13) = road_id
      Loop
      Close #3
   End If
Next i

For i = 1 To new_number_of_roads
   For j = 1 To count_fulldata_files
      If FULLDATA(j, 1) = i And DATA(i, 2) = 1 Then
         'I should have written a file number at the end of the single road _
          files (FULLDATA files), even if the number is in column 1. To rectify _
          this I will add another "IF statement"
         If FULLDATA(i, 13) <> 0 Then
            If FULLDATA(i + 1, 13) - FULLDATA(i, 13) <> 0 Then
               count(i) = count(i) + 1
               CONSTRAINTS(i, 1) = i
               CONSTRAINTS(i, count(i)) = FULLDATA(j, 13)
               CONSTRAINTS(i, 5) = Offtrack_value(i)
               CONSTRAINTS(i, 6) = count(i)
            End If
         ElseIf FULLDATA(i, 13) = 0 Then
            CONSTRAINTS(i, 1) = i
            CONSTRAINTS(i, 2) = FULLDATA(j, 1)
            CONSTRAINTS(i, 5) = Offtrack_value(i)
            CONSTRAINTS(i, 6) = 1
            GoTo nexti
         End If
      ElseIf FULLDATA(j, 1) = i And DATA(i, 2) = 0 Then
            CONSTRAINTS(i, 1) = i
            CONSTRAINTS(i, 2) = FULLDATA(j, 1)
            CONSTRAINTS(i, 5) = 9999
            CONSTRAINTS(i, 6) = 1
      End If
```

```
   Next j
nexti:
Next i

DATA4 = "k:\Oberholf\HSU\mf\constraints.txt"
Open DATA4 For Output As #4
   For i = 1 To new_number_of_roads
      Write #4, CONSTRAINTS(i, 1), CONSTRAINTS(i, 2), CONSTRAINTS(i, 3), _
            CONSTRAINTS(i, 4), CONSTRAINTS(i, 5), CONSTRAINTS(i, 6)
   Next i
Close #4

'Split all the roads that contain three sections.
For i = 1 To new_number_of_roads
   If CONSTRAINTS(i, 6) = 3 Then 'The number "3" indicate that there are 3 roads
      For j = 1 To new_number_of_roads
         If FULLDATA(j, 1) = i Then
            cum_feet = FULLDATA(j, 11) * 3.28
            offtrack_location = distance(i) - cum_feet
            If offtrack_location <= 0 Then


               Stop

               CONSTRAINTS(i, 8) = CONSTRAINTS(i, 2)
               CONSTRAINTS(i, 9) = CONSTRAINTS(i, 3)
               CONSTRAINTS(i, 10) = CONSTRAINTS(i, 4)

               CONSTRAINTS(i, 2) = FULLDATA(j - 1, 13)
               CONSTRAINTS(i, 3) = FULLDATA(j + 1, 13)
               CONSTRAINTS(i, 4) = 0
               CONSTRAINTS(i, 6) = 2
               CONSTRAINTS(i, 7) = 9
               GoTo next_constraint
            End If
         End If
      Next j
   End If
next_constraint:
Next i

Dim road_attributes(2000, 2) As Double

DATA8 = "k:\oberholf\hsu\mf\attributes_of_roads.txt"
Open DATA8 For Input As #8
   Do While Not EOF(8)
      Input #8, fnode, tnode, lpoly, rpoly, length, road#, road_id, route_no, roadtype, km, roadwidth
      count_roads = count_roads + 1
      road_attributes(count_roads, 1) = road#
      road_attributes(count_roads, 2) = roadwidth
   Loop
Close #8


' Put in the Constraints array a section for the width of the road. I will get this _
from the FULLDATA array.
For i = 1 To new_number_of_roads
   If CONSTRAINTS(i, 1) = 1 Then
      If CONSTRAINTS(i, 6) = 1 Then
         For j = 1 To count_roads
            If CONSTRAINTS(i, 2) = road_attributes(j, 1) Then
               CONSTRAINTS(i, 4) = road_attributes(j, 2) * 3.28 'CONSTRAINTS(i, 7)
            End If
         Next j
      ElseIf CONSTRAINTS(i, 6) = 2 Then
         For j = 1 To count_roads
            If CONSTRAINTS(i, 2) = road_attributes(j, 1) Then
               CONSTRAINTS(i, 4) = road_attributes(j, 2) * 3.28
            ElseIf CONSTRAINTS(i, 3) = road_attributes(j, 1) Then
               If road_attributes(j, 2) > CONSTRAINTS(i, 4) Then
```

```
                CONSTRAINTS(i, 4) = road_attributes(j, 2) * 3.28
              End If
            End If
          Next j
        End If
    End If
Next i

'Now I am done with editing the files. I have what information I need
'to determine which

SKIP_EDITING:
'Put the "FROM_TO_SORTED.TXT" file in an array. This file was created in the
'VB program: "n:\oberholf\HSU\MF\VB_Programs\Sorting_from_to\Sorting_from_to.vbp"
'The "90...." series of coded_id's indicate the "RIGID & DRAWBAR" configuration

'Before the "FROM_TO_SORTED.TXT" file is opened, do the following:
'1. Make sure that the roads leading into the plantation have the FROM_TO nodes represented
'2. Sort it according to the "FROM_NODE" (column 3 )

'This is also where I can write the "FROM_TO_FILE" for the SHORTHAUL configuration
'I have to do the following to achieve this:
'   a. change "from_to_sorted.txt" to "from_to_sorted_SH.txt" in "LINE 1"
'   b. Take the "from_to_sorted_B.txt" file created in "LINE 2",
'      and save it as "from_to_sorted_SH.txt"
'   c. Take this file straight over the next program: NETWORK.VBP

'To write a "FROM_TO_FILE" for the longhaul configuration, I need to do the following
'   a. Copy all the "attributes_of_roads" information for the desired roads (i.e., route 1,5,6,7)
'      Save this file as "attributes_of_mainroads
'   b. Use the "SORTING_FROM_TO.VBP" program to get these files in the desired order.
'      Change the following in the program:
'      b1. Select the correct PREFIX under the EDITNODE subroutine
'      b2. Change the DATA2 filename from "from_to_sorted.txt" to "from_to_sorted_LH.txt"
'      b3. Select the correct filename for DATA1: two options are given
'   c. Follow the same procedures as above to get the "from_to_sorted_LH.txt" file again.

DATA7 = "k:\oberholf\hsu\mf\from_to_sorted_SH.txt"        'LINE 1
Open DATA7 For Input As #7
   Do While Not EOF(7)
   counting_nodes = counting_nodes + 1
   Input #7, coded_id, road_id, from_node, to_node, meters_
      from_to_sorted(counting_nodes, 1) = counting_nodes
      from_to_sorted(counting_nodes, 2) = coded_id
      from_to_sorted(counting_nodes, 3) = road_id
      from_to_sorted(counting_nodes, 4) = from_node
      from_to_sorted(counting_nodes, 5) = to_node
      from_to_sorted(counting_nodes, 6) = meters_
   Loop
Close #7

For i = 1 To counting_nodes
   If from_to_sorted(i, 1) = i Then
      For j = 1 To counting_nodes
         If from_to_sorted(j, 4) = from_to_sorted(i, 5) Then
            count_new_nodes = count_new_nodes + 1
            from_to_nodes(count_new_nodes, 1) = count_new_nodes
            from_to_nodes(count_new_nodes, 2) = from_to_sorted(i, 2)
            from_to_nodes(count_new_nodes, 3) = from_to_sorted(j, 2)
            from_to_nodes(count_new_nodes, 4) = (0.5 * from_to_sorted(i, 6) + 0.5 * from_to_sorted(j, 6))
            from_to_nodes(count_new_nodes, 5) = from_to_sorted(i, 3)
            from_to_nodes(count_new_nodes, 6) = from_to_sorted(j, 3)
         End If
      Next j
   End If
Next i



'This routine will take out all the duplicated sections
```

```
DATA10 = "k:\oberholf\hsu\mf\from_to_sorted_B.txt"        'LINE 2
Open DATA10 For Output As #10
   For i = 1 To count_new_nodes
      If from_to_nodes(i, 2) <> from_to_nodes(i, 3) Then
         Write #10, from_to_nodes(i, 1), from_to_nodes(i, 2), from_to_nodes(i, 3), _
               from_to_nodes(i, 4), from_to_nodes(i, 5), from_to_nodes(i, 6)
      End If
   Next i
Close #10

Erase from_to_nodes
'Put everything back into the array again
counting_nodes = 0
Open DATA10 For Input As #10
   Do While Not EOF(10)
      Input #10, countings, fromnodes, tonodes, meterdistance, from_id, to_id
         counting_nodes = counting_nodes + 1
         from_to_nodes(counting_nodes, 1) = countings
         from_to_nodes(counting_nodes, 2) = fromnodes
         from_to_nodes(counting_nodes, 3) = tonodes
         from_to_nodes(counting_nodes, 4) = meterdistance
         from_to_nodes(counting_nodes, 5) = from_id
         from_to_nodes(counting_nodes, 6) = to_id
   Loop
Close #10

'Flag all the road combinations here. I can specify how wide the critical offtrack should be.

critical_offtrack = 3.28
For i = 1 To counting_nodes
   For j = 1 To new_number_of_roads
      If CONSTRAINTS(j, 6) = 1 Then
         If from_to_nodes(i, 5) = CONSTRAINTS(j, 2) Then
            If CONSTRAINTS(j, 5) - CONSTRAINTS(j, 4) > critical_offtrack Then
               from_to_nodes(i, 7) = 0 'can not be used
            ElseIf CONSTRAINTS(j, 5) - CONSTRAINTS(j, 4) <= critical_offtrack Then
               from_to_nodes(i, 7) = 1 'can be used
            End If
         ElseIf from_to_nodes(i, 6) = CONSTRAINTS(j, 2) Then
            If CONSTRAINTS(j, 5) - CONSTRAINTS(j, 4) > critical_offtrack Then
               from_to_nodes(i, 7) = 0 'can not be used
            ElseIf CONSTRAINTS(j, 5) - CONSTRAINTS(j, 4) <= critical_offtrack Then
               from_to_nodes(i, 7) = 1 'can be used
            End If
         End If
      ElseIf CONSTRAINTS(j, 6) = 2 Then
         If from_to_nodes(i, 5) = CONSTRAINTS(j, 2) And from_to_nodes(i, 6) = CONSTRAINTS(j, 3) Then
            If CONSTRAINTS(j, 5) - CONSTRAINTS(j, 4) > critical_offtrack Then
               from_to_nodes(i, 7) = 0 'can not be used
            ElseIf CONSTRAINTS(j, 5) - CONSTRAINTS(j, 4) <= critical_offtrack Then
               from_to_nodes(i, 7) = 1 'can be used
            End If
         ElseIf from_to_nodes(i, 5) = CONSTRAINTS(j, 3) And from_to_nodes(i, 6) = CONSTRAINTS(j, 2) Then
            If CONSTRAINTS(j, 5) - CONSTRAINTS(j, 4) > critical_offtrack Then
               from_to_nodes(i, 7) = 0 'can not be used
            ElseIf CONSTRAINTS(j, 5) - CONSTRAINTS(j, 4) <= critical_offtrack Then
               from_to_nodes(i, 7) = 1 'can be used
            End If
         End If
      End If
   Next j
Next i

'Now write a LINKFILE that is acceptable to NETWORK 2000

DATA11 = "k:\oberholf\hsu\mf\from_to_sorted_RAD.txt"
Open DATA11 For Output As #11
For i = 1 To counting_nodes
   Write #11, from_to_nodes(i, 1), from_to_nodes(i, 2), from_to_nodes(i, 3), _
         from_to_nodes(i, 4), from_to_nodes(i, 5), from_to_nodes(i, 6), _
```

```
            from_to_nodes(i, 7)
Next i
Close #11

'Now go to the VB program "NETWORK.VBP" to write the network files


End Sub
```