

DEC 23 1997

## United States Government Memorandum

FEDERAL COMMUNICATIONS COMMISSION  
OFFICE OF THE SECRETARY

**To:** Magalie Roman Salas  
**From:** William W. Sharkey  
D. Mark Kennet  
**Subject:** Additional Information Pertaining to the December 11, 1997 Release of the Hybrid Cost Proxy Model  
**Date:** December 23, 1997

**I. How HCPM addresses the ten criteria discussed in paragraph 250, FCC 97-157**

*1. The technology assumed in the model must be the least-cost, most-efficient, and reasonable technology for providing the supported services that is currently being deployed....*

HCPM chooses from among off-the-shelf technologies available today and minimizes the total investment in outside plant assuming that the switch locations are fixed at today's locations. The set of feasible technologies do not incorporate loading coils and do not impede the provision of advanced services; our engineering consultant has certified that the technologies modelled will provide service at the 4 mhz level. If appropriate input data are used, the model will report line counts equal to actual ILEC wire center line counts. The model will report average loop lengths consistent with the distance approximation metric assumed, and offers a user-adjustable factor that can be set to appropriately calibrate looplengths.

*2. Any network function or element, such as loop, switching, transport, or signaling, necessary to produce services must have an associated cost.*

HCPM, as a hybrid module, reports costs for all network components that it models as part of outside plant.

*3. Only long-run forward-looking economic cost may be included....*

HCPM models only those components of the network that are associated with providing residential and business service using the least-cost equipment available today. As such, it is a long-run model in that it assumes that the network being modelled would be built today to meet the user-determined demand. All costs used have been based on vendor prices communicated to us in public channels or from data placed in the public record.

*4. The rate of return must be either the authorized federal rate of return on interstate services....*

The HCPM outside plant module will interface with a capital cost and expense module using any user-defined rate of return.

*5. Economic lives and future net salvage percentages used in calculating depreciation expense must be within the FCC-authorized range....*

The user may determine the appropriate economic life and salvage percentage in the chosen capital cost and expense module.

*6. The cost study or model must estimate the cost of providing service for all businesses and households within a geographic region....*

HCPM models service to all businesses and households that are passed as input to the modules.

*7. A reasonable allocation of joint and common costs must be assigned to the cost of supported services....*

HCPM permits any allocation of joint and common costs desired by the user. In particular, the relevant joint and common cost that may be at issue is the cost of feeder plant in the event that support is to be provided at the Census

block or density zone level. HCPM permits either a line-weighted proportional allocation of feeder plant for the density zone approach or, more consistently with economic theories of subsidies, a Shapley value approach for allocation at the Census block level.

*8. The cost study or model and all underlying data, formulae, computations, and software associated with the model must be available to all interested parties for review and comment....*

All data used in HCPM are either in the public record or available from commercial vendors "off the shelf." All formulas, computations, and software have been placed on the public record.

*9. The cost study or model must include the capability to examine and modify the critical assumptions and engineering principles....*

All assumptions are accessible as user inputs. In particular, the cost of capital and depreciation rates are open to the user in the capital cost and expense modules chosen. Fill factors, input costs, overhead adjustments, retail costs, structure sharing percentages, and terrain factors are all easily editable data. Engineering crossover points are also user-editable, with the understanding that the model does NOT automatically switch, say, from copper to fiber at the crossover point but may determine that the switch occur at a lower value based on economic criteria (it will never switch at a higher value).

*10. The cost study or model must deaverage support calculations to the wire center serving area level at least....*

HCPM permits support calculations be deaveraged to the wire center, and, under appropriate assumptions for allocating feeder cost, to the grid level or density zone level. With effort, the Shapley approach permits deaveraging support calculations to the Census block level or even the level of an individual geocoded customer (if input data are individual geocoded customers).

## **II. Use of geocoded customer locations with HCPM**

The Hybrid Cost Proxy Model has been designed with the anticipation that geocoded customer locations will eventually be available for use with it. Currently the model accepts inputs in the form of data files containing a line of ascii text representing each census block whose interior point is contained in the wire center boundaries. For each census block, the model reads the geocoded location of the interior point, the households and business lines contained in the block and certain other geological information relevant to the block. As written, the model will readily accept as inputs geocoded customer location data for individual households and business without any modification, as long as the data are formatted in conformance with the model input requirements. In order for the data to be preprocessed to match HCPM's input requirements, the following steps would need to be taken.

1. Using MapInfo® (or other GIS software product), associate each customer location with appropriate geological/terrain data. The proponents of the BCPM model have made such data available at the Census Block Group (CBG) level. Thus, the process would likely take the form of determining in which CBG each customer resides using MapInfo and CBG boundary data (available at cost from the Census Bureau), and then associating the BCPM terrain/geologic data with that customer.
2. Write the customer location information to an ASCII file, with each line corresponding to one customer location. The format of these data would be that required by HCPM.

The appropriate HCPM module (CENBLOCK) would be able to process this file exactly as it currently processes files with the Census block constituting each record. Processing time is likely to be somewhat longer than with Census blocks, since the machine would have to read through more records, but the operation would otherwise be unchanged. The user would have to set the MicroGridSize parameter to a number such that no more than 2500 microgrids -- or 50 rows of 50 columns -- would be created within a grid block. For example, if 18 kf were the grid size, then the MicroGridSize parameter would need to be set to at least 0.36 ( $0.36 \times 50 = 18$ ). Thus, the resolution of the CENBLOCK output would be 360 feet, less than the average city block (about 500 feet).

### III. How the HCPM modelling team has adjusted line counts

When a reliable source of household and business line counts becomes available at the wire center level, it will be a straightforward matter to true-up the publicly available household and business line count information to reflect the best available data for each wire center. Until such a data source is available, the model must make use of more aggregated public information to determine line counts. For our default inputs for each wire center, we have adjusted the data on households from the Census Bureau to match the household totals put in the public record by the BCPM proponents. Additionally, we have allocated the business line counts available from the public record, again from the BCPM proponents, to the Census blocks we use as inputs. The process for these two adjustments is as follows.

The data from the Census Bureau on households at the Census block (CB) level is accurate as of 1990. BCPM has reported 1995 updated estimates for household counts at the Census Block Group (CBG) level. We have adjusted our CB-level household counts as follows:

$$\text{HCPM CB households} = \left( \frac{\text{Census CB households}}{\text{Census CBG households}} \right) \times (\text{BCPM CBG households})$$

where CB refers to an individual Census block; and CBG refers to the Census block group of which the Census block is a member.

Intuitively, this formula is saying that the *proportion* of CBG households within a given CB has remained constant, but that *growth, emigration, etc.*, have caused the CBG population to change from the Census year (1990) to the time of BCPM's update (1995). Thus, our new household counts reflect the changes that BCPM records at the CBG level.

A similar heuristic process is used to allocate the BCPM business line counts from the CBG level to the CB level. The formula is as follows:

$$\text{HCPM CB Business lines} = \left( \frac{\text{Census CB households}}{\text{Census CBG households}} \right) \times (\text{BCPM CBG Business lines})$$

Intuitively, this formula says that BCPM's CBG business lines occur in the component CBs in direct proportion to the household population share.

It should be noted that these values are default values only; the HCPM modelling team does not take a position as to whether the BCPM line counts should be deemed accurate. Users remain free to adjust these values if their research suggests that other values are appropriate. We do feel that there is an advantage to defaulting to one proponent's line counts, though, in order to facilitate direct comparisons between model platforms/algorithms.

The HCPM estimates the number of residential lines by applying a user defined multiplier to the number of households (as computed above) in order to determine the total number of residential lines. The appropriate value of the multiplier can be determined from publicly available data for any state by taking the ratio of residential access lines from 1995 ARMIS data (reported in Table 2.5) to the number of households determined from Census data. In computing universal service support levels for single line business, data from the same ARMIS table can be used to estimate the ratio of single line business lines to total business lines.

### IV. Total Monthly Cost Calculation in the HCPM

A direct calculation of total monthly cost would involve the following components: (i) the capital costs associated with the loop plant investment and monthly expenses associated with this investment; (ii) the capital cost of switching and its associated expense; (iii) an allocation of the capital cost of signaling and transmission and their associated expenses to the wire center. Total monthly cost is the sum of these components. For the purpose of estimating universal service support levels based on a hybrid cost proxy model that uses the HCPM 2.0 customer location and loop design modules, combined with switching, signalling and transport, and expense modules from other

models, we have adopted the following indirect approach. Using BCM2 data, total monthly cost can be associated with corresponding data at the wire center level on loop investment and total lines. Using standard econometric techniques we fit a trans-log functional form relationship to this data in order to derive a "reduced form" expense model estimator. We then applied this function to wire center data on loop investment and line counts as determined by the HCPM in order to estimate the monthly cost and corresponding universal service support levels that would have been generated by a hybrid model having these components. This prediction therefore reflects the same cost of capital, depreciation rate (asset lives), and operating expenses that would be observed in a direct calculation. Since our estimation procedure results in statistically significant estimates of parameters of a cost per line function and represents a good fit to the underlying BCM data (with an  $R^2 = .998$ ), we may reliably use the indirect calculation in order to predict aggregate support levels under the hybrid model approach.

A more detailed description of our estimation procedure is given as follows. In our preliminary capital cost/expense module of the HCPM, the monthly cost per line for a wire center is estimated as a function of loop plant investment per line and lines. These variables are available by wire center and a function predicts the total monthly cost per line, including switching, transmission, and other cost that are relevant to the provision of universal service for the wire center.

For any wire center, our total monthly cost per line is derived by estimating

$$\ln\left(\frac{Cost}{L}\right) = \beta_0 + \beta_1 \ln\left(\frac{Inv}{L}\right) + \beta_2 \ln(L) + \beta_3 \left(\ln\left(\frac{Inv}{L}\right)\right)^2 + \beta_4 (\ln(L))^2 + \beta_5 \ln\left(\frac{Inv}{L}\right) \ln(L)$$

where Cost is total monthly cost and L is lines. The variable

$\frac{Inv}{L}$  is loop investment per line.

We estimate this function using data from the BCM2 model. We took a random sample of states, aggregated the data by wire center, and, then, pooled the data over the states of the sample. The states are Alabama, Wisconsin, MA, RI, and CO. Our estimation is based on 1582 observations (wire centers) across five states.

We first estimated our function under OLS assumptions and, then tested for heteroscedasticity. Using the Breusch Pagan test, we found heteroscedasticity. We implemented GLS estimation and corrected for heteroscedasticity by deflating all variables by the square root of the natural log of lines.

We then estimate total cost per line as

$$\frac{Cost_j}{L_j} = e^{(\beta_0^* + \beta_1^* \ln(\frac{Inv_j}{L_j}) + \beta_2^* \ln(L_j) + \beta_3^* (\ln(\frac{Inv_j}{L_j}))^2 + \beta_4^* (\ln(L_j))^2 + \beta_5^* \ln(\frac{Inv_j}{L_j}) \ln(L_j))}$$

where investment and line numbers are taken from the HCPM.

Thus, our estimation of total monthly costs captures costs associated with loop plant, transmission, switching, output volumes, and all relevant costs associated with universal service. The cost of capital, asset lives, and forward looking operating expense will reflect BCM2 assumptions. In summary, we consider our total monthly cost calculations to be a reasonable upper bound given available data. Moreover, our total monthly cost calculation is a preliminary method for integrating HCPM with other modules from either the Hatfield model or the BCPM.

## **Release Notes for HCPM 2.0**

### **December 11, 1997**

This release includes the following software and supporting materials:

- \* Cenblock 2.0: the customer location module for HCPM 2.0
- \* Feeddist 2.0: the loop design module for HCPM 2.0
- \* Two sets of parameter inputs under which Feeddist can be run
- \* Model documentation and user guides
- \* Data inputs for 49 states and the District of Columbia. Cenblock has been run using four different parameter sets corresponding to default grid sizes of 12, 18, 24 and 30 kilofeet. These outputs are provided in compressed files designated by a state code and a number code corresponding to the grid size.
- \* Model outputs for each of the above scenarios and for a cost-minimizing run in which costs are minimized with respect to grid size. Output files are designated by a state code, a number code representing the grid size and a letter code designating the parameter input set under which feeddist was run.
- \* Batch files which allow Feeddist to be run for the entire United States under a single command. This procedure takes approximately 24 hours to complete using a Pentium II processor and the Windows NT operating system.
- \* Complete source code for all modules

In response to the Federal Communication Commission's Public Notice, released November 13, 1997, output data for five states -- Florida, Georgia, Maryland, Missouri, and Montana -- and the executable files for HCPM 2.0 was provided to the Commission on December 11, 1997. In further compliance with the Public Notice we address the following issues.

#### 1. Ability of the model to accept geocoded data.

HCPM 2.0 will accept any data that conform to the input specifications in the CENBLOCK manual, including individually geocoded customers. In the event that such data were to be made available for use with CENBLOCK, the user would merely run CENBLOCK with the microgrid size parameter set equal to a number no smaller than  $\frac{1}{50}$  of the chosen grid size.

#### 2. Ability of the model to accept wire center boundary data in standard GIS format.

Wire center boundary data are preprocessed by a standard commercial GIS software package. In our preliminary runs, we have used MapInfo software with ExchangeInfo Plus boundary data; other

packages can be substituted if they are deemed to contain more accurate data. The GIS software and boundary data are used only to assign customer locations to a wire center; their source - as long as it is accurate and the preprocessing results in datasets that conform to the CENBLOCK input specifications - is not relevant to HCPM 2.0

### 3. Documentation of assumptions and model optimization algorithms.

The HCPM, version 2.0, optimizes the design of loop plant in the following respects.

\* The model optimizes the trade-off between distribution plant, feeder plant and loop electronics in several ways. As explained in the model documentation, the model accepts as candidate serving areas a set of grids and sets of from 1 to 4 serving area interface terminals for each grid. For each such configuration, distribution plant is designed by attaching customers to their closest SAI terminal. Then the model determines the cost minimizing number of SAIs in each grid after taking proper account of the cost of terminals and the cost of interconnecting SAIs when two or more are present. Finally, the model can be run using various starting values for serving area grid size - the current outputs include grid sizes of 12, 18, 24 and 30 kilofeet. By taking the minimum cost for each wire center over each of these possibilities, the model compares the increased cost of distribution in larger grids to the reduced cost of electronics and feeder plant that typically occur using larger grids. Each step in this optimization procedure takes full account of appropriate engineering constraints governing maximum copper distance and the copper-T1 and T1-fiber crossover points.

\* The model allows for the use of both 26- and 24-gauge copper in the distribution plant. When the user specified 26-gauge distance threshold is exceeded for any customer in the serving area, a cost multiplier is applied to account for the additional cost of 24-gauge copper. Whenever the maximum copper distance is exceeded for any customer in a serving area, a further penalty multiplier is applied to account for the cost of thicker copper or T1 electronics that could be used to reach distant customers. By setting this penalty sufficiently high, the user can effectively impose the maximum copper distance as a distance ceiling for every customer served.

\* For each grid location, and each feasible technology (analog, T1, or fiber) the model optimizes the number of T1 terminals and/or fiber terminals at each serving area interface. Based on the results of this optimization, the model then selects the technology giving minimum cost for that location. In addition, the model optimizes over technology type during the construction of the feeder and sub-feeder system, since the distance of each SAI from the central office is a function of the feeder system under consideration.

\* The model optimizes both the number and location of sub-feeder routes using an algorithm described in the model documentation. In both the distribution and feeder sections of the model, rectilinear (L1) distances are used in all computations rather than airline (L2) distances.

4. Flexibility of the loop design algorithms to allow alternative definitions of supported services in light of future "changes in technology, network capacity, consumer demand, and service deployment."

As described above, the model seeks to find a cost minimizing network design based on a large number of technological options and user specified parameters that reflect grid size and technology crossover points. Through suitable choices of the available user inputs, the model can reflect a network design that is consistent with virtually any quality of service standard.

5. Ability of the model to incorporate wireless cost thresholds at the level of the wire center or smaller geographic unit.

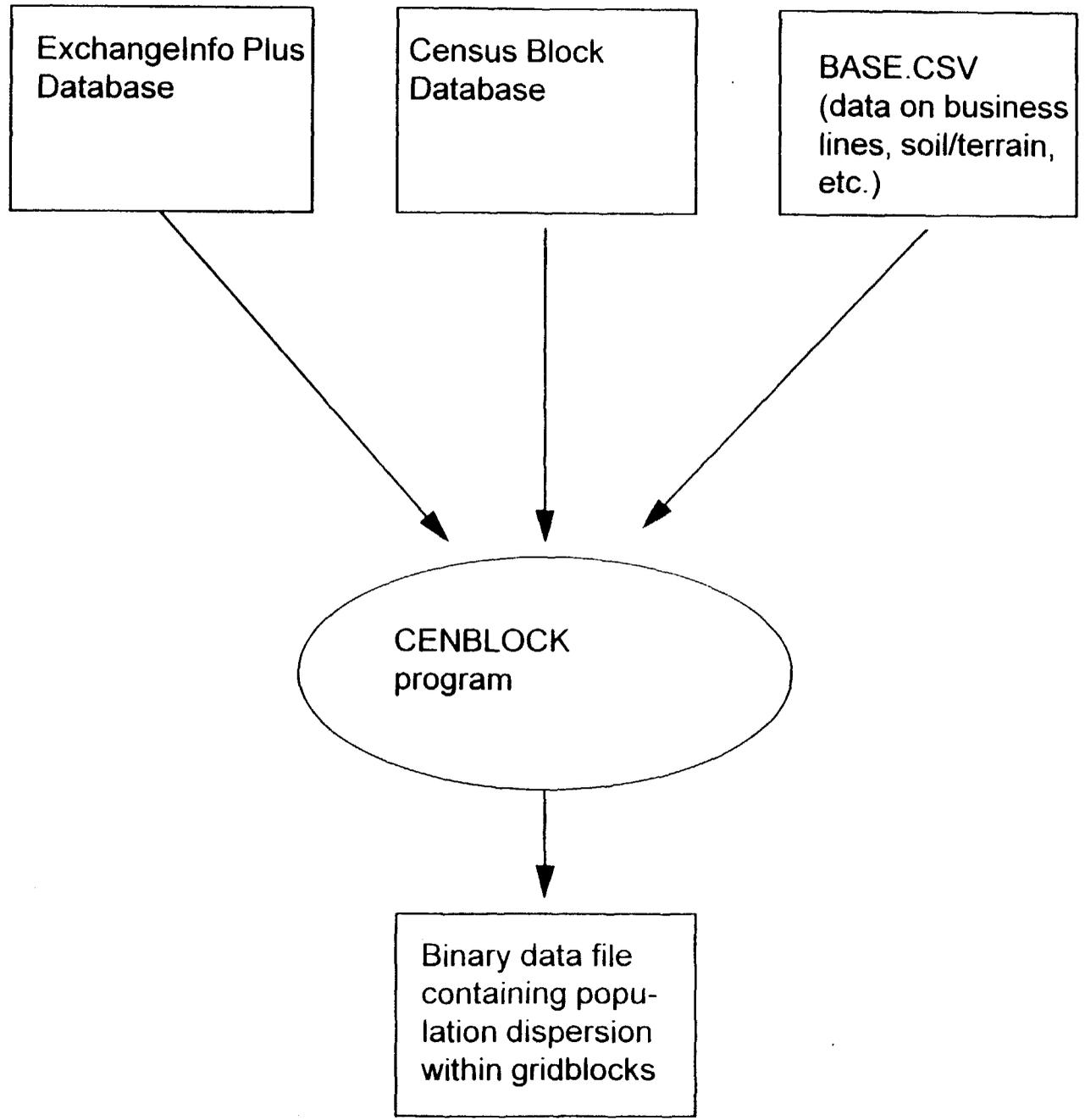
The model currently reports all outputs at the wire center level. Through a simple user computation using these outputs, any desired wireless threshold could be immediately incorporated into the model outputs. If cost estimates based on a smaller geographic unit are desired at a future time, these could be provided through minor programming revisions.

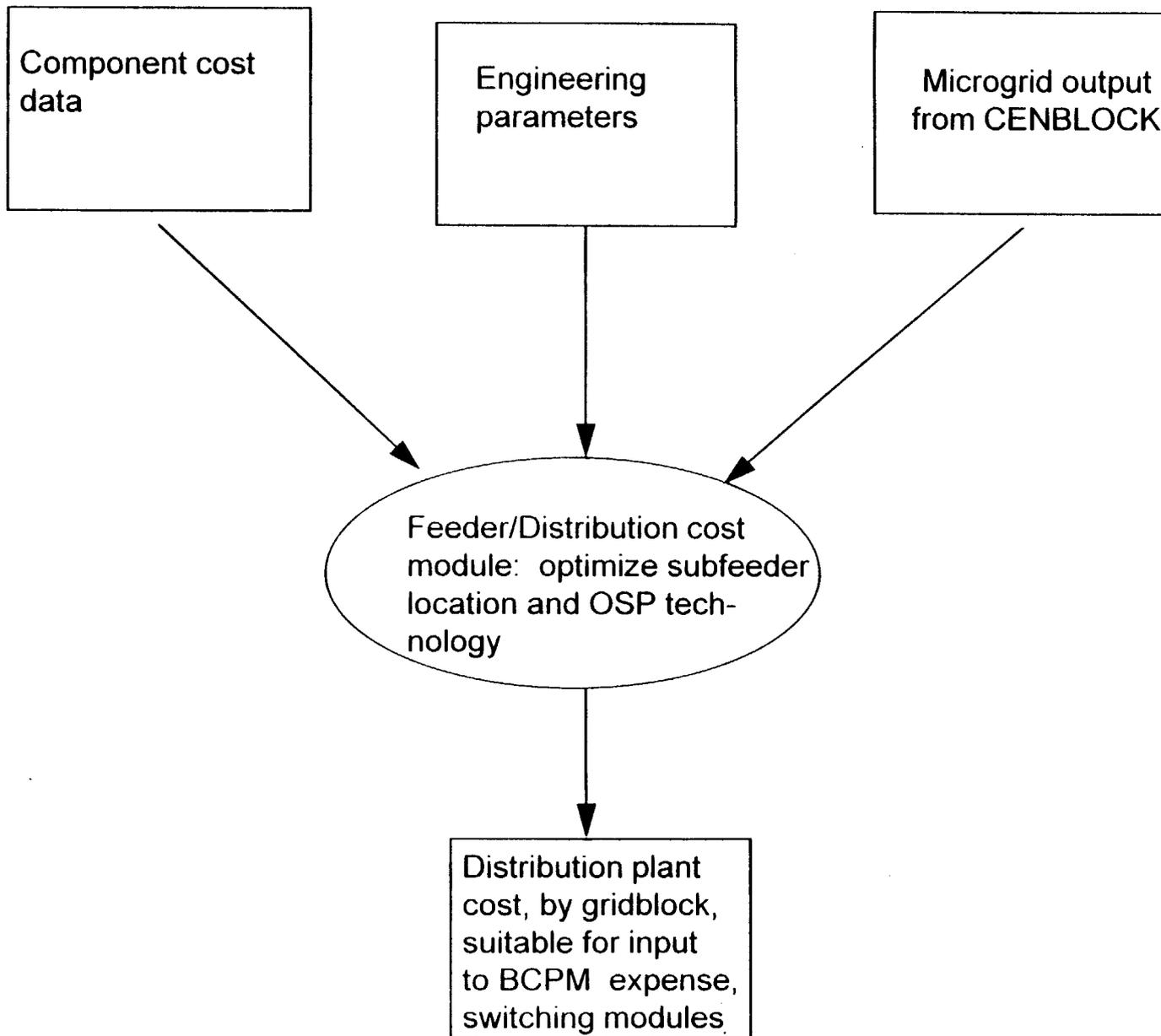
6. Fiber-copper crossover point.

As noted above, the model outputs will provide all requested information for five states based on grid size and parameter inputs that define the appropriate crossover points. For the 12 kilofoot output reports, both the maximum copper distance and the copper-T1 crossover point are equal to 12 kf; the T1-fiber crossover is equal to 18 kf and the penalty for exceeding the maximum copper distance is equal to 1.5 (reflecting a 50% cost increment). For each of the 18, 24 and 30 kilofoot output reports, the maximum copper distance and the copper-T1 crossover point are equal to 18 kf; the T1-fiber crossover is set at 24 kf; the 24 gauge multiplier is equal to 1.1736 and the copper distance penalty is equal to 1.25.

7. Proprietary or confidential information.

The HCPM makes no use of proprietary or confidential information. As described in the model documentation, inputs include data from the Bureau of the Census; geological and business line count data from previous publicly available versions of the BCPM; and wire center boundary data from ExchangeInfo Plus, a product of On Target Mapping, Inc. Users who wish to obtain the raw data inputs must either purchase the wire center boundary data from On Target Mapping, or negotiate with On Target Mapping the terms under which the HCPM input data will be used. No payments or negotiations are required in order to make use of all model data supplied with this release.





## CENBLOCK CUSTOMER LOCATION MODULE EQUATIONS/PSEUDOCODE

### Adjust\_large\_blocks

This subroutine adjusts all input mesh areas (Census blocks) over the user-specified maximum area. If the MA is smaller than the critical value, it leaves it alone. If it is larger, it performs the following operation:

```
Number of grids := round( area of MA/Critical Area + 0.5 ); {approximate number of}  
                                                         { slices of MA}  
    Number of rows := round( sqrt(number of grids) + 0.5 );  
    Number of columns := number of rows;  
  
    Number of grids := (number of rows)*(number of columns);  
  
    side := sqrt( area of MA*5.28*5.28/2590 );           {This is the side of a square }  
                                                         { equal to area of MA }  
  
    lower left x-coordinate := x-coordinate of MA - 0.5*side; {These are the lower}  
    lower left y-coordinate := y-coordinate of MA - 0.5*side; {left coordinates of}  
                                                         {the square}  
  
    area of MA := area of MA/number of grids; { Each artificial MA will have this}  
                                                         { area. }  
  
    if residential population of MA>0.0 then  
    begin  
        residential population := residential population /number of grids;  
        if round(residential population)=0 then residential population := 1.0;  
    end;  
  
    if business population of MA>0.0 then  
    begin  
        business population := business population/number of grids;  
        if round(business population)=0 then business population := 1.0;  
    end;  
  
{ Now calculate internal points for each artificial MA created. Lower_left[1] is the x-}  
{coordinate of the lower left corner of the wire center data; lower_left[2] is the y-}  
{coordinate. Ref_latitude is the reference latitude used to calculate east-west }  
{distances. }  
  
    for i := 0 to number of rows-1 do  
        for j := 0 to number of columns-1 do
```

```

begin
  ylat := lower left y + i*side/number of rows + 0.5*side/number of rows;
  ylat := ylat/(KFPerStatMi*StatMiPerMin*MinPerDegree) + lower_left[2];
  xlong := lower left x + j*side/number of rows + 0.5*side/number of
    columns;
  xlong := xlong/(KFPerStatMi*StatMiPerMin*MinPerDegree*
    cos(ref_latitude*pi/180.0)) + lower_left[1];
end;

```

### Setup\_grid

This procedure sets up the grid structure for the wire center territory.

*{ Define length and width of wire center territory }*

```

North-South length := abs(lower_left[2] - upper_right[2])*
  MinPerDegree*StatMiPerMin*KFPerStatMi;

```

```

East-West length := abs(lower_left[1]-upper_right[1])*
  MinPerDegree*StatMiPerMin*KFPerStatMi*cos(ref_latitude*pi/180.0);

```

*{ Calculate number of "tick marks" on each axis, which corresponds to the number of grids in each direction. }*

```

North-South ticks := round(North-South length/gridsize + 0.5);
East-West ticks := round(East-West length/gridsize + 0.5);

```

*{ Calculate the number of grids, which equals the number of North-South ticks times the number of East-West ticks. }*

```

Number of grids := North-South ticks*East-West ticks;

```

```

for j := 1 to Number of grids do
begin

```

```

  column := j mod (East-West ticks);
  if (column = 0) then column := East-West ticks;

```

```

  row := j div (East-West ticks) + 1; if (j mod (East-West ticks)) = 0 then row :=
    row-1;

```

*{ The following are the lower left (x,y) and upper right (x,y) coordinates of each grid }*

```

  glx := (column-1)*gridsize;
  grx := column*gridsize;

```

```
gly := (row-1)*gridsize;  
gry := row*gridsize;
```

```
end;
```

### **Attach\_mesh\_areas**

This procedure takes each grid block and loops through the input dataset and determines whether each mesh area record belongs in the grid block. If so, it adds it to the grid block, keeping track of the total number of lines in the grid block.

```
for j := 1 to number of grid blocks do  
  for i := 1 to number of input records do  
    begin  
      read_data_record;  
      IF ( x, y ) of input record inside grid block j  
      THEN grid population := grid population + mesh population;  
    end;
```

### **Adjust\_grid\_areas**

This procedure adjusts those grid areas that have a line population greater than the user-specified maximum.

```
For j := 1 to number of grid blocks do  
  begin  
  
    if ( ( grid line population ) > maxgridpop )  
    and ( number of mesh areas in this grid block > 1 )  
    then  
    begin  
      divisor := 2 ;  
      additional grid blocks := additional gridblocks + divisor*divisor - 1;  
      height := abs(y distance of grid block)/divisor;  
      width := abs(x distance of grid block)/divisor;  
  
      for k := 1 to divisor do  
        for l := 1 to divisor do save_grid_information;  
      end  
    else save original grid information;
```

### **penalty\_function**

This function assigns a penalty to any proposed location of a serving area interface (SAI). The program will try to minimize this penalty when it reports SAI locations to the feeder/distribution cost module.

```
tcost := 0.0;
  for i := 1 to number of rows do
    for j := 1 to number of columns do
      if (Lines[i,j])>0
      then
      begin
        if ( distance from microgrid to SAI[1] > gridsize )
        then kmincost := 1.0e16
        else kmincost := ( distance from microgrid to SAI[1] ) * Lines[i,j];

        for k := 2 to n do
          begin
            if ( ( distance from microgrid to SAI[k] ) > gridsize )
            then kcost := 1.0e16
            else kcost := ( distance from microgrid to SAI[k] ) * Lines[i,j];
            if kcost < kmincost then kmincost := kcost;
          end;
          tcost := tcost + kmincost;
        end;
        dist_cost := tcost + distance from SAI[1] to switch;
      end;
```

### **Build\_serving\_areas;**

This procedure writes the grid information to a file. In addition, it reports the optimized location of 1, 2, 3, and 4 SAIs. These locations are calculated by minimizing the penalty cost function described above.

## FEEDDIST FEEDER AND DISTRIBUTION MODULE

### I. STRUCTURE MODULE (STRUCTUR.PAS)

#### Structure\_cost\_fn

This function serves as a lookup for structure costs. It takes density and terrain data as an argument, and looks up percentage of underground, buried, and aerial cable; sharing percentage; and total structure cost based on terrain factors.

```
if (depth_to_bedrock < critical_depth) and (hardness='HARD') then
    { use hard rock values }
begin
    temp1 := pct_ugd*ugd_share*HardRockStruc[i]^FeedUgd +
             pct_bur*bur_share*HardRockStruc[i]^FeedBur +
             pct_aer*aer_share*HardRockStruc[i]^FeedAer

    NumberOfDucts := round( copper_lines/feed_copper_cable_capacity + 0.5 ) +
                     round( fiber_lines/fiber_cable_capacity + 0.5 ) + 1

    temp2 := ManholeCost[i]^HardCost/ManholeSpacing;
             { manhole cost per foot for underground }
end
else
    if (depth_to_bedrock >= critical_depth) and (soil_texture_indicator=1) then
        { use normal values }
    begin
        temp1 := pct_ugd*ugd_share*NormalStruc[i]^FeedUgd +
                 pct_bur*bur_share*NormalStruc[i]^FeedBur +
                 pct_aer*aer_share*NormalStruc[i]^FeedAer

        NumberOfDucts := round( copper_lines/feed_copper_cable_capacity + 0.5 ) +
                         round( fiber_lines/fiber_cable_capacity + 0.5 ) + 1

        temp2 := ManholeCost[i]^NormalCost/ManholeSpacing;
                 { manhole cost per foot for underground }
    end
    else
        { use soft rock values }
    begin
        temp1 := pct_ugd*ugd_share*SoftRockStruc[i]^FeedUgd +
                 pct_bur*bur_share*SoftRockStruc[i]^FeedBur +
                 pct_aer*aer_share*SoftRockStruc[i]^FeedAer
```

```

NumberOfDucts := round( copper_lines/feed_copper_cable_capacity + 0.5 ) +
                round( fiber_lines/fiber_cable_capacity + 0.5 ) + 1

temp2 := ManholeCost[i]^SoftCost/ManholeSpacing;
        { manhole cost per foot for underground}

end;

temp1 := temp1 + pct_ugd*ugd_share*temp2;

if (MinSlope < MinSlopeTrigger) and (MaxSlope > MaxSlopeTrigger) then temp1 :=
    temp1*CombSlopeFactor
else
    if (MinSlope < MinSlopeTrigger) then temp1 := temp1*MinSlopeFactor
    else
        if (MaxSlope > MaxSlopeTrigger) then temp1 := temp1*MaxSlopeFactor;

structure_cost_fn := temp1*1000.0; { result in dollars per kilofoot }

```

## II. DISTRIBUTION PLANT MODULE (DISTRIB.PAS)

This module calculates the investment in distribution plant. Within each microgrid, it calculates the number of lots based on the number of customers and an assumption that no lot has a length more than twice its width. The model provides plant to the microgrid, and joins microgrids on distribution backbone that feed into the serving area interface (SAI). The model tries up to four SAIs, and chooses the number of SAIs that minimizes investment cost. In the case of multiple SAIs, the model uses an algorithm invented by Prim (*Bell Research Journal*, 1957) to approximately minimize the structure needed to connect the secondary SAIs to the primary SAI. This algorithm is contained in the PRIMDIST module. The module also uses a library heapsort routine as implemented in *Numerical Recipes* by Press, Flannery, Teukolsky, and Vetterling (1986) with slight modifications to handle vectors that have been dynamically allocated to extended memory.

### lot\_divide

This procedure minimizes wasted lots within a square microgrid, subject to the constraint that lots have lengths no more than twice their widths. It returns the "optimal" number of lots in the NS and EW direction.

```

sqrt2 := sqrt(two);
waste := number_of_lots;

```

```

minwaste := number_of_lots;
sqnl := sqrt(number_of_lots);
for i := round( sqnl/sqrt2 ) to round(sqnl) + 1 do
    { Check from square root of number of lots/2 to square }
    { root of number of lots. This guarantees that max }
    { length - width ratio is no more than 2. }
begin
    EW_try := i;
    NS_try_d := number_of_lots/EW_try ;
    NS_try := round(NS_try_d);
    waste := NS_try*EW_try - number_of_lots;
    if (waste < 0) then waste := number_of_lots;
    if (waste <= minwaste) then
        begin
            minwaste := waste;
            EW_lots := round(EW_try);
            NS_lots := round(NS_try);
        end
    end; {for EW_try}

```

### Calculate\_Microgrid\_Cost

This procedure calculates the cost of providing service to a microgrid whose lots have been configured by the lot\_divide procedure above. Starting at lower left of microgrid, we walk north up every other lot line, accumulating lines and cable. If we accumulate enough lines for a new cable, we add it, repeating the exercise until we reach either the the northern boundary.

```

i := 1;
while i <= EW_lots do
    begin
        j := 1;
        while j <= NS_lots do
            begin
                { Take in lots on both sides, top and bottom, unless this is a microgrid }
                { border, in which case take in lots only on one side. If it is the }
                { corner, take in only one lot. }
                if ( i=EW_lots ) or ( j=NS_lots ) then factor := 2.0 else factor := 4.0;
                if ( i=EW_lots ) and ( j=NS_lots ) then factor := 1.0;
            end
        end
    end

```

```

lines := lines + factor*lines_per_lot;
cable_cost := dist_cable_cost(lines,density,gauge);
structure_cost := structure_cost_fn(
    lines,0,density,GR.hardness,GR.DepthToBedrock,GR.SoilTexture,
    GR.MinSlope,GR.MaxSlope,GR.WaterTb,0,1,0);
if ( j <= NS_lots-2 ) then
begin
    microgrid_cost := microgrid_cost +
        (2.0/NS_lots)*GR.MicroGridNS*DistRoadFactor* { frontage of 2 lots }
        ( cable_cost + structure_cost );
    line_feet := line_feet + (2.0/NS_lots) *
        GR.MicroGridNS*lines*DistRoadFactor;
    drop_terminal_cost := drop_terminal_cost +
        drop_terminal_cost_fn( factor*lines_per_lot, density );
end
else if ( j = NS_lots-1 ) then
begin
    microgrid_cost := microgrid_cost +
        (1.0/NS_lots)*GR.MicroGridNS*DistRoadFactor* { frontage of 1 lot }
        ( cable_cost + structure_cost );
    line_feet := line_feet + (1.0/NS_lots) *
        GR.MicroGridNS*lines*DistRoadFactor;
    drop_terminal_cost := drop_terminal_cost +
        drop_terminal_cost_fn( factor*lines_per_lot, density );
end
else if ( j = NS_lots ) then
    { at the border, we only have drop terminals; no cabling }
    drop_terminal_cost := drop_terminal_cost +
        drop_terminal_cost_fn( factor*lines_per_lot, density );

    j := j + 2;
end; { for j }

```

```

i := i+2;
end; { while i }

```

*{ Now we need to calculate drops to customer locations. The following }  
 { formula takes a weighted average of the distance from the corner of }  
 { the lot to the center of the lot and half the road frontage of the lot. The }  
 { weight, user\_lambda, is chosen by the user. If the calculated drop }  
 { length exceeds the user-determined maximum, it is set equal to that }  
 { maximum. }*

```

drop_length := user_lambda*0.5*
  sqrt(
    sqr( (1.0/NS_lots)*GR.MicroGridNS*DistRoadFactor ) +
    sqr( (1.0/EW_lots)*GR.MicroGridEW*DistRoadFactor )
  ) +
  (1.0 - user_lambda)*0.5*(1.0/NS_lots)*GR.MicroGridNS*DistRoadFactor;

```

```

if drop_length > max_drop_length then drop_length := max_drop_length;

```

```

drop_cost := total_lots*drop_length*cost_per_drop_kf;
drop_feet := total_lots*drop_length;

```

```

{ Finally, calculate cost of nids for this microgrid }

```

```

MG_nid_cost := nid_cost*total_lots;

```

### Calculate\_grid\_distribution\_cost

This procedure connects all microgrids to the appropriate SAI, given the number of SAIs. The routine that optimizes the SAI arrangement will call this procedure using 1 through 4 SAIs and determine which arrangement minimizes cost. If there are more than one SAI, this procedure determines a near-optimal interconnection arrangement using the Prim algorithm.

*The following section of pseudo-code connects the southwest "quadrant" of microgrids to its respective SAI.*

```

for j := 1 to divider_col do {from western border to column where SAI is located}
  begin
    if (flag^[i,j]=n) and (lines^[i,j]>0) then
      { if microgrid below is populated }
      begin
        lots := round(GR.households[i,j]*takerate) +
          round(GR.buslines[i,j]/lines_per_bus);
        microgrid_lines := lines^[i,j];
        lot_divide;
        calculate_microgrid_cost;

        grid_distribution_cost := grid_distribution_cost +
          microgrid_cost;
        grid_drop_cost := grid_drop_cost + microgrid_drop_cost;
        grid_terminal_cost := grid_terminal_cost +

```

```

        microgrid_terminal_cost;
        grid_nid_cost := grid_nid_cost + microgrid_nid_cost;
        grid_line_feet := grid_line_feet + microgrid_line_feet;
        grid_drop_feet := grid_drop_feet + microgrid_drop_feet;
end;

rows_completed := i+1;

if (flag^[i+1,j]=n) and (lines^[i+1,j]>0) then
    { if microgrid above is populated }
begin

    lots := round(GR.households[i+1,j]*takerate) +
            round(GR.buslines[i+1,j]/lines_per_bus);
    microgrid_lines := lines^[i+1,j];
    lot_divide;
    calculate_microgrid_cost;
    grid_distribution_cost := grid_distribution_cost +
        microgrid_cost*penalty;
    grid_drop_cost := grid_drop_cost + microgrid_drop_cost;

    grid_terminal_cost := grid_terminal_cost +
        microgrid_terminal_cost;
    grid_nid_cost := grid_nid_cost + microgrid_nid_cost;
    grid_line_feet := grid_line_feet + microgrid_line_feet;
    grid_drop_feet := grid_drop_feet + microgrid_drop_feet;
end;

{ Bring forward lines from previous microgrids }

if ( there are any new lines ) then
    bring them to first interconnection point;
else
    No lines here, so cross microgrid ;

    Capture lines from these microgrids;

    Bring forward lines to next microgrids;

end; { for j }

```

*The above is repeated for each quadrant, for each SAI. Now all secondary SAIs must be joined, which is handled by the Prim algorithm, discussed in the PRIMDIST module.*

## Optimize\_SAI\_arrangement

This procedure determines the optimal configuration of primary and secondary SAIs by calculating the distribution cost of 1 through 4 SAIs. In so doing, it recognizes the tradeoff between the extra cost of T-1 terminals at the secondary SAIs and the structure cost associated with serving a possibly quite diffuse customer base.

```
mincost := 1.0e+16;
for number_of_SAIs := 1 to 4 do
begin
  SA.TypeOfSAI[1] := primary;
  if number_of_SAIs > 1 then
    for i := 2 to number_of_SAIs do SA.TypeOfSAI[i] := secondary;

  calculate_grid_distribution_cost;

  if grid_distribution_cost < mincost then
  begin
    mincost := grid_distribution_cost;
    SA.number_of_SAIs := number_of_SAIs;
    SA.X := X;
    SA.Y := Y;
    SA.grid_distribution_cost := grid_distribution_cost - term_cost;
    SA.secondary_term_cost := term_cost;
    SA.snc96 := nc96;
    SA.snc24 := nc24;
    SA.grid_line_feet := grid_line_feet;
    SA.grid_drop_feet := grid_drop_feet;
    SA.density := density;
    SA.drop_cost := drop_cost;
    SA.drop_terminal_cost := drop_terminal_cost;
    SA.nid_cost := nid_cost;
    SA.MaxDistance := MaximumDistance;
  end;
end; { for number_of_SAIs }
```

### III. PRIM DISTRIBUTION MODULE (PRIMDIST.PAS)

This module sets up a matrix of costs of connection between each SAI (assuming T1 connection) and arranges the network so that each SAI (primary or secondary) is attached to its nearest neighbor that is on the network. For details, consult the source code listing or the article by Gower and Ross, "Minimum Spanning Trees and Single Linkage Cluster Analysis," *Applied Statistics*, 18, 54-64, and the associated algorithms (copy attached).

### IV. FEEDER MODULE (FEEDER.PAS)

This module optimizes the feeder-subfeeder arrangement in each quadrant. It will try all combinations of subfeeder arrangements from one through the number of serving areas, and choose the cost-minimizing configuration. As it calculates feeder cost, it will optimize the technology choice for serving each S.A. by determining the cost minimizer subject to any engineering constraints. To accomplish this, we use the Technology and Terminal modules, described below.

#### Optimize\_feeder\_arrangement

```
FOR QUADRANT := 1 to 4 DO
  BEGIN

    if num_SAs > 1 then
      sort2(num_SAs,dist,q_SA_array);

      { q_SA_array is now sorted in ascending order according to X distance }
      { (quadrants 1 and 3) or Y distance (quadrants 2 and 4) from the switch. }

      { Now create subfeeders along the midpoints of the 1/i quantiles, with i }
      { going from 1 to q1_SAs. }

      if num_SAs > 0 then
        for i := 1 to num_SAs do { i will index arrangements of subfeeders }
          begin
            for j := 1 to i do
              begin
                Get quantile breaks
                Set subfeeder locations equal to midpoints of each quantile
              end; { for j }

            { Now, in quadrants 1 and 3, walk down subfeeders from North to South first, and then }
            { from South to North, both ending at the main feeder. In the other quadrants, do the }
```

*{ same thing but from East to West. }*

for n := 1 to num\_SAs do find nearest subfeeder to each SAI;  
adjust subfeeder location s.t. structure cost of connecting to it is minimized;

calculate\_quadrant\_density;  
optimize feeder technology;  
calculate\_quadrant\_structure\_costs;

for j := i downto 1 do *{ j indexes subfeeders for arrangement i }*  
begin

sort SAIs north of main feeder;  
*{ Do subfeeder north of main feeder }*

for k := c1 downto 1 do  
*{ k indexes SAs on this subfeeder, north of main feeder }*  
begin

bring SA lines to subfeeder;  
bring subfeeder to either next point of contact with SA or main  
feeder;

end; *{ for k }*

*{ Do subfeeder south of main feeder }*

for k := 1 to c2 do *{ k indexes SAs on this subfeeder, south of main feeder }*  
begin

bring SA lines to subfeeder;  
bring subfeeder to either next point of contact with SA or main  
feeder;

end; *{ for k }*

Now bring main feeder either to next subfeeder point of contact or to  
central office;

End; *{ for j -- go to next subfeeder }*

If configuration I is cost minimizing, then save it;

End; *{ for I - try another subfeeder configuration }*

END; *{ for quadrant - go to next quadrant }*

## V. TERMINAL MODULE (TERMINAL.PAS)

### Fiber\_terminal\_cost\_fn

This function solves the problem

$$\text{Minimize FTC} = a_{2016} + b_{2016} * L_{2016} + a_{672} + b_{672} * L_{672} + a_{96} + b_{96} * L_{96} + a_{24} + b_{24} * L_{24} + \text{cable\_cost}$$

N2016

N672

N96

N24

Subject to:

$$L_{2016} = \min(2016 * N_{2016}, \text{lines})$$

$$L_{672} = \min(672 * N_{672}, \text{lines} - L_{2016})$$

$$L_{96} = \min(96 * N_{96}, \text{lines} - L_{2016} - L_{672})$$

$$L_{24} = \min(24 * N_{24}, \text{lines} - L_{2016} - L_{672} - L_{96})$$

Where

N2016 = number of 2016-line terminals

N672 = number of 672-line terminals

N96 = number of 96-line terminals

N24 = number of 24-line terminals

L2016 = number of lines on 2016-line terminals

L672 = number of lines on 672-line terminals

L96 = number of lines on 96-line terminals

L24 = number of lines on 24-line terminals

a2016 = fixed cost of 2016-line terminal

a672 = fixed cost of 672-line terminal

a96 = fixed cost of 96-line terminal

a24 = fixed cost of 24-line terminal

b2016 = per line cost of 2016-line terminal

b672 = per line cost of 672-line terminal

b96 = per line cost of 96-line terminal

b24 = per line cost of 24-line terminal

cable\_cost = cost of cable given terminal configuration (number of fibers = number of terminals x 4)

lines = total DS0 lines in distribution area

## T1\_terminal\_cost\_fn

This function solves the problem

Minimize T1TC = ac96 + bc96\*LC96 + ac24 + bc24\*LC24

NC96

NC24

Subject to:

LC96 = min( 96\*NC96, lines )

LC24 = min( 24\*NC24, lines - LC96 )

Where

NC96 = number of 96-line terminals

NC24 = number of 24-line terminals

LC96 = number of lines on 96-line terminals

LC24 = number of lines on 24-line terminals

ac96 = fixed cost of 96-line terminal

ac24 = fixed cost of 24-line terminal

bc96 = per line cost of 96-line terminal

bc24 = per line cost of 24-line terminal

lines = total DS0 lines in distribution area

## VI. TECHNOLOGY MODULE (TECH.PAS)

This module calculates a provisional feeder cost for each technology type for each serving area grid, and chooses the cost minimizer, subject to the constraint that no engineering crossover point is violated.

```
c26 := feed_cable_cost( copper26 ) + analog cross-connect cost;  
c24 := feed_cable_cost( copper24 ) + analog cross-connect cost;  
ct1 := feed_cable_cost( t_1 ) + T1 terminal cost; { optimized }  
cf := feed_cable_cost( fiber ) + fiber terminal cost; { optimized }
```

```
technology := copper26;
```

```
if ( c24 < c26 )  
or (feeder_distance + MaxDistance > copper_gauge_xover)  
then technology := copper24;
```

```
if ( ( ct1 < min( c24,c26 ) ) )  
or (feeder_distance + MaxDistance > max_copper_distance)  
or (feeder_distance > copper_t1_xover)  
then technology := t_1;
```

```
if ( cf < min( min(c24,c26), ct1 ) )  
or (feeder_distance > t1_fiber_xover)  
then technology := fiber;
```