

Assuring Connectivity in an Electric Utility GIS Distribution Model

A requirement for completion of the
Master of GIS (MGIS) program at the
Pennsylvania State University

Matthew D. Coram, GIS Analyst

Murfreesboro Electric Department, TN

Advisor: Dr. Amy Griffin, Pennsylvania State University

April 28, 2017

Introduction

Many electric utility distribution models have for years followed a traditional format, in that they have had a numbering system within their GIS that both uniquely identifies each section of line and its source, often termed the “upstream” neighbor. A section of line can best be thought of as a single span, such as one that runs from pole to pole for overhead facilities, or from ground-level cabinet to padmount transformer, etc., for underground lines. Only those unique numbers or a text-based identifier, stored as a database attribute value, were used to determine how each piece was connected to another. A model like this has persisted for many years, though some vendors’ standards have changed. A move has been seen recently that transitions from this single-attribute format to one that references multiple database tables, similar to ESRI’s geometric network model.

A change such as this could be the result of several factors. One such reason ensures that only the vendor-supplied tools are used to make changes within the GIS, both with the on-screen mapping feature and in those related geodatabase tables which govern the connectivity relationships among features. An additional consideration for storing related data within multiple tables is to normalize the data, where the data are grouped together based upon their purpose. Connectivity data stored separately may be used to perform upstream or downstream tracing, to quickly determine or verify how a particular line feeds throughout a city. A schema such as this would increase the database efficiency by reading only the connectivity data and would therefore decrease the overhead of retrieving entire tables of data which are unnecessary to perform these network traces. Only data which are needed to tie the two tables together are included in the feature class database table.

Since the Outage Management System (OMS), whose data are always updated from current GIS data, relies on data with a high degree of phasing and connective accuracy, edits should always be made with the vendor tools. Any connectivity or relationship errors may thus be avoided, so that outages may be pinpointed and a count of all affected customers may be obtained, both for service crews and for governmental reporting. The term “phasing” refers to a group of three separate lines which must be tracked carefully and their order maintained. The standard within many countries is a three-phase electrical system, which consists of a group of three lines plus a neutral wire. All three phases are fed from a substation and are kept together on a pole to serve larger customers. This configuration is typically maintained until smaller residential streets may be adequately served with only two wires. Any

spatial location where two circuits meet and may be connected temporarily should always have matching phases, that is, A phase from one circuit to A phase from another circuit.

Statement of Problem

Within the last several years, limitations of Murfreesboro Electric Department's (MED) previous GIS (and other key systems) helped to convince our management that a needed upgrade was on the horizon. In moving toward these new systems, it was discovered that the database schemas of the older and newer systems were dramatically different in many ways. The issue of connectivity within the GIS model was near the top of the list of concerns and this concept was discussed on many occasions throughout the software conversion. One particular difference between the two systems is that the older GIS did not require the endpoints of two separate line segments to be snapped together in order for them to be considered electrically connected. In contrast to this legacy system, the newer framework does have this requirement of snapped endpoints, and during the conversion process, these cases of non-intersecting features were rectified through a combination of automated tools and manual updates. Unfortunately, specific instances are still being discovered where lines are disconnected and connectivity is either completely broken, or junction points are misconfigured. Compounding the problem is the fact that these errors do not always appear on the vendor's bundled reports. Not only do these configuration errors affect the proper operation of the OMS, especially during extreme weather events, they come into play as new lines are constructed and drawn in the GIS. Furthermore, the engineering analyses used to ensure the system is adequately sized for growth may also be adversely affected. As these configuration issues are discovered, valuable time must be devoted to researching the proper configuration and then correcting it manually within the GIS. The concerns related to the time needed for manual identification of errors and their correction will be discussed in subsequent sections.

It should also be mentioned that lines that meet at a particular spatial location are not necessarily electrically connected to one another. A specific example of this configuration is when two separate circuits are located on the same pole, often called a "double circuit pole" within the utility industry. Within the GIS, these double circuits, which are shown as offset lines in Figure 1, can have the appearance of being connected at the pole, though they must be regarded as two individual sets of features. When viewed in the field, the difference can easily be seen as the circuits have a vertical separation on the shared pole and are, therefore, electrically isolated from one another. In addition to

the electrical lines, protective devices are a part of the utility's model, are classified as point features, and occur at junction points.

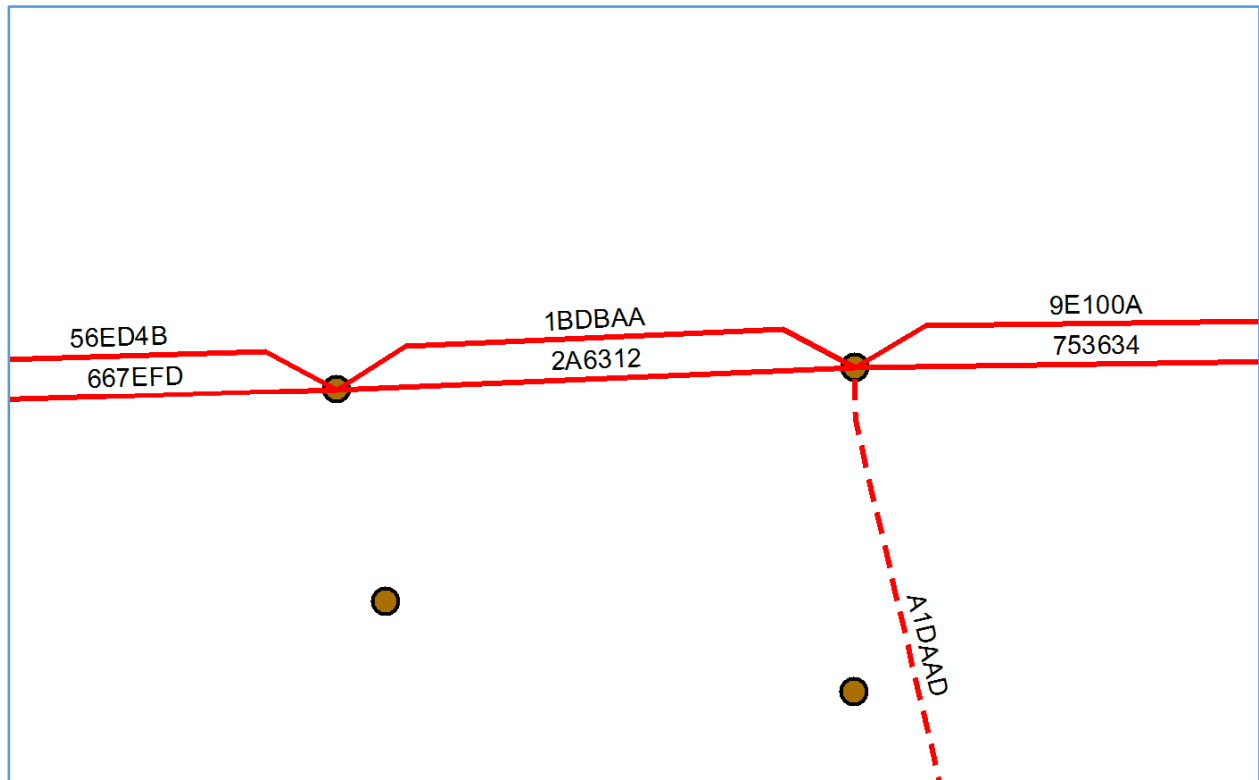


Figure 1, an example of the GIS representation of a double circuit, with one circuit on top (Span 1BDBAA) and offset from the second, separate circuit (Span 2A6312).

Classification of Features

There are similarities and differences between our model and an ESRI-based Geometric Network. Spans in a geometric network are termed “edges” and may be categorized as simple or complex edges. Complex edges may be thought of as being similar to a water main, which can have one or more taps that are spliced into it at single points along its length. The other type, a simple edge, is analogous to a service line that connects to an individual home or business, with no taps branching from itself. Thus, the entire flow, which is measured at the starting point, is equal to the flow measured at the ending point, minus any internal losses, called the impedance.

In our specific model, all lines are of the simple edge type from the ESRI geometric network specifications. These lines may not be tapped into along their length, and any branches will occur only at a line's endpoint. In those instances where a new tap is required along an existing simple edge, it is split into two new simple edges at the branch point and the new lateral line is then constructed.

Table 1 lists all of the symbology used within the map document, along with summaries detailing the purpose of each device. Of particular interest in this table are the spans and stations, which form the basis for establishing electrical connectivity within the GIS model.









Component	Symbol	Purpose
Capacitor Bank		A device which counteracts the effect of inductive loads, such as electrical motors.
Electric Station (Substation)		A collection of equipment which lowers voltage for distribution to a large number of customers.
Open Point		A semi-permanent break in an electrical circuit.
Overcurrent Device (Breaker / Feeder Bay, Recloser, or Fuse, respectively)		A device which prevents the flow of too much electrical current.
Span (Primary Overhead, Primary Underground, respectively)		A collection of two or more wires for conducting electricity.
Station (Junction)		The intersection of one or more electrical spans and / or equipment.
Support Structure (Pole)		Any structure which raises electrical lines and equipment to a pre-determined height to ensure safety of the public.
Surface Structure (Vault)		A partially buried container for housing electrical junctions.
Switch		A device which either interrupts (Open State) or conducts (Closed State) an electrical current.
Voltage Regulator		A device which raises or lowers the electrical voltage in response to varying loads. Larger loads will lower the voltage, potentially resulting in quality issues with a customer's electrical service.

Table 1, A listing of the utility's symbology and the purpose of each device type

Application Purpose

Because of what ultimately amounts to inconsistencies between database tables, it was desirable to devise a software tool that could compare the endpoints of each line segment with the database's connectivity records. As the data were imported from a Microsoft SQL Server database into Microsoft Access, it was a straightforward process to use the Visual Basic for Applications (VBA) development environment that is built into Access for coding this data cleansing tool. As this tool is designed to work in a minimally-modified database format, many more numerous steps to convert it to a geometric network format and to obtain that data type's functionality may be avoided. This is particularly important if and when the application needs to be run on a recurring basis. Keeping the format as close as possible to our original database schema will avoid the time-consuming work of ad-hoc conversions.

At this time, the application only analyzes lines and equipment on the primary network. While errors may exist in both the primary and secondary networks, the primary network was chosen as a priority for the scope of this Capstone Project. Primary and secondary networks may best be thought of in terms of high and low voltage, respectively. Primary lines are used to deliver power from the electrical substation to a point along a street where it then connects to a transformer. This transformer steps the voltage down to a safe and useful level for one or several customers. The transformer is considered an intermediary device and bridges the primary network to the secondary network, which carries the lower voltages into individual homes. Within a secondary network, any errors will typically affect between one and ten customers. In contrast, errors within the primary network will affect a given line section and all of its downstream neighbors, which may ultimately affect many more customers. Pinpointing those specific spans with errors will help to increase confidence in the various utility models that govern our daily operations.

The entire issue of correcting these system-wide errors may be thought of as a data cleaning operation. With regard to Murfreesboro Electric's GIS database, data cleaning here refers to making certain that the endpoints of lines and the insertion points of features that are electrically connected to one another, as defined by the stand-alone connectivity table, are coincident or snapped together. This type of error may be classified as a "single-source problem" at the "instance level" (Rahm & Do, 2000). As these correctly-formatted endpoint coordinate data are stored within a database that has an existing schema that governs the type of information stored (numerical, coordinate data), the error is not classified as a "schema-level" error (Rahm & Do, 2000).

For features which have been imported into the GIS database from other systems, a means of verifying and reporting on the data quality is essential. In the case of Murfreesboro Electric, data about electrical network connectivity was imported from a single legacy system, but from multiple sources (geodatabase tables) within that system. Inconsistencies within data sources, whether from multiple systems or from a single, multipart system, may prohibit the effective use of analyses to answer organizational queries (Raman & Hellerstein, 2001). A question that is commonly asked by local newspapers during storms is, “How many people are currently without power?” The Outage Management System that is used by Murfreesboro Electric is capable of quickly answering this question, but an accurate answer may only be expected when the connectivity is properly configured; that is, if the quality of the utility data is strong.

Had the application been able to directly interact with our working database, it would have been possible to allow the program to automatically correct certain types of errors. Parameters which govern the extent to which these automated changes could be made would be necessary to include within the application. Such constraints would likely include allowing the application to only move a line’s end point or insertion point within a specific distance. Additionally, the updates would only be applied to specific cases of errors, such as connecting two line segments. Any automated changes should be logged so that a sample of corrections could be scrutinized for accuracy after the analysis is complete. In those cases where more than two features join at a single point, an analyst would need to conduct in-office research or potential field work to determine which lines are truly connected, and to manually make those updates. The feature dataset of errors and the original electrical lines can be used in an Overlay Analysis (O’Sullivan, 2016) that will allow the editor to correct any anomalies in the connectivity. Lastly, in the feature dataset that is managed by the utility, each database record could include an attribute field that could be altered by the user to signify that the error has been corrected.

Application Logic

As mentioned earlier, all connectivity data is stored in multiple database tables (SQL Server is used to house all of the utility’s GIS data). Each object within the database’s tables is given a randomly-generated Globally Unique Identifier (GUID) of 32 hexadecimal characters plus four hyphens and enclosing brackets, so that each feature may be uniquely referenced. Though each feature class’ entities are stored in one table (with additional, hidden tables that store related spatial data (ESRI, n.d.)), the connectivity relationships are defined by the Entity Relationship Diagram (ERD) shown in

Figure 2. Included in this model are not only the underlying records containing the spans themselves, but also feature intersections called “junctions” or “stations.” These junctions occur at both the endpoints of each span and the insertion point of other primary and secondary equipment, such as fuses or meters.

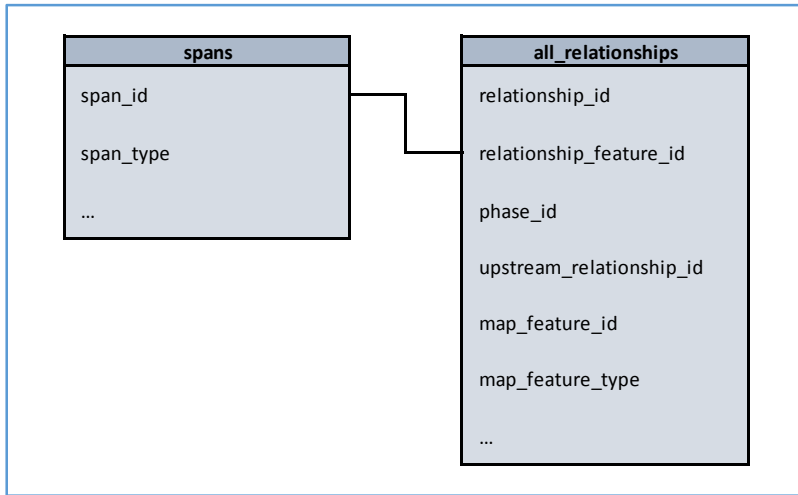


Figure 2, an Entity Relationship Diagram detailing the distribution model's connectivity

As the application steps through the utility's data model, it analyzes features in groups of two and will encounter several different classifications of these “feature pairs”. They will always be some combination of junctions and spans.

Spans may be connected to:


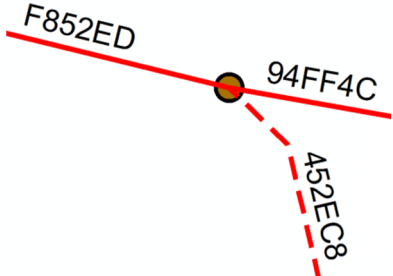
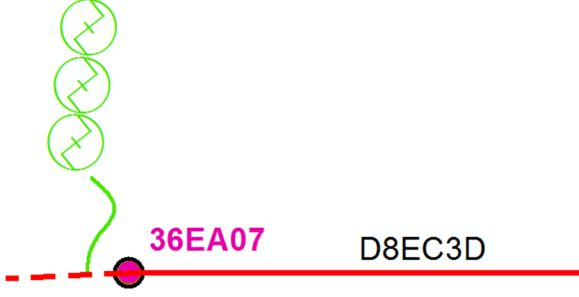

- Spans,
- Multiple spans, or
- Junctions with electrical equipment, such as substation transformers, fuses, capacitors, or reclosers

Junctions and their related equipment may be connected to:

- Spans or
- Multiple spans

Junctions only serve to connect equipment and spans. As electrical power moves along each span from pole to pole, for example, junctions are only necessary when the current is directed through one of

several types of devices. A common example is that of a fuse being inserted to serve a lateral line. Furthermore, junctions are never connected directly to other junctions, as there will always be one or more spans between them. A summary of the potential feature pair combinations is shown in Table 2.

Feature Pair	Description	Example
Single Span to Single Span (poles are only shown to denote span endpoints)	Span 814993 (upstream feature) connects to Span 5A9ACB	
Single Span to Multiple Spans	Underground Span 452EC8 (upstream feature) connects to both Overhead Spans F852ED and 94FF4C	
Single Span to Junction	Span D8EC3D (upstream feature) connects to Junction 36EA07 (the bank of three fuses are related to this junction)	
Junction to Single Span	Junction 36EA07 (upstream feature) connects to Span 7EA14A	

<p>Junction to Multiple Spans</p>	<p>Junction E611DA (upstream feature, with two related fuse banks) connects to Spans 73C7E8 and 453DD2</p>	<p>The diagram illustrates a junction feature E611DA (pink dot) connected to two spans: 73C7E8 and 453DD2 (dashed red lines). A main line CDFD32 (solid red line) is also shown. Two fuse banks, 6956F9 and 1CFE08 (green circles with X's), are connected to the spans.</p>
-----------------------------------	--	--

Table 2, A Listing of the potential classes of feature pairs within the utility model

Within the geometric network format, ESRI has also provided a junction feature that directs the flow from each edge to its adjacent neighbors. Without these junctions acting as a binding feature among edges, operations that trace all connected upstream or downstream elements would not be possible. Functionality similar to this in MED’s OMS is what allows the system to predict where service crews may begin their search for downed power lines during emergency events.

To best explain the ERD, it is fairly straightforward to use the standard format of [table name].[field name]. For example, when referring to the span_id field from the spans table, it would be written as [spans].[span_id]. The left table, spans, is equivalent to what is expected for a standard feature class within a geodatabase. If we were to construct a new feature class in ESRI’s ArcGIS for Desktop software, this would be the typical attribute table that a GIS user would encounter. This span_id value is unique and allows the user to pinpoint an individual span for any necessary correction after it has been processed.

However, in order to model the connectivity, the addition of another stand-alone table, derived from the working geodatabase, is necessary. Moving from the left to the right table, the [spans].[span_id] field, which contains the primary key values, is equivalent to the [all_relationships].[relationship_id]

field, which contains the foreign key values. With the features defined in this manner, the program is able to analyze a subset or the entire range of electrical features by moving from the source to the each downstream feature.

Program Operation

As shown in the attached flowchart (Figure 3), the application begins by finding the first substation or the first segment of the circuit that is chosen by the user, which is at the source end of an electrical line. The process is also shown within the companion video, which demonstrates the application analyzing MED's entire electrical system. The circuit begins at a junction and is connected to a primary network feature termed a busbar. The substation (`gs_electric_station`) is identified from the `[all_relationships].[map_feature_type]` field and the application begins the search for the next downstream feature by staying on the same record and retrieving the value from the `[all_relationships].[relationship_id]` field. The retrieved value is then searched for in the `[all_relationships].[upstream_relationship_id]` field. Stated another way, the program asks the question: the substation is the source for which downstream feature(s)?

The steps above have now located two features: a junction and a span, along with its start point. In knowing which junction and which span are tied together from a connectivity standpoint, the application can use the junction's insertion point and the starting coordinate values of the span stored in the `all_relationships` table of the geodatabase in its separation distance calculations. If those coordinates are coincident, then the first feature (in this case, the substation) is marked as processed, the `check_code` is marked with a value of 1 (to indicate that the coordinates are coincident), and the procedure repeats to check the span and the next downstream feature located at the junction of the span's endpoint. This process continues, analyzing each set of features as a pair, until each object on the entire electrical system has been checked.

At those points where an upstream feature has multiple downstream features, an additional field in the database is marked to indicate the existence of a branch point. As features on one branch of the circuit are completed, the application returns to these branch points and follows the other, unprocessed lines of the adjacent branch to their completion. Given the complexity of the electrical distribution system, there are many of these branch points that must be carefully tracked so that no branches are omitted. Without the analytic capabilities of this program, it would be an extremely time consuming (or

impossible) task to check each one of the 55 circuits and approximately 24,807 primary spans and their adjacent features.

All features are initially coded as unprocessed. As the application processes each feature, a database attribute is updated to reflect each line or junction's status; those features which retain their initial status after the program has completed can be examined by the user to determine the cause as to why they were not processed.

Also shown in the flowchart are two subroutines, apart from the main procedure, which is named Run_Flow_Checks(): Store_Feature_Information() and Process_Coordinates().

Store_Feature_Information() is used to record the values of either the starting X and Y coordinates (in the case of junctions) or the ending X and Y coordinates (in the case of spans), based upon the upstream feature type of the current feature pair. The calculation must have the correct coordinates for determining the correct separation distance between the set of features currently being analyzed.

In the Process_Coordinates() subroutine, the actual separation distance calculations are carried out. Based on the results of this process, the feature is either marked with a check_code value of 1, signifying that the features' points match, or with a value of 2, in cases where their points do not match.

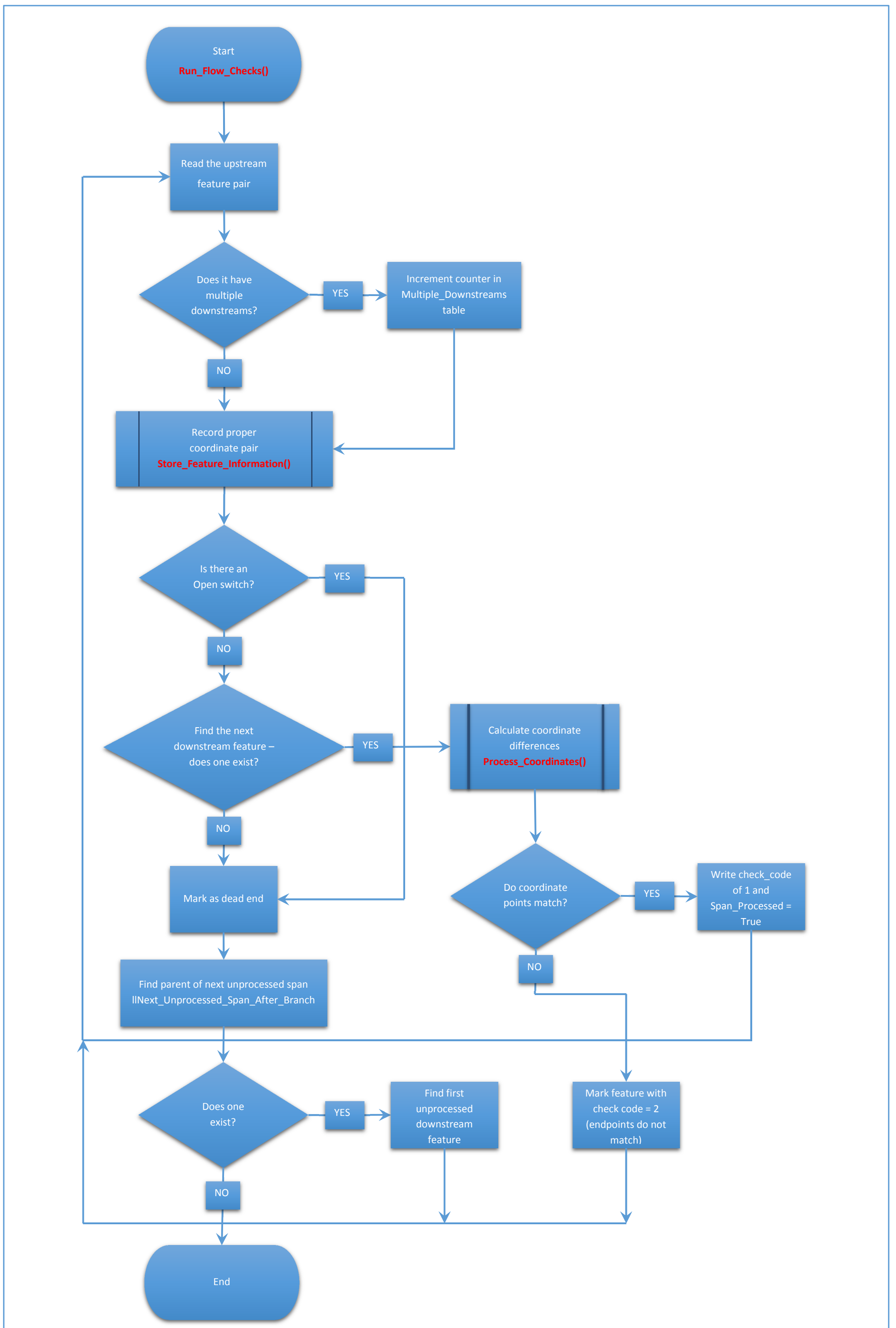


Figure 3, A flowchart detailing the application's operation

It is probably worth noting here that there are, arguably, several ways to simplify these relationships. One possibility would be to return to the older method of storing a unique number for every feature, along with its related upstream feature. In order to trace the path of the entire line, these two fields could be read by an automated process to determine which feature serves a particular neighboring feature.

In Figure 4, an underground line segment has been disconnected from its source, but the database's connectivity records indicate that it is still connected. Here, the start point of the underground primary line labeled F97FB7 should be connected to the end point of line C05DC2, though they are clearly not snapped together. It is possible that this line was temporarily moved and had not been manually corrected before the database conversion process was begun. The checks that can be run within the vendor software would likely not flag this line segment as an error, since its connectivity records are still intact.

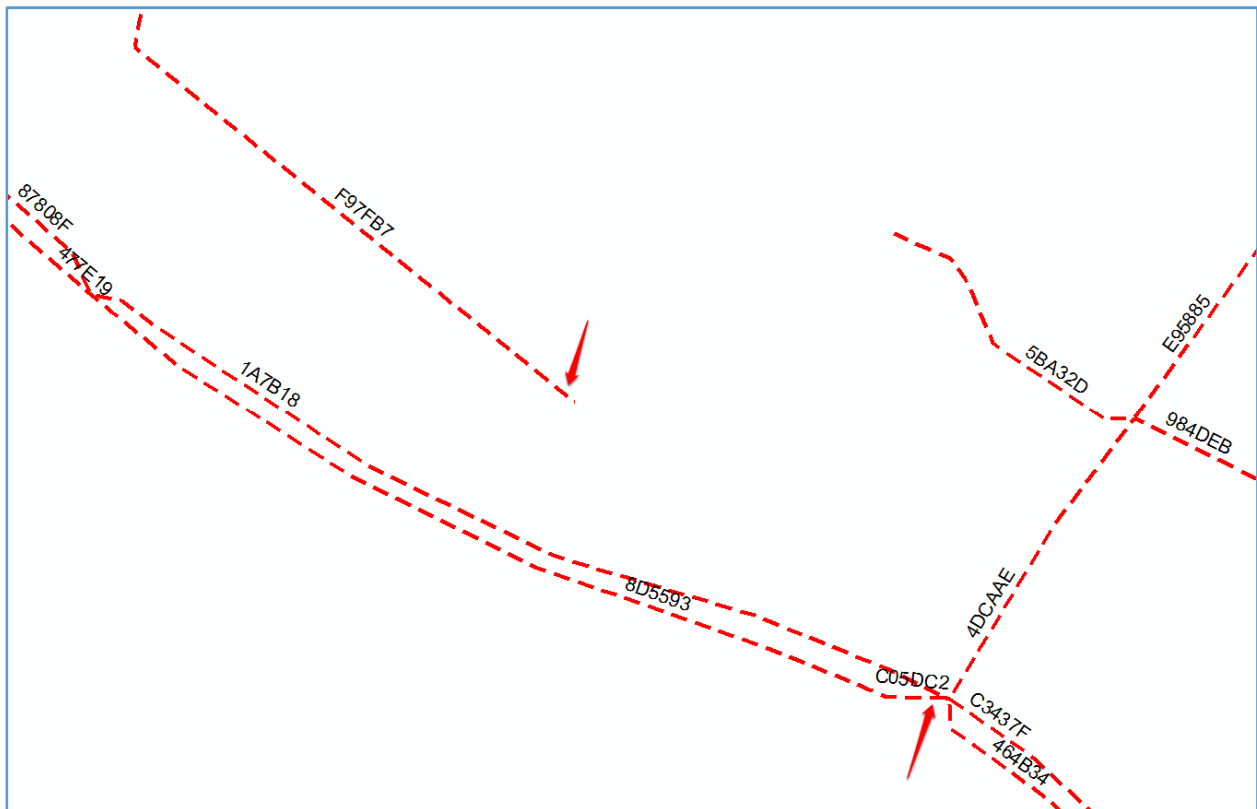


Figure 4, a screen capture of two line segments which have been disconnected in the map view, but not in the underlying connectivity records

In Table 3, the records for the all_relationships table for the two spans depicted in Figure 4 are shown. The start and end points for each span were calculated and stored in the database tables by using the Calculate Geometry tool in ArcMap and may be seen in the lower portion of Table 3. Though their connectivity records are correct, the starting coordinates of span F97FB7 do not coincide with the ending coordinates of span C05DC2.

The image shows two screenshots from Microsoft Access. The top screenshot shows the 'all_relation' table with the following data:

Abbrev_span_id	Abbrev_upstream_relationship_id	Abbrev_relationship_id
F97FB7	6365C4	F46AB7
C05DC2	802960	6365C4

The bottom screenshot shows the 'ships' table with the following data:

Start_X_Coordinate	Start_Y_Coordinate	End_X_Coordinate	End_Y_Coordinate
1820201.48101376	556906.215288041	1820005.02998437	557063.130041575
1820372.51429072	556741.041913141	1820412.91995342	556736.480061933

Table 3, all_relationships table (in two pieces) examined with Microsoft Access

Thus it can readily be seen that the two lines are separate from one another, but the database relationships still indicate the connection is present. It is this detection functionality that the custom program offers.

In the case of point features, the span is analyzed to see if its endpoint is snapped to the feature's insertion point; these point features only have coordinates stored in the Start_X_Coordinate and Start_Y_Coordinate fields. In both cases, the database tables are also checked to make sure that the GUID of each span or device mirrors the information found in the connectivity tables. If this piece of the analysis finds that the spans' records are in agreement, it marks that particular database record as checked and correct, and then moves to the next downstream feature. This process continues along the entire electrical circuit until the circuit's endpoint is reached.

Results

Since the application codes each instance of an error, the number of errors throughout the set of processed spans was able to be accurately reported. The total program run-time for all features which were successfully processed took approximately 15 minutes. This amount of time will likely vary, depending on the configuration of the computer upon which it is run. For this study, the application was run on the author's computer and there were 20.9 feature pairs analyzed per second.

An unexpected issue with the program's execution was the fact that not all features on the network were successfully analyzed. As the application was written to begin with the most upstream feature and then work its way downstream to the end of each line, those features besides substations which are improperly configured as having no source will be bypassed. In reality, these features should be considered as incorrectly configured and must be their electrical flow must be reconfigured to reflect actual connectivity in the field for the program to properly function. Stated another way, any non-substation source will result in one or more features not being successfully processed.

Once the proper flow is restored within the electric model through research and manual correction, the application can be expected to process the full set of features and that run-time is predicted to be approximately 19 minutes 45 seconds. The number of features analyzed per second should remain constant.

Upon examination of the database table after the program completed, a total of 24,807 primary features existed. Of those, 18,847 (75.97%) features were processed, with 5,960 (24.03%) features which were not successfully analyzed. The complete number of errors within the successfully-processed set was 3,693. If this error rate of 0.196 errors per feature holds true, we can predict that the number of errors throughout the entire system will be 4,861.

The type of errors within the processed feature set may be classified as those where connectivity is indicated, but where the end and start points, respectively, of each feature pair are not coincident. This type of error corresponds with a Check Code of 2; the complete list of Check Codes is shown in Table 4. Within this error class, there were 3,602 (97.54% of the total error count) features. The other, existing type of error is indicated when a single feature lists multiple spans or devices as being its source and of

this type (Check Code = 4), there were 91 instances (2.46%). While the application was designed to report upon the number of features not snapped together, it was necessary during the course of the project’s development to also include those errors where features were configured to have more than one source. While this particular error may be reported upon by a tool provided by Murfreesboro Electric’s vendor, the report had not been generated and the 91 instances were unexpected.

Check Code	Description
0	Feature has not been checked.
1	Feature endpoints match.
2	Feature endpoints do not match.
3	No downstream primary span exists.
4	Span has multiple upstream neighbors.

Table 4, A listing of the application’s Check Codes, which indicate each feature’s status after processing

It is doubtful that many of these errors would have been located without the use of the program, unless they were discovered through adding a new line section to one of the misconfigured features, or when an outage occurred at that span. This is revealed within the OMS when a known set of customers are without power, but they fail to appear on detailed reports about the outage. As the user is working within the GIS software, the error may be visible as a feature which has no source or upstream neighbor. Many of the spans within this error class have a separation distance between their endpoints that is less than 1 foot. At the map scales typically seen in our GIS (1:1,200), this separation is very unlikely to be noticed from a visual inspection alone, unless the user significantly increases the magnification within the map.

It is important to note that the 5,960 unprocessed features can most likely be attributed to the flow being reversed in key spans. Since the application is designed to begin with the source feature on every circuit and progress downstream, any scenarios where a line has this flow reversed will not allow the

program to properly advance. In essence, every span and device from that point downstream will be bypassed as the program scans throughout the system.

An example of this is shown in Figure 5 where Span 84659A feeds both Spans 737E51 and DE109A. The program is able to process the path from Span 84659A through Span DE109A and continue to Span 9A7352. However, when it returns to analyze Span 737E51 and expects its source to be Span 84659A, it finds that its source is incorrectly configured as Span 59B411 and is unable to continue through that span to analyze the remaining features. The end result is that everything which is truly downstream from Span 737E51 remains unprocessed. From this example, it can be shown that a single misconfiguration can readily impact the analysis of a very few or many primary features. In the related video, this is apparent as all primary lines are normally colored red, and those lines which have been successfully analyzed are highlighted in blue. For the application to be successful, it is necessary to run connectivity checks from within the vendor software to ensure that no features are omitted from the analysis. For this reason, it may be necessary to alternate between running this application as well as the vendor flow checks, to ultimately reach the goal of successfully processing every feature.

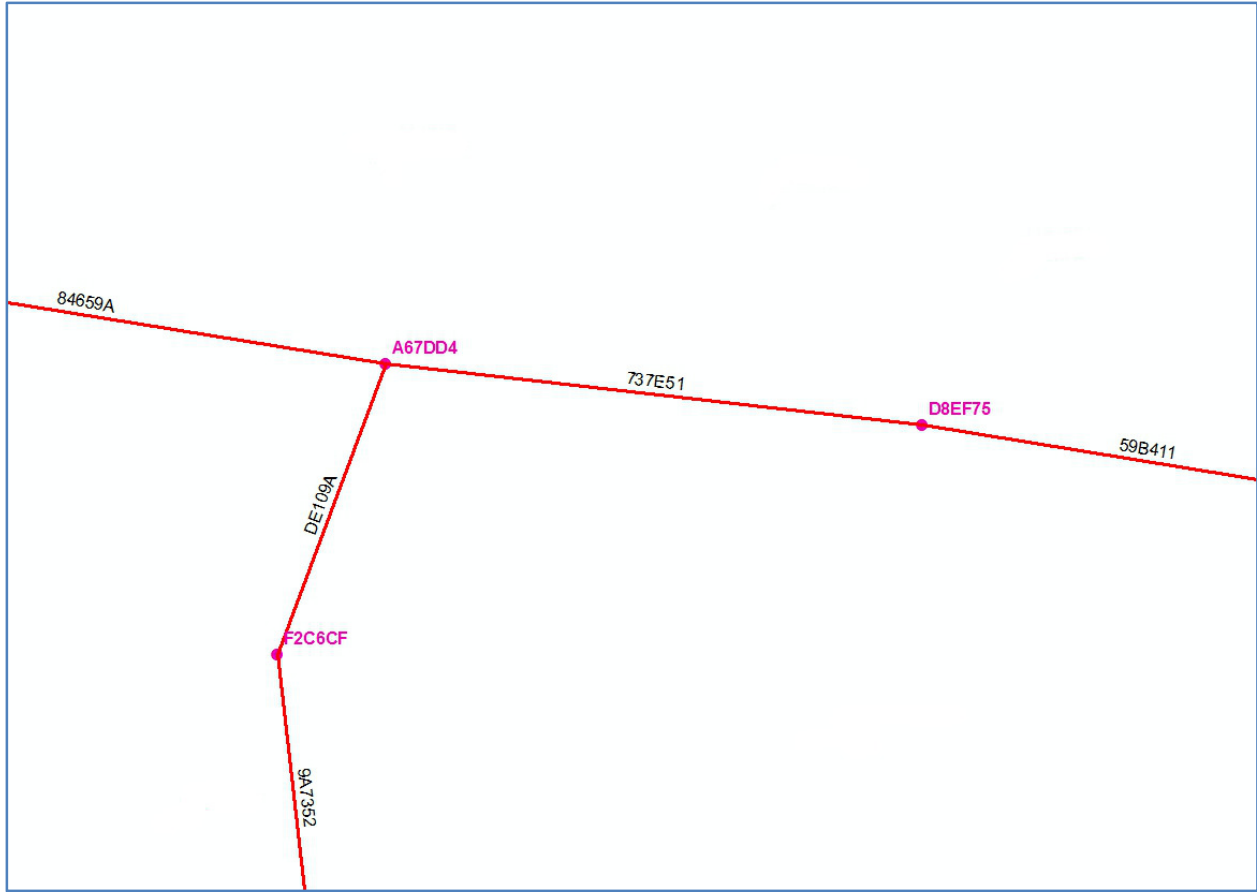


Figure 5, An example of an improperly configured span, resulting in reversed flow

During the initial stages of this project, the anticipated number of errors could not be accurately calculated. However, after completion of the program, there was a certain amount of disbelief that the number of errors was so large. Manually increasing the magnification and examining each junction on the entire system would have been a very daunting task and one that could not be easily completed by a single analyst. If an estimate on the amount of time to examine the junction between each feature pair could be made, a reasonable amount of time would be 30 to 45 seconds.

For an examination taking 30 seconds, it may be calculated that $(24,807 \text{ features} * 0.5) / 60 = 206.7$ hours, or 5.17 working weeks. Using that same method, the 45 second examination time would yield 7.75 working weeks. As this time is only for inspection, the time to correct any errors that are found would take additional time.

With the number of errors now quantified, the average amount of time needed to correct a typical incident yields some idea of the scope of the corrective action needed. On the lower end of the range, we can use a 5 minute repair time and a count of 4,861 incidents, which would yield $((5 * 4,861) / 60) = 405.08$ person-hours (or approximately 10.13 work weeks) for correction of these errors. To calculate the maximum time needed for correction, we can use 10 minutes to correct each issue and a count of 4,861 individual incidents to conclude that $((10 * 4,861) / 60) = 810.17$ person-hours (or approximately 20.25 work weeks) to correct.

Within any business, there is the need to consider how much Return on Investment may be expected for any given project. Since the GIS is one of the more prominent repositories for the records of the utility's field inventory, it is essential that accurate records are kept. An additional consideration is that of the OMS, as it operates on the data that is imported from the GIS to make accurate predictions about outages. When this connectivity is broken, service crews could conceivably be routed to areas where a problem does not truly exist. This undue expense can certainly not only be a drain on the resources of a utility, but it can divert service personnel from repair of true issues.

An additional drawback to disconnected features is that of user confidence. Since the proper operation of the GIS is of great importance, users who are relatively new to the GIS could encounter these errors and lose confidence in the integrity of the entire database. Those systems, which are carefully maintained, should accurately reflect the current state of the entire electrical system. While minor errors may be anticipated, lines which have a sizeable distance between endpoints which should be snapped together could reduce the overall usability of the system. Quality therefore is a key factor in the development and maintenance of the GIS. Given the importance of having accurate GIS data, the investment into writing this program to pinpoint errors has proven to be very worthwhile. The time and effort required will certainly help to ensure that the confidence in the GIS and OMS data are bolstered, and that users will find outage predictions and information gathered from reports to be very useful when restoring service and communicating with interested media services. Other publications mention the importance of data quality, so that any conclusions based upon that data are judged to be sound (Rahm & Do, 2000). We can further conclude that checks and balances within any digital system are required for proper operation.

Future Enhancements

The program could benefit from incorporating additional capabilities so that different causes of errors may be examined. At this point, the program stops execution once it reaches certain types of configurations. One specific error occurs where a fuse should only protect a single downstream span, and is misconfigured in our model so that it protects two downstream spans simultaneously. In this particular instance, the fuse should only protect a single span, though there are situations where a single device can serve two downstream sections. In practice, these instances should be marked for later review by the analyst.

The use of VBA for coding this project was relatively easy to perform, though a more robust platform could be used to further develop the application. Within the VBA environment, individual variables can be observed as they change throughout the course of the application's operation, but grouping the variables onto a user form where they may be more easily seen would even more helpful. By coding the application with another language, such as Microsoft's C#, the application can directly interact with ArcMap and provide additional functionality. A desirable function would include the ability to run the application downstream from a feature selected within the map document versus copying and pasting the value into Microsoft Access.

With the completion of this program, the author is looking forward to its analytical capabilities being put to good use. Since other systems connect to our GIS and receive their data from it, other departments are certain to reap the benefits of a strong and accurate electrical utility model.

Sources

Environmental Systems Research Institute, Inc. (n.d.). ArcGIS Help. Retrieved July 27, 2016, from <http://desktop.arcgis.com/en/arcmap/10.3/manage-data/geodatabases/design-an-overview-of-table-properties.htm>

Environmental Systems Research Institute, Inc. (n.d.). ArcGIS Help. Retrieved June 30, 2016, from <http://desktop.arcgis.com/en/arcmap/10.3/manage-data/geodatabases/exercise-5-building-a-geometric-network.htm>

Environmental Systems Research Institute, Inc. (n.d.). ArcMap. Retrieved June 30, 2016, from <http://desktop.arcgis.com/en/arcmap/latest/manage-data/geometric-networks/about-creating-geometric-networks.htm>

Environmental Systems Research Institute, Inc. (n.d.). ArcMap. Retrieved June 30, 2016, from <http://desktop.arcgis.com/en/arcmap/latest/manage-data/geometric-networks/a-quick-tour-of-geometric-networks.htm>

ESRI 2011. ArcGIS Desktop: Release 10.2.2. Redlands, CA: Environmental Systems Research Institute.

Gilgrass, C., & Hoel, E. (2012, July 26). Geometric Networks: An Introduction. Retrieved June 30, 2016, from <http://video.esri.com/watch/2012/geometric-networks-an-introduction>

Microsoft, Inc. (2010). Microsoft Access: Release 2010. Redmond, WA: Microsoft, Inc.

Murfreesboro Electric Department. (2016). MEDGIS01 [SQL Server Database]. Murfreesboro, TN: Murfreesboro Electric Department.

O'Sullivan, D. (2014). GEOG 586 - Geographic Information Analysis. The Pennsylvania State University. Retrieved April 5, 2016 from https://www.e-education.psu.edu/geog586spring2/l8_p3.html

Rahm, E., & Do, H. (2000). Data cleaning: Problems and current approaches. IEEE Data Eng. Bull., 23(4), 3–13. <http://doi.org/10.1145/1317331.1317341>

Raman, V., & Hellerstein, J. M. (2001). Potter's Wheel: An Interactive Data Cleaning System. *Data Base*, 01, 381–390. <http://doi.org/10.1.1.39.2790>

This document is published in fulfillment of an assignment by a student enrolled in an educational offering of The Pennsylvania State University. The student, named above, retains all rights to the document and responsibility for its accuracy and originality.

APPENDIX A

' The following code belongs with the Form frm_Connectivity_Assurance, which is called when the Microsoft Access database Connectivity2.mdb is opened.

```
Option Compare Database
Option Explicit
```

```
Private Sub Form_Load()
'Run this procedure when the form loads.
```

```
On Error GoTo ErrorHandler                                'Add error handling capabilities.
```

```
fraMain.Value = 1                                        'Make scanning the entire system the default option.
```

```
Exit Sub
```

```
ErrorHandler:
```

```
MsgBox Err.Number & " " & Err.Description                'Provide information to the user about the error type and description.
Err.Clear                                                'Clear the current error.
Resume                                                    'Resume with the next line.
```

```
End Sub
```

```
Private Sub cmdCancel_Click()
'Cancel and close the current form.
```

```
On Error GoTo ErrorHandler                                'Add error handling capabilities.
```

```
DoCmd.Close acForm, "frmConnectivity_Assurance"
```

```
Exit Sub
```

```
ErrorHandler:
```

```
MsgBox Err.Number & " " & Err.Description                'Provide information to the user about the error type and description.
Err.Clear                                                'Clear the current error.
Resume                                                    'Resume with the next line.
```

```
End Sub
```

```
Private Sub cmdBegin_Processing_Click()
```

```

'This procedure runs the application based upon the user's input.

On Error GoTo ErrorHandler

Dim iScan_Type As Integer
Dim stSpan_ID As String

stSpan_ID = ""

If fraMain.Value = 1 Then

    iScan_Type = 1

Else

    iScan_Type = 2
    txtGUID_Value.SetFocus

    If txtGUID_Value = "" Or IsNull(txtGUID_Value) Or Len(txtGUID_Value) <> 38 Then

        MsgBox "You must enter a valid GUID value", vbOKOnly
        Exit Sub

    Else

        stSpan_ID = txtGUID_Value.Text

    End If

End If

cmdBegin_Processing.Enabled = False

Module_Analyze_Features.Run_Flow_Checks iScan_Type, stSpan_ID

cmdBegin_Processing.Enabled = True

MsgBox "All spans have been processed.", vbOKOnly

Exit Sub

ErrorHandler:

```

```

'Add error handling capabilities.

'Value for storing the type of scan.
'Value for storing the GUID of the beginning feature for analysis.

'Initialize var.

'Scan_Type = 1 for analyzing entire system.

'Check for an empty text box or a value not equal to
' a standard GUID that is 38 characters in length.

'If acceptable, load the value into the var.

'Disable the command button until processing is finished.

'Call the function to begin processing, based on the user-selected scan type.

'Re-enable the command button.

'Alert the user that the application has finished.

```

```
MsgBox Err.Number & " " & Err.Description  
Err.Clear  
Resume
```

```
End Sub
```

```
'Provide information to the user about the error type and description.  
'Clear the current error.  
'Resume with the next line.
```

APPENDIX B

'The following code belongs with the Module Module_Analyze_Features, which is called from the Form frmConnectivity_Assurance.

Option Compare Database
Option Explicit

```
Dim mdb As Database 'Var for holding reference to database.
Dim mrsAll_Relationships As DAO.Recordset 'Module level var for holding recordset object of All Relationships query.
Dim mrsMultiple_Upstreams As DAO.Recordset 'Module level var for holding recordset object of Multiple Upstreams query.
Dim mstUpstream_Network_Feature_Type As String 'Var for holding the type of the upstream network feature.
Dim mdStart_X_Coordinate As Double 'Var for holding the starting X coordinate of the feature.
Dim mdEnd_X_Coordinate As Double 'Var for holding the ending X coordinate of the feature.
Dim mdStart_Y_Coordinate As Double 'Var for holding the starting Y coordinate of the feature.
Dim mdEnd_Y_Coordinate As Double 'Var for holding the ending Y coordinate of the feature.
Const msComparison_Value = 0.0000009 'Constant for holding the acceptable difference between coordinate values.
Dim msX_Coordinate_Difference As Single 'Var for holding the difference between floating point X coordinate values.
Dim msY_Coordinate_Difference As Single 'Var for holding the difference between floating point Y coordinate values.
Dim mvFR_Bookmark As Variant 'Var for holding the bookmark of a particular record.

Public Function Run_Flow_Checks(iScan_Type As Integer, stSpan_ID As String) 'iScan_Type and stSpan_ID vars are passed to this procedure - indicates
' which type of scan is needed by the user.

' Author: Matthew Coram
' Date: 3-1-16
' Purpose: To identify feature in an electric network which are noted to be connected to one another, but whose features are not snapped to one another.
' Features which are not snapped to one another can cause issues with broken connectivity, which can ripple into other, related systems.

On Error GoTo ErrorHandler 'Add error handling capabilities.

Dim rsSubstations As DAO.Recordset 'Var for holding recordset object of All Relationships query - for
' retrieving substations.
Dim rsMultiple_Downstreams As DAO.Recordset 'Var for holding recordset object of Multiple_Downstreams query.
Dim stSubstations_SQL As String 'Var for holding SQL statement.
Dim stAll_Relationships_SQL As String 'Var for holding SQL statement.
Dim stMultiple_Upstreams_SQL As String 'Var for holding SQL statement.
Dim stMultiple_Downstreams_SQL As String 'Var for holding SQL statement.
Dim stSubstation_DFN_guid() As String 'Var for holding dynamic array of substation Display Feature GUIDs.
Dim iCounter As Integer 'Var for counting through array of substations.
Dim stCurrent_Relationship_ID As String 'Var for holding the current relationship_id.
Dim stUpstream_Relationship_ID As String 'Var for holding the current upstream_relationships_id.
Dim lBookmarked_Record_Number As Long
Dim lRecord_Counter As Long
Dim bSwitch_Enabled As Boolean 'Var for holding the current switch's status - disabled switches act as a dead end.
Dim iNumber_Substations As Integer 'Var for holding the number of substations.

Set mdb = CurrentDb() 'Set the var equal to the current database.
iNumber_Substations = 1 'Initialize var.
```

```

If iScan_Type = 1 Then
    'For Scan_Type = 1, load substations for analyzing entire system.

    'Define and open the listing of substations with an SQL query.
    stSubstations_SQL = "SELECT * FROM All_Relationships WHERE [map_feature_subtype] = 'gs_electric_station'"
    Set rsSubstations = mdb.OpenRecordset(stSubstations_SQL, dbOpenDynaset)

    rsSubstations.MoveLast
    rsSubstations.MoveFirst
    'Enumerate the records by moving to the last record, then
    ' moving to the first record.

    iNumber_Substations = rsSubstations.RecordCount
    'Set the variable equal to the number of substations.

    ReDim stSubstation_DFN_guid(1 To iNumber_Substations)
    'Update the size of the array based on the number of substation records.

    For iCounter = 1 To iNumber_Substations
        'Initialize the counter to move from 1 to the number of records
        ' returned by the query.

        stSubstation_DFN_guid(iCounter) = rsSubstations("display_feature_id")
        rsSubstations.MoveNext
        'Load the first substation's display feature GUID into the array.
        'Move to the next substation record.

    Next iCounter
    'Increment the counter.

    rsSubstations.Close
    'After all substations have been loaded into the array, close the recordset.

End If

    'Define and open the listing of features with Multiple Upstreams
    ' with an SQL query.
stMultiple_Upstreams_SQL = "SELECT First(All_Relationships.span_id) AS [span_id Field], " & _
    "Count(All_Relationships.span_id) AS Duplicate_Count, " & _
    "All_Relationships.map_feature_type " & _
    "FROM All_Relationships " & _
    "GROUP BY All_Relationships.span_id, All_Relationships.map_feature_type " & _
    "HAVING (((Count(All_Relationships.span_id)) > 1)) " & _
    "ORDER BY First(All_Relationships.span_id);"

Set mrsMultiple_Upstreams = mdb.OpenRecordset(stMultiple_Upstreams_SQL, dbOpenDynaset)

mrsMultiple_Upstreams.MoveLast
mrsMultiple_Upstreams.MoveFirst
'Enumerate the records by moving to the last record, then
' moving to the first record.

stAll_Relationships_SQL = "SELECT * FROM All_Relationships"
'Redefine the SQL statement to read all records and then open the query.

Set mrsAll_Relationships = mdb.OpenRecordset(stAll_Relationships_SQL, dbOpenDynaset)

mrsAll_Relationships.MoveLast
mrsAll_Relationships.MoveFirst
'Enumerate the records by moving to the last record, then
' moving to the first record.

lBookmarked_Record_Number = 0
'Initialize the var.

```



```

lRecord_Counter = 0                                     'Initialize the var.

                                                         'Define and open the listing of features with multiple downstream features
                                                         ' with an SQL query.
stMultiple_Downstreams_SQL = "SELECT * FROM Multiple_Downstreams ORDER BY upstream_relationship_id"
Set rsMultiple_Downstreams = mdb.OpenRecordset(stMultiple_Downstreams_SQL, dbOpenDynaset)

rsMultiple_Downstreams.MoveLast                         'Enumerate the records by moving to the last record, then
rsMultiple_Downstreams.MoveFirst                       ' moving to the first record.

For iCounter = 1 To iNumber_Substations

    If iScan_Type = 1 Then                             'Depending on the scan_type, either process the first substation, or
                                                         ' the downstream feature specified by the user.

                                                         'Find the first / next substation record in the recordset.
        mrsAll_Relationships.FindFirst "[display_feature_id] = '" & stSubstation_DFN_guid(iCounter) & "'"

    Else

                                                         'Seed value for testing from a set point downstream.
        mrsAll_Relationships.FindFirst "[span_id] = '" & stSpan_ID & "'"

    End If

llProcess_Next_Downstream_Record:

    Do While Not mrsAll_Relationships.EOF               'Only process records until the end of the recordset.

        mvFR_Bookmark = mrsAll_Relationships.Bookmark   'Bookmark the record for ease of locating it later.

        stCurrent_Relationship_ID = mrsAll_Relationships("relationship_id") 'Set the var equal to its Attached Assembly guid.

                                                         'Find the first record where the upstream guid equals the variable.
        rsMultiple_Downstreams.FindFirst "[upstream_relationship_id] = '" & stCurrent_Relationship_ID & "'"

        If rsMultiple_Downstreams.NoMatch = False Then 'Check if current feature has multiple downstream features.

            rsMultiple_Downstreams.Edit                 'Edit the record.
                                                         'Increment the value of the Number Result Processed field.
            rsMultiple_Downstreams("Number_Results_Processed") = rsMultiple_Downstreams("Number_Results_Processed") + 1
            rsMultiple_Downstreams.Update

        End If

        Store_Feature_Information

        If mrsAll_Relationships("switch_position") = "0" Then 'If the switch's status is "0" for "OPEN," then treat the span as a deadend
                                                         ' and jump to the next unprocessed span.

            mrsAll_Relationships.Bookmark = mvFR_Bookmark 'Update the field if this span is a Dead_End_Span.
            mrsAll_Relationships.Edit

```

```
mrsAll_Relationships("Dead_End_Span") = vbYes
mrsAll_Relationships("Span_Processed") = True
mrsAll_Relationships.Update
```

```
GoTo llNext_Unprocessed_Span_After_Branch
```

```
End If
```

```
                                'Find the first record with an upstream guid equal to the current record's AA guid.
mrsAll_Relationships.FindFirst "[upstream_relationship_id] = '" & stCurrent_Relationship_ID & _
                                "' AND [Check_Code] <> 4"
```

```
If mrsAll_Relationships.NoMatch = False Then                                'If the NoMatch property is false, indicating a record has been found, then continue.
```

```
    mrsMultiple_Upstreams.FindFirst "[span_id Field] = '" & mrsAll_Relationships("span_id") & "'" & _
    "AND [map_feature_type] = 'gs_span'"
```

```
If mrsMultiple_Upstreams.NoMatch = False Then                                'This section checks current feature against the table of features
                                ' with multiple upstreams and codes it with a check code of 4
                                ' for manual correction at a later point.
```

```
    mrsAll_Relationships.Edit
    mrsAll_Relationships("Span_Processed") = True
    mrsAll_Relationships("Check_Code") = 4
    mrsAll_Relationships.Update
```

```
    mrsAll_Relationships.FindFirst "[span_id] = '" & mrsAll_Relationships("span_id") & _
    "' AND [upstream_relationship_id] <> '" & stCurrent_Relationship_ID & "'"
```

```
    mrsAll_Relationships.Edit                                'Running this sequence again will ensure that both records of the feature
    mrsAll_Relationships("Span_Processed") = True            ' with multiple upstreams will be marked with a check_code = 4.
    mrsAll_Relationships("Check_Code") = 4
    mrsAll_Relationships.Update
```

```
                                'Seed value after multiple upstreams has been found.
    mrsAll_Relationships.FindFirst "[relationship_id] = '" & mrsAll_Relationships("upstream_relationship_id") & "'"
    GoTo llProcess_Next_Downstream_Record
```

```
End If
```

```
mvFR_Bookmark = mrsAll_Relationships.Bookmark                    'Bookmark this downstream record, as it will become the next feature to be processed.
lBookmarked_Record_Number = mrsAll_Relationships.AbsolutePosition + 1
```

```
llCalculate_Coordinate_Difference:
```

```
    Process_Coordinates
```

```
Else
```

```
    mrsAll_Relationships.Bookmark = mvFR_Bookmark                'Update the field if this span is a Dead_End_Span.
    mrsAll_Relationships.Edit
```

```

    mrsAll_Relationships("Dead_End_Span") = vbYes
    mrsAll_Relationships.Update

llNext_Unprocessed_Span_After_Branch:                'Find the next downstream feature that has not been processed.

    rsMultiple_Downstreams.FindFirst "([Number_Results_Processed] > 0) AND ([Number_Results_Processed] < [Results])"

    If rsMultiple_Downstreams.NoMatch = False Then

        mrsAll_Relationships.FindFirst "[upstream_relationship_id] = '" & rsMultiple_Downstreams("upstream_relationship_id") & "' AND [Span_Processed] = 0"

    Else

        GoTo llNext_Substation

    End If

    mvFR_Bookmark = mrsAll_Relationships.Bookmark                'Bookmark the downstream record for comparison.

                                                                'Find the upstream record for the current record.
    mrsAll_Relationships.FindFirst "[relationship_id] = '" & mrsAll_Relationships("upstream_relationship_id") & "'"

    stCurrent_Relationship_ID = mrsAll_Relationships("relationship_id")

    rsMultiple_Downstreams.Edit
    rsMultiple_Downstreams("Number_Results_Processed") = rsMultiple_Downstreams("Number_Results_Processed") + 1
    rsMultiple_Downstreams.Update

    Store_Feature_Information

    mrsAll_Relationships.Bookmark = mvFR_Bookmark                'Return to the bookmarked record.

    Process_Coordinates

End If

lRecord_Counter = lRecord_Counter + 1

Loop

llNext_Substation:

Next iCounter                'Increment the counter and process the next substation.

mrsAll_Relationships.Close                'Close the recordset.
rsMultiple_Downstreams.Close                'Close the recordset.
mdb.Close                'Close the database connection.
Exit Function                'Return to the Form_frmConnectivity_Assurance code.

ErrorHandler:

```

```
MsgBox Err.Number & " " & Err.Description
Err.Clear
Resume
```

```
'Provide information to the user about the error type and description.
'Clear the current error.
'Resume with the next line.
```

```
End Function
```

```
Public Sub Store_Feature_Information()
```

```
On Error GoTo ErrorHandler
```

```
'Add error handling capabilities.
```

```
Select Case mrsAll_Relationships("map_feature_type")
```

```
'Store either the start or end coordinates of the UPSTREAM FEATURE,
' depending on the type of network feature.
```

```
Case Is = "gs_station"
```

```
'If the type is gs_station, then store the start coordinates.
```

```
mstUpstream_Network_Feature_Type = "gs_station"
mdStart_X_Coordinate = mrsAll_Relationships("Start_X_Coordinate")
mdStart_Y_Coordinate = mrsAll_Relationships("Start_Y_Coordinate")
```

```
Case Is = "gs_span"
```

```
'If the type is gs_span, then store the ending coordinates.
```

```
mstUpstream_Network_Feature_Type = "gs_span"
mdEnd_X_Coordinate = mrsAll_Relationships("End_X_Coordinate")
mdEnd_Y_Coordinate = mrsAll_Relationships("End_Y_Coordinate")
```

```
End Select
```

```
Exit Sub
```

```
ErrorHandler:
```

```
MsgBox Err.Number & " " & Err.Description
Err.Clear
Resume Next
```

```
'Provide information to the user about the error type and description.
'Clear the current error.
'Resume with the next line.
```

```
End Sub
```

```
Public Sub Process_Coordinates()
```

```
On Error GoTo ErrorHandler
```

```
'Add error handling capabilities.
```

```
Select Case mstUpstream_Network_Feature_Type
```

```
'Based on the upstream feature type, determine which coordinates
' to subtract from one another.
```

```
Case Is = "gs_station"
```

```
'Check the absolute difference between the X coordinates.
```

```
msX_Coordinate_Difference = Abs(mdStart_X_Coordinate - mrsAll_Relationships("Start_X_Coordinate"))
```

```
'Check the absolute difference between the Y coordinates.
```

```

msY_Coordinate_Difference = Abs(mdStart_Y_Coordinate - mrsAll_Relationships("Start_Y_Coordinate"))

Case Is = "gs_span"

msX_Coordinate_Difference = Abs(mdEnd_X_Coordinate - mrsAll_Relationships("Start_X_Coordinate"))
msY_Coordinate_Difference = Abs(mdEnd_Y_Coordinate - mrsAll_Relationships("Start_Y_Coordinate"))

End Select

'If the difference between coordinates is less than the tolerance,
' then consider the points equal and continue processing.
If msX_Coordinate_Difference < msComparison_Value And msY_Coordinate_Difference < msComparison_Value Then

mrsAll_Relationships.Edit 'Begin the record editing process.
mrsAll_Relationships("Check_Code") = 1 'Set the check code to 1 - feature endpoints match.
mrsAll_Relationships("Span_Processed") = True 'Set the processed flag to true.
mrsAll_Relationships.Update 'Update the record.

Else

mrsAll_Relationships.Edit 'Begin the record editing process.
mrsAll_Relationships("Check_Code") = 2 'Set the check code to 2 - feature endpoints do not match.
mrsAll_Relationships("Span_Processed") = True 'Set the processed flag to true.
mrsAll_Relationships.Update 'Update the record.

End If

Exit Sub

ErrorHandler:

MsgBox Err.Number & " " & Err.Description 'Provide information to the user about the error type and description.
Err.Clear 'Clear the current error.
Resume 'Resume with the next line.

End Sub

```