REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	**2. REPORT TYPE**	**3. DATES COVERED** *(From - To)*
07-09-2011	Technical Paper	SEPT 2011 - OCT 2011

4. TITLE AND SUBTITLE

Mission Mapping

5a. CONTRACT NUMBER

FA8720-05-C-0002

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Jeffrey Pattillo

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

MIT Lincoln Laboratory
244 Wood Street
Lexington, MA 02420

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

AFLCMC/PZE
20 Schilling Circle, Bldg 1305
Hanscom AFB, MA 01731

10. SPONSOR/MONITOR'S ACRONYM(S)

AFLCMC/PZE

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

DISTRIBUTION STATEMENT A. Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Given a network, our problem is to identify all "missions" within the network. From this we can determine how many missions a node is involved as well as the size of those missions. The overarching purpose for this is to assess network vulnerability. If a node is destroyed it has the potential to derail all missions in which it is involved. Thus nodes involved in large missions or a large number of missions may be critical to protect. There are a few key assumptions about what characteristics a "mission" will exhibit within a network. Missions require communication. Even amongst background chatter or noise, missions should have higher density than an average set of nodes because of this required communication. Two nodes in a mission may not be required to directly communicate. However, we assume most will share some common neighbors, and hence the neighborhoods of nodes involved in a mission should overlap, revealing similar communication patterns.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
U					Zach Sweet
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**	SAR	11	**19b. TELEPHONE NUMBER** *(include area code)*
U	U	U			781-981-5997

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39.18

MS-60140

Mission Mapping

Jeffrey Pattillo

Statement of the Problem And Assumptions

Given a network, our problem is to identify all "missions" within the network. From this we can determine how many missions a node is involved as well as the size of those missions. The overarching purpose for this is to assess network vulnerability. If a node is destroyed it has the potential to derail all missions in which it is involved. Thus nodes involved in large missions or a large number of missions may be critical to protect.

There are a few key assumptions about what characteristics a "mission" will exhibit within a network. Missions require communication. Even amongst background chatter or noise, missions should have higher density than an average set of nodes because of this required communication. Two nodes in a mission may not be required to directly communicate. However, we assume most will share some common neighbors, and hence the neighborhoods of nodes involved in a mission should overlap, revealing similar communication patterns.

Discussion of potential grouping methods

Groups in a network often exhibit a few basic characteristics. They tend to have a high volume of communication, much of which is direct, and tend to be robust, not severed by the destruction of a few nodes. An ideal group in a network is a clique. A clique is defined to be a set of nodes which all directly communicate. They exhibit the highest possible volume of communication and this communication cannot be destroyed without removing every node from the network. While exhibiting nice properties, the structure of cliques often do not match groups in real world networks, because members of the same group often do not have direct communication. Thus many alternatives to cliques have been proposed for grouping.

One alternative structure for identifying groups is a community. Communities are a set of nodes that may not have perfect communication, but exhibit more communication within the group than to the outside. Some nice features about communities is that they make no assumptions whatsoever about the local structure of a mission and will produce a set of nodes that will most certainly be dense. However, communities by very definition tend to not overlap. If a node x in community A is also in a community B, then A necessarily has a significant number of connections to the outside since x necessarily has a significant number of connections to B. Because communities do not overlap, they are not ideal for our purposes since our whole goal of identifying key nodes via mission identification is based off the assumption that missions overlap.

Some other alternative structures for identifying groups are known as clique relaxations. Clique relaxations take a defining feature of a clique and preserve it in a relaxed form. For instance, a clique can be defined as a set of nodes such that every pair of nodes is distance 1 apart. We can relax this

This work is sponsored by the Department of the Air Force under Air Force contract FA8721-05-C-0002.Opinions, interpretations, conclusions and recommendations are those of the author and are not necessarily endorsed by the United States Government.

definition to create an object called a k-clique where every pair of nodes is at most distance k apart. There are six basic clique relaxations, each of which chooses a different defining property of a clique and relaxes it. The k-clique, as just described, requires all vertices to be at a distance of k or less. The k-club clique relaxation requires vertices to be distance at most k on the induced subgraph. The gamma-quasi-clique relaxation requires the set of vertices to have density gamma as an induced subgraph and the k-connected subgraph requires every pair of vertices to have k vertex independent paths between them. The k-core relaxation requires every vertex to have degree at least k in the induced subgraph while the k-plex relaxation requires each vertex to be missing edges to at most k other vertices in the induced subgraph. Note that we will refer to k as the "missing degree" of the vertex in the case of k-plex.

Each clique relaxation has different properties that make it the ideal choice for grouping in a specific application. None, however, match very well our description of a mission in the presence of noise. If we chose to group using 2-cliques and 2-clubs, since most communication is distance 2 or less in a mission, we would likely end up combining several missions together. A single shared neighbor allows for two nodes to be grouped that should not be and so distance alone fails to separate nodes enough. The k-core and k-plex relaxations also fail to separate nodes satisfactorily because even if we knew the minimum degree or "missing" degree of a mission, there is no guaranteed separation between the degree of the least involved nodes in a mission and outsiders, since missions are defined on more of a global scale. The k-core will likely not separate any nodes at all if k is the minimum degree of a node in any mission, and the k-plex relaxation will likely not separate into the right pieces. Searching for k-connected subgraphs seems to be very unrelated to mission identification, since paths can have any length in a k-connected subgraph and we care only about paths of length 2 or less. The quasi-clique relaxation does have potential to identify missions but fails for a different reason than the others. We know missions have distance 2. We have no idea, however, what density a mission must exhibit a priori. We can continually lower density until gradually sets of nodes surface. We do not know, however, at what grade this density should be lowered and when to stop. There may be a very fine line between densities that our missions exhibit and groups based upon chatter alone, and if any mission exists between "introverts," we will not be able to detect it at all without allowing groups based on chatter to infiltrate our list of missions.

In the presence of no noise, several clique relaxations might be adequate because there likely will be enough natural separation between missions that the appropriate parameters can be discovered. Even though we do not know in advance the appropriate parameters to choose, we can detect when our groups remain roughly fixed as we continue to relax the structure we are searching. We quit relaxing parameters when the groups start combining rapidly and declare the groups based upon the preceding parameters. Thus in environments such that no noise exists, or in environments where we can scrub the data to have a minimal amount of noise, these methods would likely be suitable.

Instead of using a basic clique relaxation, we can blend clique relaxations to form groups we will refer to as an r-robust k-clique and an r-robust k-club. Both these clique relaxations have high connectivity and low distance, making them a hybrid of k-connected subgraphs and k-cliques. More specifically, an r-robust k-clique is a set of nodes such that every pair has distance k, and every pair remains distance k if

This work is sponsored by the Department of the Air Force under Air Force contract FA8721-05-C-0002.Opinions, interpretations, conclusions and recommendations are those of the author and are not necessarily endorsed by the United States Government.

less than r vertices are removed. An r-robust k-club has the same definition except the distance and connectivity requirements apply to the induced subgraph. We concentrate specifically on the r-robust k-clique on this paper for reasons we explain momentarily. We chose to call it an r-robust k-clique instead of a r-connected k-clique because the term r-connected k-clique is ambiguous as to whether the set remains a k-clique under the removal of less than r nodes. We do not want to identify a set of nodes with only 1 path of length k between every pair of vertices and r-1 paths of longer length, which might be considered an r-connected k-clique. It is less ambiguous that only paths of length k are counted toward connectivity with the title r-robust k-clique.

Both the r-robust k-clique and r-robust k-club seem much more suited for identifying missions as we defined them. We know from our definition of missions that the parameter k can be fixed to 2 because we only care about direct and indirect communication with a single intermediary. Further, while the parameter r is unknown, we know to decrement it in whole numbers and thus do not run into some of the degradation questions we encountered with quasi-cliques. We may encounter similar problems to k-plex and k-core if we lower the connectivity enough to accommodate even the least connected members of the group, unable to separate outsiders at that point. However, with connectivity there is a clear starting point (connectivity 1) and clear ending point (maximum connectivity) and so we can enumerate all potential groups without ambiguity as to when to quit. If we have some notion of the minimum connectivity of a group, we can use this to limit our candidates. This is not the case with k-plex or gamma-quasi-clique, where it is unclear what k or gamma value at which to stop. With r-robust k-cliques and r-robust k-clubs, we ultimately produce a list that certainly separates missions, when connectivity is high enough, and with parameters that are much easier to guide.

The choice between r-robust k-cliques and r-robust k-clubs should likely depend on the presence of noise and the time available to solve the problem. The r-robust k-clique is much more noise tolerant and much easier to identify computationally, which is why we concentrated on it in this paper. It has the disadvantage that neighbors used to connect two nodes indirectly may not be included in the group, and hence the missions may be more likely to come out in pieces. The r-robust k-club requires the r paths of length k between any pair of nodes exist in the induced subgraph and thus the necessary neighbors will always be included in the group. This may not equate to missions coming out in fewer pieces, however, because these requirements are much more strict and thus it is much more likely only portions of a group satisfy the requirements. Further, they are less noise resistant and they are computationally much more difficult to identify, as we will next discuss.

An r-robust k-club is much more difficult to identify than an r-robust k-clique because it does not exhibit pseudo-heredity. Heredity and pseudo-heredity describe a property p and how it behaves in a dynamic graph as the graph changes. A property p is hereditary if any subgraph S with property p retains property p upon the removal of a node. A property p is pseudo-heriditary if any subset S' of a subgraph S with property p also has property p. To exhibit heredity, the property must remain upon removal of a node from S whereas to exhibit pseudo-heridity, the property must only remain when a nodeis outlawed from S but remains in the graph. Properties that do not focus on the induced subgraph may demonstrate pseudo-heredity without heredity. While heredity is more ideal since the group is more

This work is sponsored by the Department of the Air Force under Air Force contract FA8721-05-C-0002.Opinions, interpretations, conclusions and recommendations are those of the author and are not necessarily endorsed by the United States Government.

robust, both are significant computationally. Having heredity or pseudo-heredity means the set can be built up one element at a time while always having property p, since by definition the set can be torn down one element at a time and retain property p. Maximal sets containing a hereditary or pseudo-hereditary property can be identified by the inability to add any single element without having to check subsets of larger size, which often significantly increases the speed at which a problem can be solved. An r-robust k-clique has pseudo-heredity. Every subgraph of an r-robust k-clique will be an r-robust k-clique because the r paths of distance k need only exist in G and not in an induced subgraph. In addition to making maximal sets easier to identify, not focusing on the induced subgraph allows us to not have to be constantly recalculating distance and connectivity as we add and subtract nodes from our subset. The distance and connectivity are independent of the subgraph induced by the subset. We do not have this same luxury with an r-robust k-club and hence the r-robust k-clique is much more efficient at identifying groups computationally.

To address the difference in noise resistance between the r-robust k-club and r-robust k-clique, it will be easiest to consider an example. We were given the task of grouping HR and travel, two missions with a very defined purpose, based off connectivity habits to the internet. The graph we built was bipartite, with websites in one division and people in the other and edges representing the websites each person connected to. The graph was full of noise because browsing sites other than for work was not prohibited. Consider what types of subgraphs the r-robust 2-club and r-robust 2-clique specifically will identify. If direct edges are only counted as a single path between nodes (even though the two nodes by definition have infinite connectivity), an r-robust 2-club would not exist for r >1. No edge exists within the partitions, and no vertex has more than 1 path of length 2 or less to the opposite partition. If a direct path between two nodes is counted as infinite connectivity, an r-robust 2-club would produce a complete bipartite graph in our example. If any two vertices in opposite partitions are missing an edge, there is no path of length less than 3 between them (using one intermediary on each side), much less r such paths. It is not likely that everyone in HR or everyone in travel connects to a single mission related website, much less r websites, and hence the complete bipartite graph identified as an r-robust 2-club would only identify a piece of the mission. Consider instead the r-robust 2-clique. The neighbors need not be included and hence it does not matter that only 1 path of length 2 exists between websites and each people. Any set such that every two people share r websites in common, even if the r websites that person A and person B share differ drastically from the r-websites person A and person C share, can be grouped together. It is likely a benefit that websites not be included in the HR and travel groups, but even if desired, the r-robust 2-club produces HR and travel in pieces as well. Both methods have problems with producing missions in pieces, but the r-robust 2-clique is more forgiving of variations in browsing than the r-robust 2-club.

This example highlights another advantage that r-robust k-cliques have over k-plex and gamma-quasi-clique. In a bipartite setting such as our browsing example, where members of the same mission necessarily do not communicate, k-plex and quasi-clique punish groups for having larger size. The more members that exist within a partition, the more edges the group will necessarily be missing. Under no circumstances will the same value of k or of gamma work to identify all groups. In the presence of noise,

This work is sponsored by the Department of the Air Force under Air Force contract FA8721-05-C-0002.Opinions, interpretations, conclusions and recommendations are those of the author and are not necessarily endorsed by the United States Government.

it is likely outsiders will be included with the smallest groups because we increment k and decrement gamma far beyond what is necessary to identify the smallest groups. The r-robust 2-clique does not punish groups for having different sizes. If the connections via intermediaries exist, it does not matter that members of a mission are missing edges whatsoever.

A structure similar to the r-robust k-club and r-robust k-clique has had success in grouping websites based on content (cite Terveen). Using a set of n related websites as a seed, the authors grouped websites that linked to at least r of the n sites. If we have knowledge of good seeds upon which to build groups, we should most certainly use it. Returning to our HR and travel example, if we are not content agnostic then we could choose the travel websites and HR websites and likely do a much better job grouping. However, when this knowledge is not available, the r-robust k-club and r-robust k-clique do not require it. If we fed in the n websites used by the r-robust k-clique or r-robust k-club into Terveen's method as the seed, it would likely perform much more poorly. Terveen's method requires vertices connect to only r total websites of the n total, and not share r in common with every other member of the group, and so outsiders are much more likely to be included that connect to r of the least utilized nodes. Thus while Terveen's method is much more effective when we are not agnostic as to the content of the nodes, the r-robust 2-clique seems to be the most effective and resistant to noise when we don't have this knowledge and is thus the focus of this paper.

There is one alternative for grouping that is not based off of structure of any kind. The method is called the stochastic mixed membership block model. Based off of an assumption about the number of groups X in the model, and assuming group membership is what is causing the communication, it finds the most likely X groups to explain the communication, attributing missing edges to noise. It allows for members to be involved in multiple groups and thus does not run into the perils that community encountered. Like clique relaxations, it is likely to be very effective for a graph with no noise. However, in graphs with noise, the noise will likely force extra groups to be identified that will be interspersed amongst the true missions. The grouping mechanism we create in this paper can utilize the knowledge from both identifying r-robust k-cliques, and from the mixed membership stochastic block model. We chose to use r-robust k-cliques to reduce the list of possible missions and then the mixed membership stochastic block model to help score the groups. There is no reason, however, the groups cannot be used in opposite order.

"Failed" Implementations

As to be expected based off the discussion in the previous section, k-cliques, k-clubs, k-cores, quasi-cliques, k-plexes, and k-connected subgraphs failed to be effective at identifying groups in the presence of noise. While the r-robust k-clique proved to be effective, there were several implementations of it that failed as well. Ideally we would report a final list containing only missions and a few false positives by aggregating overlapping groups and eliminating groups. However, when we tried to do so, we sometimes lost all information, ending up with a single giant group. This happened even when we put safeguards in such as limitations to how many groups can be combined at once. It is certainly problematic to run code for any length of time and get no information back. Thus even though a nice

This work is sponsored by the Department of the Air Force under Air Force contract FA8721-05-C-0002.Opinions, interpretations, conclusions and recommendations are those of the author and are not necessarily endorsed by the United States Government.

and tidy list is ideal, we chose to report every possible group in a hierarchy, where the more likely candidates were near the top. While the list is much longer and does not allow groups that overlap to be aggregated, we chose it as our final implementation because it is guaranteed to return information.

"Successful" Implementations

The user gets to specify the minimum connectivity of any group that will be in the final report (often 1), the maximum distance that is accepted as mission communication (typically 2), an estimate for the smallest group size, whether or not noise exists in the data, whether or not to he wants to use training data, and whether or not he wants to check other alternatives to see how they perform at grouping. From this input, the basic flow of our code is as follows. First we identify all r-robust k-cliques to trim the list of possible groups from all possible subsets into something maneagable. After finishing, if the user has inputted training data to help identify what a group looks like, we use the data to weight tests that we use as a scoring system to separate r-robust k-cliques that are missions from those that are not. If no training data exists, all tests are weighted equally because all tests are measures of characteristics that typically appear in groups. Upon receiving the weights of all tests, the program looks at the real data set, identifies all r-robust k-cliques, and then orders the list by the scores they receive based of the tests and their weights. This list is then re-organized to list groups in levels, where there is no containment of one group in another for groups in the same level. This is to prevent the list from being inundated with lots of groups receiving the same score that involve the same set of vertices. In re-ordering the list, we ensure no group with too low a score jumps too far simply because it looks "different" from all other groups. We also ensure if a lot of groups that overlap significantly appear not far from a group that contains them all, then only one such group appears in that level and the rest are moved down a level so as to not inundate the list with groups that look like they likely should be collected. The final list we report does not necessarily contain all the groups with the highest score at the top, but rather a set of representatives of the fundamentally different groups that appear with a high score near the top.

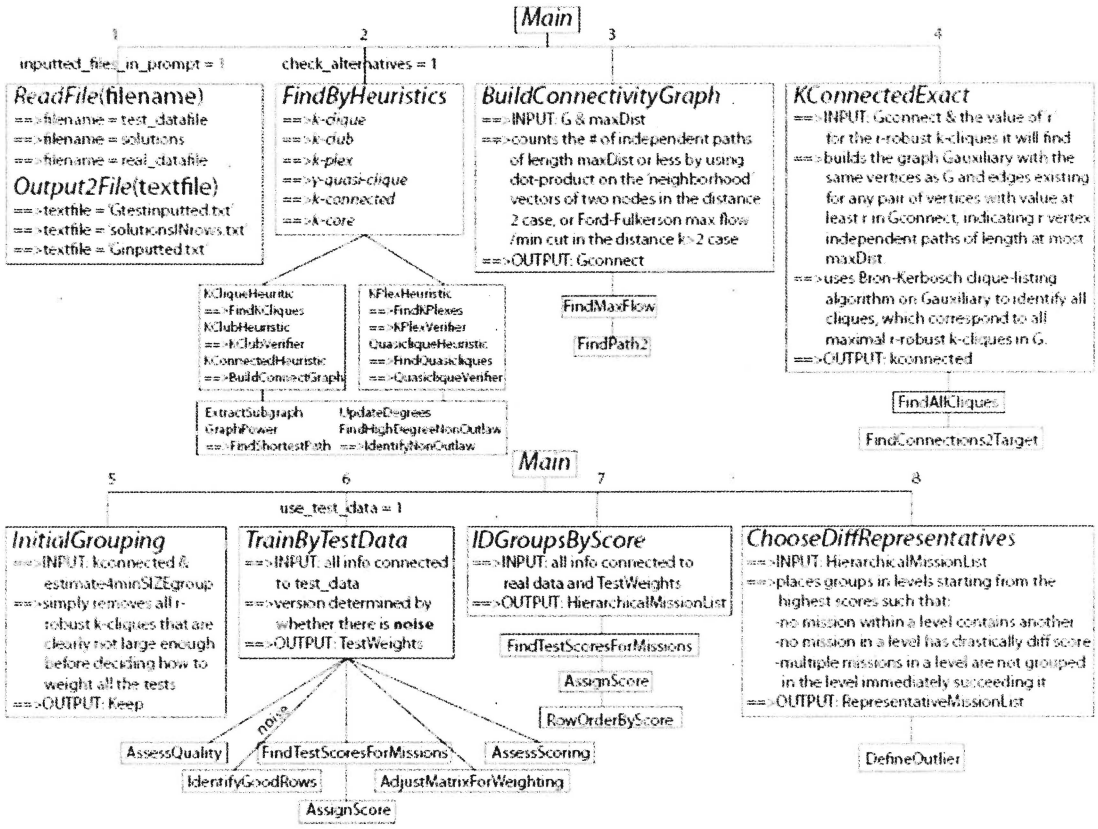
One implementation choice that we made turned out to be very important. Upon receiving the list of r-robust k-cliques, we prune the list of any groups that are not at least half as big as the estimate for the minimum group size given by the user. Such groups we likely do not want report on our final list, and if we have training data, can severely throw off our scores for separating groups. Small subsets of missions and of non-missions are likely much more difficult to differentiate and can mess up our scoring system we obtain from the training data and thus it is crucial to have a minimum size we send in to help weigh our tests.

Note our implementation does not allow for groups to be aggregated at any point, even if they overlap significantly. Our assumption is that if two groups should be aggregated, they will appear together somewhere on the list with lower connectivity. If they do not, they are likely separate missions and should not be aggregated. The only danger in this is that at lower connectivity, outside nodes might infiltrate the group. However, if a low enough connectivity is required to combine two groups such that outsiders obtain entrance, we choose to accept this rather than completely flood our list with groups,

This work is sponsored by the Department of the Air Force under Air Force contract FA8721-05-C-0002.Opinions, interpretations, conclusions and recommendations are those of the author and are not necessarily endorsed by the United States Government.

because there is little reason to believe the two overlapping groups should be fused without the outside node.

Details



There are 11 parameters that must be given to *Main* to specify how it performs its 8 basic functions. The code can be run by typing something like

Gconnect,kconnected,Keep,HierarchicalMissionList,RepresentativeMissionList,TranslatedMissions,stat] = Main(2,1,5,0,1,1,'test_data.txt','solutions.txt','real_data.txt',1,1);

where the parameters in order are: 1) *maxDist,* 2) *minConnect,* 3) *estimate4minSIZEgroup,* 4) *noise,* 5) *inputted_files_in_prompt,* 6) *use_test_data,* 7) *test_datafile,* 8) *solutions,* 9) *real_datafile,* 10) *columns_labeled_in_graph,* and 11) *check_alternatives*. While this is the order they are specified to *Main*, we will discuss them in the order in which they are relevant to the program. The first parameter to pay attention to is *use_test_*data. If no test data is to be used, give this a value of 0, and consequently, the values for *test_datafile* and *solutions* will be ignored. If test data is to be used, give this parameter a value of 1. If the data (whether this includes test data or not) is to be specified using text files with information passed in by the prompt, set the parameter *inputted_files_in_prompt* equal to 1. Otherwise set it equal to 0 and the program will have to be manually changed to import the data. This parameter also controls whether or not the parameters *test_datafile,* and *solutions* are used or

This work is sponsored by the Department of the Air Force under Air Force contract FA8721-05-C-0002.Opinions, interpretations, conclusions and recommendations are those of the author and are not necessarily endorsed by the United States Government.

ignored, as well as the parameters *real_datafile* and *columns_labeled_in_graph*. The data files must be typed with the appropriate extension between apostrophe's in the prompt if they are to be passed in this way. Both *test_datafile* and *real_datafile* should be **adjacency matrices** and should have every row labeled with the "name" for the row. Do not use numeric characters to name/label the rows because that will confuse the function *ReadFile*. If the columns in the graph are also labeled by a row of "names" at the top of the input files, specify *columns_labeled_in_graph* to be 1. Otherwise make it 0. The *solutions* parameter should have the groups that exist in the training data as its rows, with each row having a label naming the group. Since this is not an adjacency matrix, the columns should not have any labels.

The remaining parameters deal with how it handles the input data once it has been gathered. If the parameter *check_alternatives* is set to 1, then the program identifies samples of six first order clique relaxations on the real data. The parameters for each clique relaxation have to be manually changed in the program by the user since this is not the focus of the program. These are simply provided in case there is interest if perhaps a different clique relaxation might work better. The parameters *maxDist* and *minConnect* are used to specify the values of r and k before identifying all r-robust k-cliques. In fact, k = *maxDist* and *minConnect* represents the lowest value for which we will let r range, with the maximum possible connectivity between any two vertices in the graph being the upper bound for r. The parameter *estimate4minSIZEgroup* is used by the function *InitialGrouping* to remove r-robust k-cliques that are clearly too small to be groups. At present it gets rid of all r-robust k-cliques that are not at least 30% of the size of *estimate4mizSIZEgroup*, being very conservative so as to not lose any groups. This helps out our scoring system significantly as mentioned previously. The final parameter, *noise*, is a Boolean that controls what we do with the inputted training data. If the training data has significant non-mission related communication (noise) then this parameter should be 1, otherwise 0. We are more cautious with our scoring system when noise is present, since real groups are most likely less pronounced, and so it is not allowed to bias the scores to "punish" groups quite as harshly since it may in the process punish good groups. False positives are accepted as inevitable in the presence of noise and the scoring system concentrates more on helping the most ideal groups surface near the top of our list.

The 8 basic functions, along with most of their basic sub-functions, are listed in the figure. The basic flow is as follows. The data is gathered either from the prompt or by manually adjusted code in *Main* and then alternatives are checked, if specified by the user. The code then transitions to building *Gconnect*, which represents how many vertex independent paths of length *maxDist* or less exist between every pair of vertices in G. The number of paths can be found by adapting a max-flow min-cut algorithm such as Ford Fulkerson to only accept paths of length *maxDist* or less, though this process can be circumvented when maxDist = 2 specifically. From this we cycle through values of r from *minConnect* to the maximum connectivity level and find all r-robust k-cliques. We do this by building an auxiliary graph called *Gauxiliary* that has the same set of nodes as G and edges that exist between vertices that have value at least r in Gconnect. Such vertices have r vertex independent paths in G, and so any clique in Gauxiliary will be an r-robust k-clique in G, since r-robust k-cliques consider **distance** in G and NOT in the induced subgraph (which would be measured by **diameter**). A form of the Bron-Kerbosch algorithm is used for listing all maximal cliques in Gauxiliary. Once the list of maximal r-robust k-cliques is obtained, it is trimmed to get rid of r-robust k-cliques that are too small by the function *InitialGrouping*. The above procedure of building Gconnect, multiple copies of Gauxiliary, and trimming is always applied to the real data, but it is also applied to test_data when it is given. If test data is given, we use it to find *TestWeights*, which helps to separate ideal groups from poor ones. If test data is not given, all tests are weighted equally. The real data is then assigned scores using *TestWeights* by the function

This work is sponsored by the Department of the Air Force under Air Force contract FA8721-05-C-0002.Opinions, interpretations, conclusions and recommendations are those of the author and are not necessarily endorsed by the United States Government.

IDGroupsByScore. From there we list from the top scores in an intelligent way using the function *ChooseDifferentRepresentatives*.

The program outputs the following values: *Gconnect* (as described above), *kconnected* (which is all r-robust k-cliques for all values of r), *Keep* (which is the trimmed list of r-robust k-cliques), *HierarchichalMissionList* (which is the missions listed in order of score alone), *RepresentativeMissionList* (which is the missions listed in an intelligent order from the top scores), *TranslatedMissions* (which translates *RepresentativeMissionList* so that its groups, which are in rows, are written using the names of the members of the adjacency matrix), and *stat* (which contains statistics of how many members of each actual group the rows/missions of *RepresentativeMissionList* contain). Note that *stat* will be nonsense if you are not using training data and real data with the same groups. It simply is an assessment tool when we know what groups we ultimately want to distinguish using our scoring system, but when the groups are unknown in advance, it is nonsense to count how many members of each group our final list contained.

The main place a user may need to edit the code is the function *AssignScore*. Currently there are tests for **density, modularity, minimum degree**, and **uniformity**. These functions are not included in the chart because they can be added and removed as needed. When training data exists, it should not matter how many tests are used in *AssignScore*, assuming the scoring system is "smart", because it can give unimportant tests a score of zero. However, when training data does not exist it may be necessary to remove some tests intelligently because all remaining tests will be given equal weight by the program. In adding a new function to *AssingScore*, it must be tacked on to the back end of the array *scores4group*. The program assumes connectivity constitutes the first scores in the array, which it may have to adjust because the connectivity of test data may not match the real data. If any test is placed earlier in *scores4group* than connectivity, it will confuse the program. If a new test is added, the number of tests should be adjusted using *num_tests* at the **first line** of code in *Main*. This counts how many tests besides connectivity itself exist in *AssignScore* and tells constrained least squares how many coefficients it needs to identify. Variables corresponding to the new tests can also be included in *TrainByData* for printing purposes only but are not necessary unless you desire to know the weights that were ultimately chosen.

One other function that has a few design decisions that might be changed is *TrainByData*. At present it does constrained least squares on the matrix containing test scores for each mission and the ideal score we would like it to achieve. There may be non-linear methods that are superior to constrained least squares and in order to test one, replace the appropriate code in *TrainByData*. If the numeric choices for the ideal scores themselves need to be changed in order to work with some new method, these can be changed in *IdentifyGoodRows* or *IdentifyGoodRowsNoise*. Since the code uses constrained least squares at present and there are often significantly more false missions than true ones, the code is designed to have the option of padding the matrix of mission test scores with the best rows many times. This makes least squares as concerned with getting these rows correct as with getting the overwhelming number of false missions the correct score. It is inconclusive whether this ultimately helps or hurts the final scores that result so it is something that might be tested more. The parameters *pad_matrix* and *separation_method* to the function *TrainByData* inform it to use either constrained or unconstrained least squares and whether or not to pad the matrix with good scores. These parameters provide the simplest way to add a non-linear separation method and to decide whether or not to pad the matrix.

Conclusion and Future Work

This work is sponsored by the Department of the Air Force under Air Force contract FA8721-05-C-0002.Opinions, interpretations, conclusions and recommendations are those of the author and are not necessarily endorsed by the United States Government.

In this paper we outlined a method for grouping nodes into missions by communication habits so as to ascertain the value of each node within a network. While the best choice between community, a clique relaxation, and the mixed membership stochastic block model may vary from setting to setting, we have created a method for grouping that is likely adaptable to any setting. Our implementation can take the results of any grouping method to help determine a score, and thus is extremely versatile. While no r-robust k-clique may match the exact group desired that perhaps another method might find, it is hard to imagine a setting where the results would not be similar under the appropriate choice of r and k.

The most useful aspect of the way we ultimately implemented the program is indeed its versatility. It should be easy to adapt it to recognize groups in any setting where there is a defining characteristic to their structure. Every r-robuts k-clique Q of the graph is assigned a set of scores. The scores are all between 0 and 1 and represent some sort of measurement that might be used to delineate groups, such as density, minimum degree, etc. The higher the score between 0 and 1, the more Q appears to have a characteristic typically associated with groups. In order to recognize groups in settings with a defining characteristic, a way to measure the characterstic must be coded up and the measurement associated with a value between 0 and 1. It can then be incorporated into our scoring system and forced by the user, either by hand or through training data, to be given higher weight than other tests. Adding a new test to the scoring system will not compromise the effectiveness of the program on other networks, assuming there is training data, because it will be given a low weight when the test is not applicable for separating groups. An area for future work is to identify many more tests which can help to delineate between groups in a variety of settings.

Even though we want to group nodes based only on shared communication, being agnostic as to the content of nodes, we do have extra information that could be of service that we are not currently utilizing. One such piece of information is the length of communication. At present we build a network where two nodes are linked if they share any communication. Often non-mission communication takes place, which we will call noise. It is obviously more difficult to group nodes in the presence of noise than when we strictly see mission communication. We might be able to use knowledge of the length of communication to help "scrub" the data so that less noise exists, thereby creating a graph where the missions are more pronounced. If we know the subset of nodes that need to be grouped, we can further clean the data by dismissing nodes connected to all or even most of our subset as noise. There is much more to be explored with scrubbing data.

At present we typically assume we want d=2 when we create a list of k-connected d-cliques that are potential missions. When a mission has hierarchical communication, this may not be true and we may end up with the mission in several pieces. We can still count how many missions a node is involved in when a mission is divided into several pieces, and so this will not defeat that purpose. Having a mission divided into several pieces might even be of assistance to deciding the criticality of the node within the mission. For instance, if we are able to group the pieces together, having the mission divided into pieces will help reveal what nodes are at the center of all communication. At present we do not try to group the pieces and so our code, with our assumption that communication is at distance 2 within a mission, will not yield the full size of a mission. It might not be difficult to put the pieces together and could be a

This work is sponsored by the Department of the Air Force under Air Force contract FA8721-05-C-0002.Opinions, interpretations, conclusions and recommendations are those of the author and are not necessarily endorsed by the United States Government.

signficant way to improve the code. Another alternative to combining pieces would be to use d>2. However, as d increases, the structures that qualify as d-cliques drastically increase. For instance, there is a popular notion that the world as a social network is a 6-clique. Games such as "6 degrees to Kevin Bacon" have even been invented based on this assumption. We could likely increase the connectivity requirement to compromise for an increase in d, but it is not obvious the extent to which connectivity should be increased and this is an area with need for more research.

This work is sponsored by the Department of the Air Force under Air Force contract FA8721-05-C-0002.Opinions, interpretations, conclusions and recommendations are those of the author and are not necessarily endorsed by the United States Government.