Graph Partitioning of Transportation Networks under Disruption

A Thesis

In

The Department

Of

Concordia Institute for Information Systems Engineering (CIISE)

Submitted in Partial Fulfillment of the Requirements

for the Degree of Master of Applied Science (Quality Systems Engineering) at

Concordia University

Montreal, Quebec, Canada

February 2015

CONCORDIA UNIVERSITY
School of Graduate Studies

This is to certify that the thesis prepared

By: **GhavidelSyooki, Mona**

Entitled: ***Graph Partitioning of Transportation Networks under Disruption***

and submitted in partial fulfillment of the requirements for the degree of

Master of Applied Science (Quality Systems Engineering)

Complies with the regulations of the University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

Dr. Chun Wang Chair

Dr. Mourad Debbabi Examiner

Dr. Mingyuan Chen Examiner

Dr. Anjali Awasthi Supervisor

Approved by Chair of Department or Graduate Program Director

Dean of Faculty Dr.Rachida Dssouli

Date February 2015

Abstract:

Graph Partitioning of Transportation Networks under Disruption

Ghavidelsyooki, Mona

Concordia University, 2015

This research is concerned with providing a solution capable of treating network complexity and scalability effectively so that it overcomes administrative, environmental and technique boundaries. One good approach dealing with this matter is applying graph partitioning techniques. Graph partitioning is an optimization problem with the aim of dividing a large geographical network into manageable size districts called sub-networks with less complexity in favor of balancing the workload and minimizing the communication among them, with the aim of maximizing their independency as much as possible. Over the past decades various models have been developed in such a way to satisfy a multi-objective problem such as delivery time and managerial cost. In real life, due to inevitable changes during network's lifetime, it is vital to offer survivability and resilience in the existence of network failure and disruption. Further, it is essential to maintain functionality in critical facilities and high priority connections in the time of crisis. This paper suggests four partitioning techniques namely "Hierarchical recursive progression1$^+$ "(HRP1$^+$) and "Hierarchical recursive progression2$^+$ "(HRP2$^+$) and their extensions called "HRP1$^+$control" and "HRP2$^+$control" to solve the scalability as well as complexity of a network. For this matter, the initial balanced partition is produced on a predefined network. Furthermore two different approaches namely "complete failure update

“and “partial failure update” are proposed and demonstrated in the occurrence of network disruption.

In sum, the three main objectives of this thesis are as follows:

1. Modeling disruption on logistics networks
2. Assuring and strengthen connectivity in the disrupted network for routing purposes
3. Developing partitioning approaches in favor of generating roughly equal sized and balanced partitions in the disrupted network.

Dedications and Acknowledgments

This research study is dedicated to my family for their unconditional support, effort, and to my friend Mohamad Soltani for his help and guidance. I am forever in debt.

Acknowledgement goes to Concordia University's society, academic and non-academic, as well as the City of Montreal.

Table of contents

Chapter 1:
Introduction

1.1. Background

The evolutionary change in the nature of supply chain and logistics networks is the aftermath of network complexity derived from technology development and the emergence of globalization. Supply chain network is a complex combination of organs, recourses and activities interacting to provide a service/product from supplier to consumer. A number of activities are involved in logistics and supply chain management such as planning and optimization of sourcing/procurement, facility location, inventory management, warehousing, distribution, information integration etc. According to the Supply Chain Council[1] a non-profit, global corporation, "Logistics management can be considered as the process of coordinating movement of materials and its related information from its starting point to its endpoint in favor of meeting customer requirements and expectations".

The ongoing growth of this world-wide integration has forced fundamental changes to the network's infrastructure, its inherited characteristics and underlying properties that have caused great transformation in system's functionality. Today, supply chain and logistics networks are struggling with challenges such as unpredictability, uncertainty in a non-linear way more than ever due to the enlargement of markets from national to international arena.

A model capable of managing and optimizing complex systems calls for a wide fundamental change in the perspective of the company. Not only logisticians and operators need to level up their envision with a broader insight of how a decision in one section can have a great impact

[1] The Supply Chain Council's website is located at http://www.supply-chain.org/public/home.asp

on the whole system but also must acquire sufficient technological and analytical competence in order to firstly fulfill communication requirements and secondly understand and interpret data such that the system evaluation and improvement can ultimately pursuit its effective functionality. As a result network designers must foresee possible problematic scenarios and fortify their models with underlying characteristics such as resilience and reliability due to network survivability. Furthermore since the future effect of a local change in the global behavior of a network is unpredictable, it is vital for supply chain planners to take account for risks and unexpected events in order to mitigate not only the severity of an incident but also its unknown yet possible widespread impact by prompt reaction; as late detected errors can propagate rapidly and cause extreme damages in the system-wide structure and performance. It is well known that each model embeds a robust and at the same time fragile template. This means that various types of stochastic and deterministic mathematical models for formulating a network are able to address and satisfy some aspects yet it is most likely to fail sustaining other aspects. Therefore, the system must have the tolerance for further adjustments in the case of probable disruptive events that haven't been seen prior to occurrence. Hence, redesigning actions with the aim of improving the structure of the network to meet key player's expectations in the supply chain is necessary.

(A Framework for Computing Topological Network Robustness 2010) In conclusion, survivability is practiced in three areas. Preventive actions; that aim to improve network survivability by means of reliability enhancement such as upgrading equipment and machines as well as equipping critical zones with fault-tolerant and self-healing hardware. *Network design*; that plan on catering extra capacity with diverse range of paths such as multi-homing solution in order to reduce the impact of node /link failure in the system. Finally, an *end to end*

traffic management ensuring fast recovery from attacks meaning the ability to adapt quickly and perform business-usual activities in the time of crisis. Moreover utilizing the remaining capacity and other available options in order to avoid traffic jam and pursue traffic flow. With all this in mind, the scope of this thesis is to provide a solution that could solve survivability problem by utilizing the advantages of the latter mechanism. We would like to develop a solution that is to be considered a corrective action which accelerates system recovery after disruption by means of a resourceful and robust structure.

We present our problem through a set of nodes and links called graph. That is because primarily the natural framework treating complexity lies within graph theory science. Secondly studying large scale networks of irregular structure that are dynamically evolving in time through depicted graphs would lead to a better understanding of its evolutionary mechanisms and functional behavior. As a result many scientific areas of study have called attention to employ graphs and their applications in their frame work, mostly where flow of people, goods, information and funds also known as network flow models is to be closely investigated. The other important factor that led us towards proceeding our research in the field of graph optimization problem is that globally interconnected systems are a combination of many interacting parts that run on their own internal structure, performing a specific behavior or function. Network segregation proves to be very practical when working with large scale networks, where clustering components into logical units helps enhancing performance in areas such as pattern configuration, data management, data storage and virtual memory improvement. In addition our solution is known to be a heuristic-based approach due to the fact that applying ILP- based models (integer linear programming) would require computational calculation. Hence for networks of large scale, consisting of great number of nodes and links, an ILP- based

model would become very complicated and time consuming while a heuristic approach can obtain near optimal solutions within an acceptable time frame.

1.2. Research objectives

Graph partitioning is an optimization problem that divides a network into subsets of manageable size with less complexity and minimum interaction between them, with the aim of maximizing their independency as much as possible. This in turn helps improving network performance and functionality along with handling restriction and privacy issues relating to transforming information.

With the assumption that the studied network is in general a robust network we would like to develop a partitioning-based approach that not only improves the management of supply chain and logistics but also is able to account for disruption in the face of inescapable, unpredicted incidents. Nevertheless it is essential to maintain functionality in critical facilities and high priority connections in the time of crisis.

In summary, the three main objectives of this thesis are as follows:

4. Model disruption on logistics networks
5. Assure and strengthen connectivity in the disrupted network for routing purposes
6. Develop partitioning approaches to generate roughly equal sized and balanced partitions in the disrupted network

1.3. Thesis organization

This thesis is organized as follows. Chapter 2 states our problem and the main targets to be addressed in this research. Chapter 3 explains essential definitions required for the reader to understand the studied topic and reviews literature concerning graph partitioning while

presenting various methods in this area. In addition we introduce elements that are closely related to system's survivability along with the area of attention in previous works concerning supply chain disruption. Chapter 4 presents the solution approach. We define the input property and our overall strategy concerning a disrupted network and demonstrate how the suggested methods intertwine with disruption modeling. In section 5, we demonstrate the applicability of our work and its efficiency in modeling large scale networks with higher uncertainty by testing the proposed approaches on a few certified networks with different properties. A table set of calculation concerning different percentage of disruption on our pre-defined network is included. Finally we summarize, provide conclusions and propose future works in chapter 6. The organization of this thesis is illustrated in Figure 1 below.

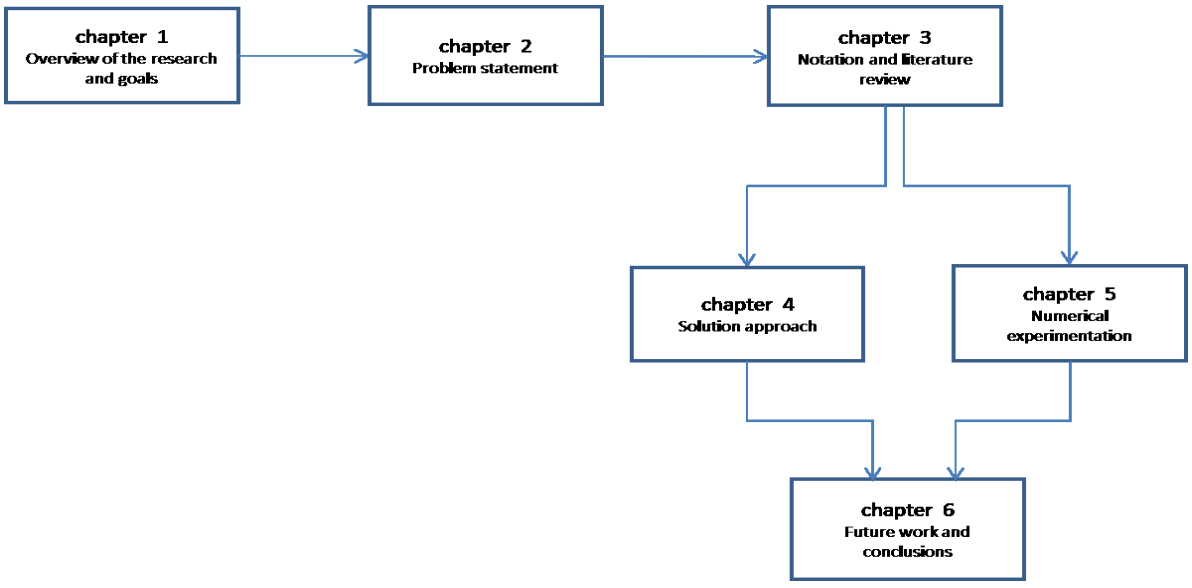


Figure 1: Organization of Thesis

Chapter 2:
Problem Statement

2.1. Problem definition

The three set of problems concerned in this work are:

- How to model disruption on logistics networks?

  Our main concern when modeling disruption is the type and level of disaster. As for the type we need to investigate the point of occurrence meaning whether it has occurred in the node or the link or both. In terms of level we must take into account state of failure, that is, whether it is a temporary damage or permanent.

- How to assure and strengthen the connectivity of the disrupted network?

  One essential act dealing with this matter is to collect all necessary information regarding supply and transportation; and transferring these raw data into practical information in a centralized establishment for future need. Nonetheless maintaining continuous flow of information to related sections is as important as the previous step. We must also investigate a procedure that will improve the system functionality in the occurrence of an event. For instance in a military-based network the complexity and sensitivity of the distribution is high as it involves moving both materiel and forces.

- How to generate roughly equal sized and balanced partitions in the disrupted network?

  Algorithms capable of providing good partitioning are important as they promise efficient execution on scientific simulations. Many groups of simulations such as parallel computers where high performance is vital can’t achieve such kind of performance through single objective techniques (traditional partitioning) and therefore

multi-objective versions are required to accomplish these set of requirements. In general, the structure of the computation in scientific simulations changes at each phase, and techniques mostly start off with an initial decomposition and then it continues balancing the load work at certain intervals of the simulation. Each segment iteratively entails a computational performance step and a data exchange step respectively. Two most common metrics that must stay balanced to indicate an efficient algorithm in the area of optimization problems are computational complexity concerned with the required processing time and optimality of solutions via capacity utilization assessment. Nonetheless, the network attributes and their underlying characteristics may change from one to another based on their expected objectives. Therefore each system might as well require particular metrics that are appropriate to specifically evaluate functionality of that certain network. For instance, in an army network it is necessary to perform process measurement via metrics to measure and evaluate the level of customer satisfaction. For example, applying supply chain and performance metrics in order to measure the performance of linked facilities. The former metric analyzes the entire supply chain with independent process integration and the latter measures a particular process in the supply chain.

Chapter 3:

Literature Review

In this chapter, we present the notations and formulations, an overview of the basic concepts of graph theory, and graph partitioning approaches which are the basis of solution approach proposed in Chapter 4. We also discuss elements that are closely related to system's survivability along with the area of attention in previous works concerning supply chain disruption.

1. Preliminaries and general description

- **Notation**

A graph G (V, E), is a network consisting of a specific number of nodes $v \in V$ with edges connecting them $e \in E$. v_i and v_j are connected if a path exists between them in which, one is the starting point and the other is the final point ,also they are called adjacent if the edge $e = (v_i, v_j)$ is a component in E. Note that each edge connects two different nodes so loops are not allowed. One other property affiliated with a graph is that each node has an assigned number indicating the degree of the node that is equal to the number of edges (arcs) connecting to it.

Plotting a graph depends on the structure of our computational network that is either performed on the nodes, elements or both. Node graph presentation is where a vertex exists for each node and displayed as a point while edges are introduced as lines displaying their connection thus each edge has two endpoints stating communication. On the contrary dual graph is when the computation is performed on elements, then the points in the modeled graph represent elements and the lines introduce shared edges /faces among them. (Figure2)

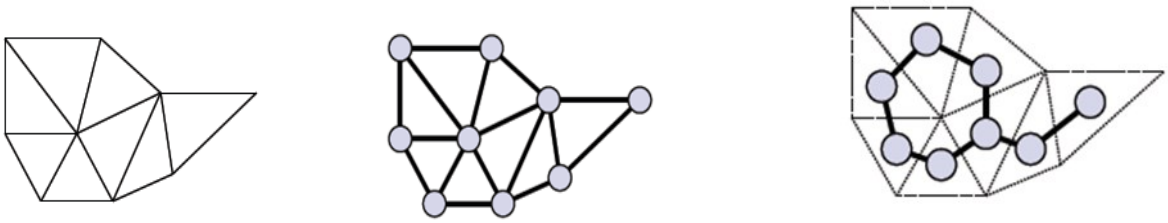


Figure 2: Graph (a) shows a 2D irregular network, graph (b) models a node graph, representing communication and graph (c) models a dual graph, performed on the mesh elements representing adjacent elements

In addition since different graph types embody particular characteristics, it is possible for each node to have a size s (v); and each edge having an assigned weight w (e) representing workload or the amount of dependency. Moreover edges can display direction separating them into directed and undirected graphs.

Undirected graph by definition is when there are no directions associated with the edges. That is, the edge from v_i to v_j and the edge from v_j to v_i are the same and interchangeable. $(v_i, v_j) = (v_j, v_i)$

A directed graph by definition is when a direction is assigned to each edge. More specifically the edge linking v_i to v_j distinguishes from the edge connecting v_j to v_i. $(v_i, v_j) \neq (v_j, v_i)$. This indicates that one node will be the successor of the other and as to a real time network this means that the execution of one task depends on the completion of its successor.

Further, in terms of associating a weight to the arcs of a directed graph, considering G_1 as an undirected graph and G_2 as a directed graph with a weight attached to each arc, then we have:

G_1 = *(N, E)* where *N* is a set of nodes and *E* is a set of edges between nodes.

G_2 = *(N, E, Le)* where *N* is a set of nodes, *E* is a set of edges between nodes, and *L* denotes a

square matrix whose entities represent the weight associated with each edge, with all entries on the diagonal equal to zero, indicating that self-loops are not permitted.

Graphs in Figure 3 display directed and undirected graphs.

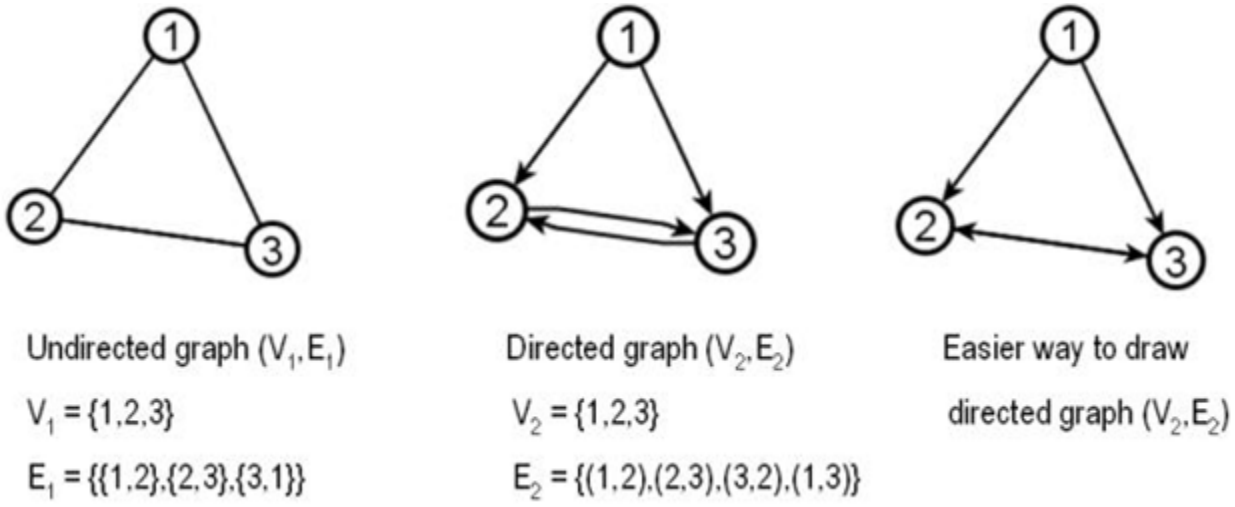


Figure 3: Directed and undirected graphs

One substantial part of our research is assigned to efficiently partitioning a graphical network. A common definition of a weighted *K*-way graph partitioning problem is the process of dividing a graph $G = (N, E, W_N, W_E)$ into approximately *K* equal sized subsets $N = N1 \cup N2 \cup \ldots \cup N_k$ while minimizing the inner connection and maximizing the intra connection among them. Where *N* stands for nodes, *E* indicates edges while W_N and W_E are associated weights to nodes and edges respectively.

The given graph will be decomposed into *p* disjoint subgroups; whose union equals *N* and their intersection is *Ø*. This will guarantee that no node is a member of more than one cluster.

Further, concerning our suggested methods, *W* and *Q* are two main parameters to be discussed in this section *W* is a user-defined value for the cardinality indicating the maximum number of

nodes present inside a subset. Let $card_i$ denotes the cardinality of subset G_i. Note that $card_i \leq W$.

Q is a user-defined value representing the density of a subset, which is the ratio of the number of external connections of a subset divided by its cardinality (number of internal nodes). Q_i Corresponds to the density of the subset G_i.

W and Q are parameters that highly affect the quality of clusters thus, must be specified with care. For this matter, the value of W must be greater than 0 and less than the total number of nodes N. Likewise, (Awasthi, et al. 2009) Q should not be too high or too low, as low density will result in isolated subgroups and high density will cause large number of inter-connections which is contrary to the stated objectives of the paper. For more information on graph theory we refer to (Bondy and Murty 1976)

2. Graph partitioning

Partitioning $v \in V$ of graph G into k disjoint subsets provides a mapping of either the mesh nodes or the mesh elements onto k processors. In simple words the goal of solving a problem through graph partitioning is to implement a computational method in a way that both the overall runtime and communication are reduced. It is desirable for each chunk to have the same amount of node weight so that each processor handles an approximately equal fraction of workload (balanced subsets) and as well minimizing the edge weight between processors of two subsets (Minimized edge-cut).

More specifically, the total communication volume during parallel processing is estimated by counting the number of edges that connect the vertices of different subsets namely edge-cut. A well-chosen decomposition should minimize this metric; although the edge cut metric and the total communication are not always the same in volume. The reason is that firstly the edge- cut

counts every edge cut and secondly as in total interaction the data must be sent once, even if two or more edges of the same vertex are cut by the same subset. Therefore this metric isn't solely adequate to predict the inter-processor communication. Accordingly a more reliable measure with accuracy is taken in to account which is the maximum time required for each processor to perform communication. This strongly depends on the number of processors; a processor must communicate with as well as the amount of information that must be swapped among processors. Figure 3 is a good example that illustrates the abovementioned issue. As depicted the total communication between A, B, C is 9 while the edge-cut is equal to 7.

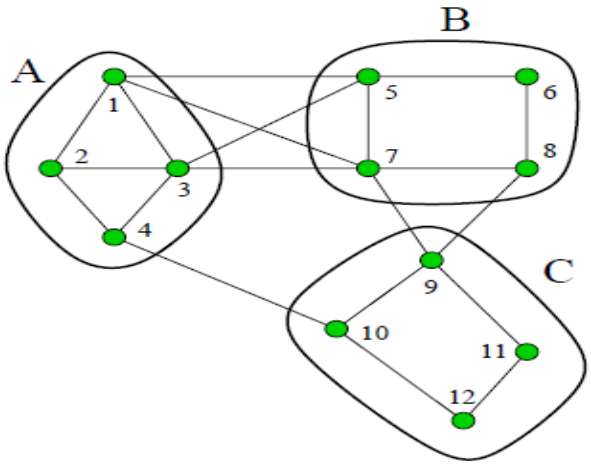


Figure 4: A partitioned network

Balanced subsets are attained through restructuring and levelizing processors in a way that each processer runs an equal execution time; also each processor deals with the same storage demand namely balancing storage. Nonetheless it is notable to mention that partitioning is done based on the type of the problem. For example in a heterogeneous cluster representing workstations it is desirable to carry the extra subset weights on to faster machines. Another example would be

a workstation with uniform graph models projected on them where most vertices have about the same number of edges; here there would be a strong correlation between edge-cuts and inter-processor communication, hence traditional balanced graph partitioning is considered which aims to minimize the cuts. *K*-way partitioning through recursive bisection is executed through following steps:

- Graphs are frequently partitioned into *k* sub graphs.
- If *k* is not a power of two, the *k*-way partitioning must be computed in a way to ensure balanced sectors. Accordingly bi-partitioning is easier to implement than *k*-way partitioning since there is no need to choose the destination part of vertices
- After computing our separator, the two separated sub graph must be reconstructed by a sequence of iterative bipartitions on half of the available processors.

In general; (Meyerhenke, Monien and Sauerwald 2008) Graph partitioning methods can be categorized into two major types' namely local improvement methods and global strategies. Local search are edge based techniques that aim to find cuts close to the starting node regardless of their connectivity status and start off with an initial arbitrary based partitioning. They claim to be fast relative to the size of the small side of the cut; however they are lower in quality compared to global strategies. Mainly because they do not take connectivity matrix in to account and system improvement is merely based on gain and loss ratio as well as its difficulty to employ on parallel computation. (Andersen, Chung and Lang 2006) Global strategies on the other hand, are mathematical based approaches that perform partitioning with respect to the properties of the entire network. All related methods require calculation and entail mathematical analysis therefore they are known to be time consuming techniques; however the quality is considerably higher.

The urge of graph partitioning strongly arises when dealing with a large scale network and handling network complexity. Graph partitioning techniques are considered to be a very fast and effective solution in various fields of computational science and engineering. In this regard many platforms such as VLSI circuit design, data mining, image processing, parallel processor computation and work distribution, spars matrix recording, telecommunication networks and many other related operational research have taken advantage of this fundamental optimization problem method.

Graph partitioning is known to be a NP complete optimization problem. Due to the nature of the problem, as the size of the network increases the chances of finding the ideal solution in a reasonable time frame is almost impossible. More specifically in practice, solutions based on primitive algorithms are approximate and known to be general problem solving methods rather than specific tailor-made solving methods. In detail the solution obtained is more than often far from optimality along with other drawbacks namely slow computation and complicated implementation.

Therefore the importance of heuristic methods has been conceived to efficiently solve optimization problems by finding a low cost, near-optimal solution in an acceptable time. As a result a large number of heuristic algorithms have been proposed over the past decades. . (Schloegel, Karypis and Kumar 2006) . Overall provides a descriptive review of different graph partitioning techniques for scientific simulations based on their nature. It defines adaptive (dynamic) and static graphs and reviews different algorithms for each by classifying static graphs into four classes of Geometric (Berger and Bokhari, 1987; Heath and Raghavan, 1995), Combinatorial (Kernighan and Lin, 1970; Fiduccia and Mattheyses, 1982), Extended combinatorial optimization techniques , Spectral (Simon et al, 1997), and Multilevel methods

(Karypis and Kumar, 1998a). Here we will take time and provide a descriptive explanation for each method.

3. Graph partitioning approaches

3.1. Static graph partitioning techniques

Static graph partitioning algorithms follow a certain mathematical procedure across the algorithm iterations. They intelligently partition a graph when the network is not in operation. These algorithms mainly aim to satisfy the edge cut criteria.

3.1.1. Geometric computation

The geometric computation is somewhat considered to be a fast technique that ignores the edge-cut factor and is merely coordinate-based information on the nodes, with the goal of grouping nodes that are spatially close to each other regardless of their connectivity status. Therefore, minimizing the inter-processor communications requires schemes, designed in a way to minimize metrics such as the number of mesh elements that are next to non-local elements. Most important features of this type of techniques are:

- Applicability for graphs with coordinate system such as Crash Simulations and Contact Detection, Adaptive Mesh Refinement, Particle Simulations and Parallel Volume Rendering
- Simplicity since no connectivity information is required.
- Iteratively splits mesh into bisections.
- Conceptually fast.
- Since connectivity between elements is not considered, partitioning is known to be low in quality.

- There is no cost control or optimization on communication.

- Forming disconnected subsets in complex calculations is common.

Coordinate Nested Dissection (CDN) is a frequently used scheme in this category. Recursive Inertial Bisection (RIB), Sphere-Cutting Approach and Space-Filling Curve Techniques are three other well-known schemes worth mentioning. (Andersen, Chung and Lang 2006)

3.1.1.1. Coordinate nested dissection (CND)

"Recursive Coordinate Bisection (RCB) "is a static load-balancing algorithm that was proposed by Berger & Bokhari in 1987 and plans on maximizing independency by minimizing inter-processor communications. In other words, smaller portions of information and data are exchanged by cutting down the boundary between the subsets. Since the technique produces incremental partitions, therefore it has also been studied under the name of dynamic algorithms. CND bisects the domain with a plane that is perpendicular to one of the coordinate axes(X, Y or Z). The procedure starts off by computing the center of the mass for each element. It then maps the correspondent points on the coordinate axis that is the longest dimension of the mesh with respect to the geometric location of the entities. The elements are sorted in an orderly fashion and then divided in half to produce bisection. The method is recursive hence subsets are iteratively divided with the same splitting technique until the optimum is achieved in other words the number of subsets matches the number of processors. (figure5a)

Advantages

CND appears to be extremely fast and requires little memory. Therefore the partitioning is quite compact. It is known for easy parallelization. Furthermore the cost associated with data redistribution is low since small changes in data causes insignificant movement in cut.

Disadvantages

In general computation is restricted due to the fact that orderings are computed on a single dimension which is one of the coordinate axes. Thus, the resulting subsets are of low quality. Also complicated geometries lead to disconnected subsets.

3.1.1.2. Recursive inertial bisection (RIB)

RIB is an advanced model of CND that was proposed by Simon in1991 and studied afterwards by Taylor & Omid in 1994. RIB is capable of computing bisections that are orthogonally rotated around the inertial axis rather than solely the coordinate axis. RIB bisects the domain (figure5b) with a plane at any angle relative to the coordinate axes. This in turn provides higher quality decomposition. The procedure starts off by computing its prime inertial axis and then maps the center of the mass onto this axis. Sorted elements are split in half and bisected. The procedure is repeated by dividing the subsets according to the technique steps to the point where the optimum is achieved. Appropriate size adjustment for each part helps generating equal-sized subsets. However in parallel computation processors with different speed must be set with non-uniform sizes.

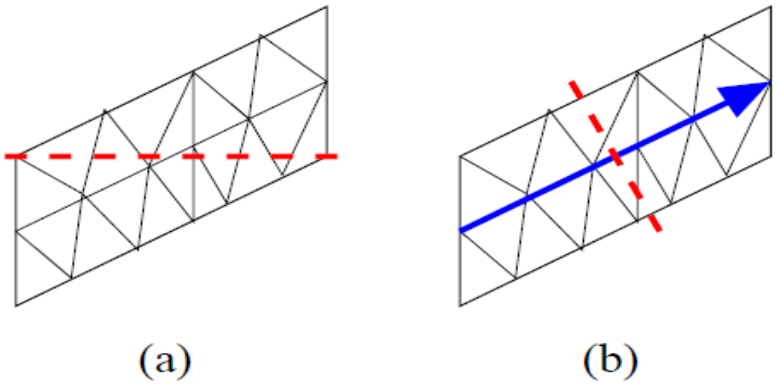


Figure 5: Bisection by CDN and RIB; a and b respectively

Advantages

The resulting decomposition is obtained in a shorter time and the shared face/edge of subsets is smaller than the one produced by CND scheme.

Disadvantages

Computation of the ordering list is done by mapping elements on a single dimension also the separators computed are known as hyperplanes. (Chamberlain October 13, 1998)

3.1.1.3. Space filling curve techniques (related self-avoiding walk)

Space filling curves method is defined as a mesh rearrangement mathematical function that attempts to map a high dimensional domain into a one dimensional hyperspace. (Aftosmis, Berger and Murman 2004). They are constructed in such a way to continuously fill up high dimensional spaces such as squares/cubes spheres into smaller unit of somewhat same shape by applying one of the many certified ordering methods such as Peano-Hilbert curves. SFC it starts off by computing the center points of mesh elements followed by sorting them based on their closeness. In other words the objective is to traverse all subdomains while preserving locality and neighborhood. Finally the list of ordered elements is split into k parts resulting in k subsets.

Advantages

It is known to be fast and simple and thrifty. It produces a higher quality decomposition compared to CND and RIB method since it considers more than one dimension at a time and has the ability to propagate repartitioning. (Figure 6)

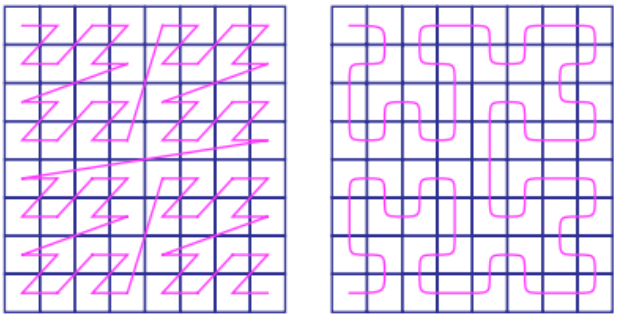


Figure 6: Space filling curve with two different ordering

Disadvantages

It performs better on certain type of problems where computation greatly depends on spatial closeness of objects for instance hierarchical methods in n-body computations.

3.1.1.4. Sphere-cutting approach

(Miller, et al. 1993) , (Chamberlain October 13, 1998) Sphere-Cutting decomposition also referred to as recursive circle bisection is known to be practical for a variety of graph problems. The method is capable of dealing with overlap graphs that basically consists of both well-shaped meshes and planar graphs and it can also tackle downsides of hyper plane-based algorithms. To that end it conducts a global search on vertices and defines separators as circle and spheres instead of line and planes that offers extra degree of freedom to the problem. These graphs have $O\left(n^{\frac{d-1}{d}}\right)$ vertex separators. A vertex separator is by definition a set of vertices that can be removed and split a graph into two roughly equal-sized subdomains without any edge linking them. This means, instead of partitioning the graph between the vertices resulting edge cuts, the graph partitions vertices side by side. For this matter, the total weight of the vertex separators should be minimized. (Constraint)

The main idea behind this approach is the usage of a neighbourhood system in decomposing an

overlap graph. In that event a *k*-ply neighborhood is applied. A *k*-ply neighbourhood system is a set of *n* spheres in a *d*-dimensional space where each point can only be surrounded with only *k* out of *n* spheres.

a *(∝, k)*-overlap graph comprises of spheres corresponding to each vertex and are such that two nodes are adjacent if spheres, expanded by α, intersect in other words a direct edge is between them.

(Gilbert, Miller and Teng 1995) The method proposed maps nodes of a *d*-dimensional graph on to the unit *d+1* dimensional sphere surrounding it. In order to decrease the level of complexity and as well accelerate the process the algorithm generates a random sample of projected nodes representing the graph. Next it computes the center point for the projected nodes. For large scale graphs center points are calculated through a heuristic algorithm instead of a linear programming algorithm to speed up computation. The result is achieved by selecting a few best circles (partitions) in a randomly fashion and choosing the best among them that is the one that has the fewest edge cuts.

Advantages

It guarantees high probability in quality when decomposing well-shaped networks.

Disadvantage

Due to randomness there are no guarantees for a perfectly balanced partition. (Gilbert, Miller and Teng 1995) Offers a modified version that will create a balanced partition by shifting the separator normally to its orientation.

3.1.2. Combinatorial techniques

Geometric partitioning are fast, understandable and easy to implement but they all have two mutual flaws that is; firstly even though most algorithms in this class select a random sample of

nodes yet each node of the input graph requires coordinates. Secondly they don't take the connectivity status of the nodes into account and consider a short connecting path between nodes by reason of spatial closeness. At some point this assumption may seem logical but is not germane for all sort of input graphs therefore other class of algorithms have been conceived to address such kind of problems. Combinatorial techniques also referred to as structural algorithms are known to be more sophisticated as they include the graph's connectivity state. Here, partitioning is solely based on the graph's adjacency information rather than locality and the coordinates of the vertices. Simply put, it attempts to group together highly connected vertices regardless of their closeness in space. Most important characteristics of this class of techniques are as follows:

This in turn leads to smaller edge-cuts as well as lower chance of isolated subsets compared to the previous class. Although combinatorial techniques are slower than geometric schemes and considered to be a less practical for parallel processor computation, they are still known to be reasonably fast.

Advantages:

The output partitioning has lower edge-cuts and a lower chance of isolated subsets compared to those by geometric schemes.

Disadvantages:

It appears to be slower than geometric schemes and is considered to be a less practical for parallel processor computation.

Levelized Nested Dissection (LND) and Refinement algorithms namely Kernighan-Lin/Fiduccia- Mattheyses (KL/FM) partition refinement are two popular algorithms of this class with the objective of exploring moves that reduces edge-cuts while satisfying balanced constraints. In this section, we explain these two approaches in detail.

3.1.2.1. Levelized nested dissection (LND)

The procedure is a graph growing algorithm that seeks to put together connected nodes, starting by randomly selecting a single node as a subset referred to as v_0 and assigned to the first partition.it is highly desirable to choose an exterior node as v_0. The next step would be to add adjacent nodes to increase the size of the subset. To that end the distance for each adjacent node to v_0 is determined through breadth-first search (BFS). This process is repeated iteratively till half of the nodes are united with v_0. At this point, the graph is split into two subsets namely assigned, unassigned and the algorithm terminates**.** Furthermore we can test LND algorithm with different starting nodes in order to achieve the best possible partition.

- LND algorithm promises at least one well-connected subset as long as the input graph is fully connected.
- The method argues that it is capable of producing subsets of same or higher quality than the previous class, though the argument is not definite.

3.1.2.2. Refinement algorithms

The output partitioning computed through a method is not always the optimal solution. Therefore, the urge of refinement techniques has been conceived. Refinement techniques can be categorized into an independent group as they attempt to enhance the quality of an initially developed partitioning. Nonetheless due to its closeness to partitioning problems, we consider a subgroup of combinatorial schemes. Recently great attention has been given to such kind of algorithms as they have not only turned out to be very useful and efficient in amending the output of other partitioning methods but also has proven to be very practical when partitioning from scratch through simply selecting a random partitioning as its input.

The first algorithm was developed by Kernighan and Lin in the late 60's, named KL partition refinement and later an improved version called KL/FM partition refinement. Generally both procedures aim to reduce the number of edge-cuts and may start off by improving a randomly selected partition by flipping over nodes from one subset to the other. Here we describe each algorithm in detail.

3.1.2.2.1. KL Partition refinement

The KL partition refinement is basically based on a metric that determines if trading nodes between subsets can be beneficial. Here an initial bisection of a graph, preferably equal-sized subsets namely A and B, is given. KL algorithm is applied by finding an $X \subseteq A$ and a $Y \subseteq B$ in a way that swapping X to B and Y to A leads to greatest reduction in edge-cut that in turn helps improving the quality level of the partitioning. Our goal is to reduce the node's gain by simply reducing the total interconnections. The trouble finding the best set of X and Y could be achieved through a greedy method that is, moving nodes with maximum positive gain, up to the point where no set of nodes with such characteristics are left unswapped. Next step is to select and restore the bisection that has the least edge cut. Finally, all remaining vertices that were moved after the selected bisection are moved back to their first place. An additional step would be to consider this new partition as an input for an extra pass.

Noteworthy points of the technique are as follows:

- In KL refinement method each pass takes approximately $O(|v|^2)$time.
- The possibility of encountering local minimum is high; therefore KL solves the problem by allowing hill climbing that is the permission to move nodes with negative gain with the aim of reaching a global minimum.

3.1.2.2.2. KL/FM partition refinement

KL/FM Partition Refinement proposed by Fiduccia and Mattheyses is a modified version of the basic KL algorithm with the purpose of reducing the overall runtime through enhanced data structures. Partitioning through a globally-based approach can improve the result by executing FM algorithm as an additional step. In general its job is to adjust the gain and loss values by minimizing the sum of the nodes after node migration. The major difference between these two techniques is that the latter method considers moving one node at a time. The procedure runs as follows:

1. Note down values of gain/loss for each node along the partition.
2. Generate a priority queue for each subset, sorted by the gain obtained by node's migration.
3. Move the node that allows both edge cut reduction and balance improvement.
   I. If the highest node in one queue maintains the balance constraint, then node migration is done.
   II. If the highest node in both queues satisfies the balance constraint, then select that node with the greatest gain.
4. Omit the migrated node from the queue and update the gains of the adjacent vertices.
5. Stop passing when there is no node left to move.
6. Here the bisection with the highest quality is selected and restored.

Strength and limitations of the scheme are as follows:

- The complexity of each pass of the FM algorithm is equal to O ($|E|$).
- Node migration is done regardless of its gain negativity or positivity.
- KL/FM is capable of escaping local minimum.

- KL/FM scheme has a restricted ability therefore the quality of the final bisection, strongly depends on the quality of the input bisection
- KL/FM algorithm has the capacity to modify and adjust itself depending on the requirements of the programmer, the area of interest and the problem definition. For instance considering weighted graphs, P-way partitions or when applying more complex standards for gain. (Chamberlain October 13, 1998)

3.1.3. Spectral methods

Spectral schemes known as challenging partitioning schemes are high in quality but expensive. A high level structure of the scheme consists of three basic steps namely: pre- processing, decomposition and grouping. The general approach is to formulate the problem into a discrete quadratic function and transform the discrete optimization problem into a continuous one.

The procedure is as follows:

- Input graph *G* with adjacency matrix *A* whose diagonal element provides matrix $D[i,i]$ = degree of node *i.*
- Pre- processing :
    - Matrix *LG* is a discrete Laplacian matrix generated to represent the dataset and is equal to *A* - *D*
    - Decomposition: values required for this scheme are eigenvalues and eigenvectors that provide the global information about the structure of the graph.
    - Since *LG* is a negative matrix its largest eigenvalue is 0 therefore the second largest eigenvalue measures graph connectivity.

- Problem minimization is solved by computing the corresponding eigenvector of the second largest eigenvalue called Fiedler vector with the purpose of measuring distance between nodes.

- Grouping:
  - Fiedler vector is sorted incrementally and graph segmentation.

Strength and limitations of the scheme are as follows:

- Produces higher quality partitioning than geometric schemes.
- Computing Fiedler vector for large data set is expensive and infeasible in real time.
- Great attention has been focused on speeding up the algorithm by taking Lanczos algorithm and multilevel methods in to account to practically compute eigenvector.
- By applying spectral bisection recursively a k-way partition is computed.
- K-way decomposition has also been investigated by clustering multiple eigenvectors in units of four and eight. This in turn prevents instability due to information loss.
- The latter *k*-way partitioning approach is more efficient and less costly compared to recursive spectral bisection, since it considers approximate spectral clustering.
- The splitting point can be calculated either by basic approaches such as the mean, median or at 0 or through more expensive approaches such as normalised cut criterion in 1-dimension.

3.1.4. Multilevel schemes

Lately, great attention has been designated to this class of optimization problem. Multilevel schemes consist of three phases respectively: graph coarsening, initial partitioning and partition refinement. The common computational structure applied recursively is as follows:

1. Graph coarsening: is to coarsen the original graph by breaking down pair of nodes that forms a matching preferably heavy –edges pairs. At each stage the coarsened graph acts as the input for another round of this procedure and continues until the smallest graph of its kind is obtained.
2. Initial partitioning: is performed on the coarsest graph (smallest) using a standard approach such as recursive bisection method.
3. Partition refinement: is executed on each graph from the coarsest up to the largest graph using KL/FM algorithm.

This paradigm appears to be very efficient while resulting in a short period of time. It performs well by hiding a large number of nodes and edges on the coarsest graph. Moreover moving a single node in a coarsened graph is equal to moving a large number of highly connected nodes in the original graph. Subsequently in this context incremental refinement schemes such as KL/FM perform better by escaping problems such as local minimum. Multilevel partitioning algorithm has been implemented in Chaco [Hendrickson and Leland, 1994], MeTiS [Karypis and Kumar, 1998], SCOTCH [Pellegrini and Roman, 1996], PARTY [Preis and Diekmann, 1997] and JOSTLE [Walshaw, 1998] software packages.

(Karypis and Kumar 1998) Studied multilevel paradigm broadly highlighting the advantages that are as follows:

- The scheme is robust.
- It surpasses the previous methods in terms of speed and quality.
- The initial partitioning method implemented on the coarsest graph does not jeopardize the overall quality of the solution, in other words supposing the initial partitioning of the coarsest graph is refined poorly, by employing a local refinement method in the multilevel scheme high quality partitioning is attained regardless.
- Capable of reducing the total edge weight. That is heavy –edge matching (HEM) is much more effective in hiding edges in the coarsest graph, compared to random matching (RM). in other words in the HEM algorithm the coarser graph combines nodes holding greater weight edges leading to minimized edge weight.
- The total run time of the multilevel recursive bisection scheme is O (|E| log k).

Other benefits of such scheme is that

- Partitioning on the coarsest graph maintains higher quality compared to the original graph and works better than the traditional single level techniques.
- Movement of a single vertex in a coarsened graph is equal to the movement of a large number of highly connected vertices in the original graph.

Multilevel *k*-way partitioning

(Karypis and Kumar 1998) presents an alternative that is a generalized form of the previous versions of the KL/FM algorithm concerning *k*-way partitioning refinement. The first step, graph coarsening, is performed as in its original scheme. But as for the second step the coarsened graph is directly divided into *k* parts. Finally the output of the previous step is refined and subsequently un-coarsened back into the original graph.

Strength and limitations of the scheme are as follows:

- The method proves to be more practical when directly applied on a graph rather than following a computation via recursive bisection. Since firstly it only computes the coarsened graph once, therefore great reduction in complexity is gained leading to faster runtime which is linear in the number of edges and is $O(|E|)$. Secondly, in general, recursive bisection is known to do worse than k-way partitioning.
- The run time is compatible with the run time of geometric recursive bisection algorithms with 2 to 4 runs but with a higher quality.
- It creates a better quality decomposition compared to multilevel recursive bisection.
- In comparison with multilevel spectral bisection it produces partitioning with generally smaller edge-cuts.

3.2. Dynamic graph partitioning techniques

(Schloegel, Karypis and Kumar 2006) also present techniques developed to undertake applications with unfixed workloads during the computation steps and are related to network flexibility and how to effectively deal with complication and complexity of data assignment. The techniques are known to be cost effective and useful when the workload evolves over time and the network dynamically changes. In large scale networks redistributing data and balancing workload through local refinement and de-refinement is rewarding as it can not only accurately capture the flow-field phenomena of interest but also considers a capacity for a diverse range of errors. In general, these methods would be less time consuming if the input partitioning is disturbed just enough to balance the workload rather than dealing with redistributing excessive data.

Dynamic load balancing is obtained by utilizing a graph-partitioning algorithm where the graph corresponds either to the adapted mesh or to the original mesh with the node weights modified

in order to resolve error estimates. The amount of rearranged data among the processors is minimized in order to balance the computation which means minimizing node migration with respect to the node size (redistribution cost of workload).

(Oliker and Biswas 1998)Present metrics to measure distribution costs as:

I. Total V defined as the total amount of nodes switching between subsets; that is the total volume of communication required to balance repartitioning.

II. Max V equal to the highest amount of relocated nodes in the network and measures the maximum time required to send/receive data.

The goal of repartitioning is to minimize both metrics. However refinement is easier through simple heuristics, thus some schemes attempt to minimize the Total V instead, that will automatically lead to a minimized Max V. One other important reason is that Max V equals an amount that is less than the total node weight transferred into an underweighted cluster.

In general, adaptive schemes are distinguished by the quality of the partitioning result, the execution speed and the amount of transferred data. For instance there may be a scheme providing a high quality result but with a high communication cost or a scheme resulting in a short time while the quality is not so very good. Here the system management has to determine and decide on the best solution strategy concerning the area of interest.

Most common approaches are namely cut and paste repartitioning method, scratch-remap repartitioners and diffusion-based repartitioners:

I. Cut and paste repartitioning method: overweighed subsets are balanced by relocating their excessive number of nodes into the underweight subsets despite their adjacency status. This in turn will minimize data redistribution but can lead to higher edge-cuts as well as disconnected clusters and is hence not used in many applications.

II. Scratch-Remap repartitioners: (Oliker and Biswas, 1998) attempt to compute the new partitioning from scratch by design. It maps the labels assigned to subsets in accordance to those of the original partitioning in order to minimize the data redistribution cost. This is done by constructing a similarity matrix, *S*, of size $k * k$. A similarity matrix is a square matrix where rows represent subsets of the old partitioning and columns represent subsets of the new partitioning. Moreover each element represents the sum of the sizes of the nodes that are both in the subset of the old partitioning and the new partitioning. The method will result in higher redistribution costs(Total V) compared to other schemes that starts off balancing the input partitioning by a minimal disruption (cut-and-paste and diffusion-based schemes). It then solves the problem by combining elements in such a way that every row and column contains one and only one selected element and the sum of the selected elements has the highest value among the rest.

III. Diffusion based repartitioners: through accumulative changes the difference between the original partitioning and the final repartitioning is minimized. In the original partitioning nodes located in overweighed subsets are transferred into their neighbors. These subsets may in turn swap their excessive number of nodes to their next adjacent subset in order to reach global balance. Therefore the possibility of creating extra edge cuts or isolated subsets is minimized.

Two main concerns when practicing diffusion-based repartitioning schemes is firstly, the amount of work transferred between processors, that is balancing the workload and secondly tasks that should be transferred to minimize the edge cut. In the context of balancing workload diffusion based schemes are generally classified as:

- Local diffusion schemes concentrate on exchanging workload among the processors solely based on their corresponding workload rather than the loads of geographically distant processors.
- Global diffusion schemes execute on a global view of workloads across processors in order to balance the partitioning. In general global diffusion schemes manage to handle diffusion by methods capable of defining the amount of work moved between two processors namely recursive bisection, space-filling curves or flow solutions.

Nevertheless there's a tradeoff between cost and communication volume in all adaptive repartitioning algorithms. More specifically minimizing the data redistribution cost conflicts edge-cut minimization thus it is critical to understand the problem's priority. In other words decisions are made based on the status of our problem for instance if mesh adjustment happens frequently or the amount associated with each element is relatively high; we persuade minimizing the data redistribution cost rather than edge-cut minimization. On the other hand for applications where repartitioning does not occur frequently, obtaining the minimal edge-cut is preferred.

3.2.1. Parallel graph partitioning parallelism

The above mentioned classifications of graph partitioning problem, static and adaptive, are generalized schemes that must be formulated in order to address their application to parallel computing. Mainly adaptive schemes are developed in order to operate on parallel procedures by rearranging data that has been already distributed.

Studies in the field of parallel graph partitioning, have been focused on

- Geometric partitioning schemes as

- Their runtime is not affected by the initial distribution.
- They can be used to compute the partitioning of the input graph for the next two schemes

- Spectral partitioning schemes
- Multilevel partitioning schemes

The last two schemes are harder to parallelize through a randomly partitioned input graph; therefore a well distributed input graph among processers is essential for high performance. Also in various problems the nature of the graph determines the level of quality required for the initial partitioning. As for static graphs consisting of unchanging sequence of nodes and links we can't expect a clustered input graph since this is what the algorithm must produce in the first place. In the case of dynamic graphs due to their unstable nature we refine the graph from an initial cluster configuration among processers that hold small edge cuts .This attempt accelerates the process and is also cost effective. Further in favor of parallelism due to higher memory and space, solving problems and graphs in large scale is possible. Also higher efficiency in the matter of time and cost is the other advantage of parallel computers. Last but not least well scheduled parallel processors will also prevent bottle necks.

3.2.2. A generalized formulation for graph partitioning

As mentioned earlier due to lack of adequacy in basic schemes, imbalanced memory space and bias computation, restriction in the scale of the problem remains. This appears to be more challenging for sophisticated classes of simulations namely multi-physics, multi-phase, and multi-mesh as former schemes are not capable of balancing more than one constraint and minimizing more than one objective. For instance constructing a balancing node weight

procedure while minimizing the number of edge-cuts. Thus an extended version of graph portioning methods must be formulated in order to guarantee their practicality and efficiency in the performance of such simulations. Here we take time to first describe each class of simulation separately. We then describe are generalized formulation of the graph partitioning problem.

Multi- physics simulation:

This class embeds the simulation of multiple materialistic constraints together. Therefore unbalanced computation and memory requirements that lead to inefficiency and size restriction are respectively addressed. In consequence a number of objectives are satisfied at the same time. (Schloegel, Karypis and Kumar 2006) provides an example in this regard. As shown in figure (7) Graph (A) is the original graph distinguishing computation and memory associated with each vertex.

Graph (B) is a balanced computation vis-à-vis non-uniform memory requirements.

Part (C) is a balanced memory requirements vis-à-vis imbalanced computation.

Part (D) is where both constraints are balanced.

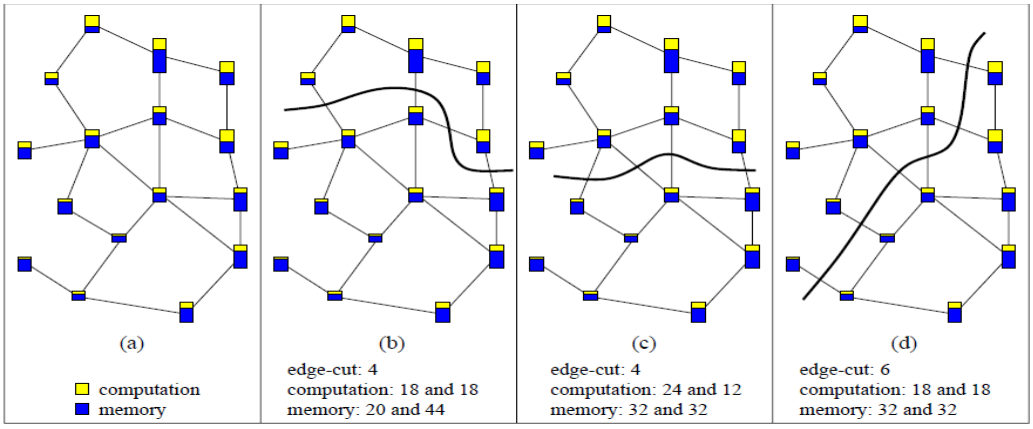


Figure 7: An example for computation and memory requirements

Multi-phase simulation

Multi-phase problem is a combination of *m* disjoint computational phase, where each phase is separately decomposed and the outcome is injected in to the next phase. In other words each phase follows an individual synchronization step that is to balance the work load. This in turn involves different amount of computation for each element in different steps of computation. Therefore instead of summing up the required time to accomplish each phase and solving the problem based on this estimated time; Problem computation in this simulation is based on an equal amount of workload for each processor, which is aggregated from all computational phases. Consequently it prevents imbalanced workload and idle machines throughout execution. Moreover maintaining a minimum inter-communication among processors as well as an approximately equal share of the network entities for each processor during different phases are critical factors.

The two following methods can help computing the problem.

1) By using *m* separate decomposition, where each one is capable of balancing the workload of one phase of the process. This in turn requires constant data redistribution between phases which will increase the communication costs.
2) By using a one shot decomposition that simultaneously balances the work with respect to multiple loads in each phase, thus no extra data redistribution is required and is more cost effective.

Multi-mesh computations

Multi-mesh computation is an important class of mathematical methods. This specific problem calculation appears efficient in problems such as radiation transport sweeps, FFT where

structured grids are used to discretize partial differential equations; while in complex geometries an unstructured mesh seems to be more practical. Nevertheless due to problem requirements in different operations as well as the quantity of variables, a particular grid that is appropriate to solve the equation maybe considered for each variable. An example would be the simulation of welding two parts together. The following steps are considered as the basic blueprint for a multi-physics procedure running on a distributed-memory parallel machine:

1) Computing a solution on the first mesh
2) Interpolating the result to the second mesh
3) Computing a solution on the second mesh
4) Interpolating the result on the first mesh
5) So on

Note that regardless of the efficacy of the applied method where partitioning meshes independently creates a balanced computation and a minimized communication through each phase nevertheless the chances of excessive data transfer during the update-step and inter-processer communication is high.

Domain decomposition preconditioners

Iterative methods are more likely to provide an efficient solution when

- Matrix-vector multiplication is efficiently implemented in parallel by:

1) Minimizing the number of non-zero elements that are off of the diagonal block matrix by taking edge cuts into account and rearranging elements.

Note that each non-zero element off the block diagonal corresponds to an inter processor communication.

2) Taking account of edge- weights will aid minimizing the magnitude of non-zero elements that are off the block diagonal.

- The number of iterations required for the method to converge is minimized by:

1) setting preconditions for each block of the matrix
2) applying a combination of all preconditions for the entire matrix

Note that non-zero elements off the block diagonal are ignored in these preconditions.

3.2.2.1. Multi-constraint and multi-objective graph partitioning

One way of formulating a multi-constraint, multi-objective graph partitioning problem follows these steps:

1) A weight vector of size "*m*" is given to each node and a weight vector of size "*l*" is assigned to each edge.
2) A partitioning that maintains a minimized edge-cut while respecting the assigned *l* weights, and maintains balanced for each *m* weight across subsets.

Consequently where our problem definition is to balance computation and memory, each node is tagged with a vector (*i, j*) where *i* and *j* represent node requirements (two constraint problem). Also where the problem is concerned with the computation of the ordering for a sparse linear system that has been preconditioned by a block diagonal method, each edge is tagged with a vector (*i, j*) where *i* and *j* represent the number of non-zero entries and their magnitude (two objective problem).

Therefore Multi mesh computations are in fact multi-constraint and multi-objective computations. The illustrated problem in figure 8 and 9 where

Figure (A) shows two overlapping meshes

Figure (B) shows the graph associated with the problem

Figure (C) shows the four- way partitioning of the graph (balancing both node types as well as minimizing their edge-cut)

Circular node (0,1)

Square node (1,0)

Red solid edges (1,0,0)

Blue dotted edges (0,1,0)

Black dashed edges (0,0,1)

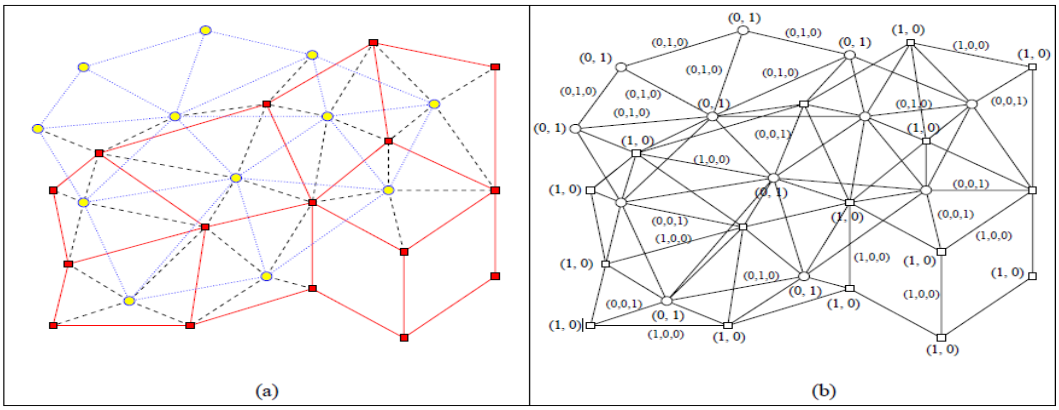


Figure 8: Overlapping networks and their corresponding graphs

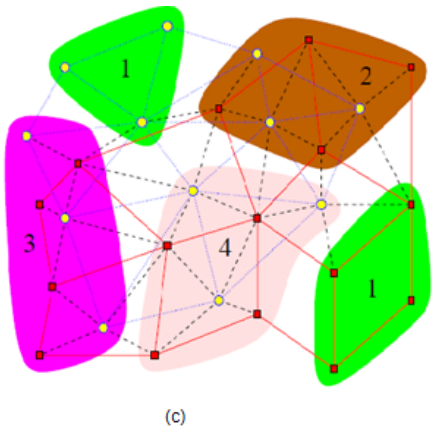


Figure 9: Partitioned network (Figure 6)

Multi-constraint graph partitioning

The problem is defined as follows:

A single surface is enough to split three connected areas in half in a three dimensional space. (h and J 1994) demonstrated that the theory is also applicable for graphs consisting of nodes that are tagged with random weights. Recent studies in this section have developed extended versions of some schemes which differ in complexity and their problem-solving scale.

A suitable scheme for multi-phase computation that modifies the traditional graph partitioning to some degree is to gradually partition disjoint subsets.

For this matter

1. Nodes are grouped based on the phase of the computation in which they are activated.
2. Grouped nodes are locked.
3. Free nodes will be partitioned with respect to the locked subsets. This means highly connected nodes to the locked subset are likely to be assigned to the same subsets as their neighbors.

(Karypis and Kumar 1998) Present a more general scheme for a wide range of complex multi-phase, multi-physics, and multi-mesh simulations in which the key factor for their efficiency is the initial partitioning algorithm.

It demonstrates that a set of *m*-weight objects can be partitioned into *m* disjoint subsets in a way that the maximum difference between weights of the sets is limited to twice the maximum weight of any object.

A parallel formulation of this scheme presented by (Schloegel, Karypis and Kumar 1999) efficiently computes partitioning for large scale graphs.

Multi-objective graph partitioning

Contrary to traditional methods, in the case of multi-objective schemes the optimal solution is presented for each objective and not as a general solution.

1) The minimum optimization values are determined
2) Greater values for a single object are considered
3) The result is then projected on the next object

Note that:

- The scheme considers independent solutions namely pareto -optimal points whether they are optimal for other objectives or not.
- Pareto-optimal points are those that improve the value of an objective without negatively impacting the value of other objectives.
- Multi-objective optimization consists of many pareto- optimal points.
- The user is to specify and adjust the area of interest among the pareto frontier. Nevertheless allowing user to offset different objectives that are substantially different in nature is challenging.
- Clarifying the area of interest among the pareto frontier will lessen the ambiguity of the desired solution.
- (Schloegel, Karpis and Kumar 1999) Provides a new method for multi-objective graph partitioning utilizing traditional methods. The solution translates the trade-offs between objectives in a user-specified preference vector.

3.2.2.2. Comparison of graph partitioning techniques

It is noteworthy to mention that graph partitioning schemes differ in the edge-cut quality

produced, run time, degree of parallelism, and applicability to certain kinds of graphs. The main problem in all schemes is that sufficient data on the edge-cut quality and run time for a common pool of benchmark graphs is not available. (Schloegel, Karypis and Kumar 2006)

The best way to obtain practical partitioning results with respect to the problem is to take advantage of multiple schemes by consolidating them in such a way that the overall quality is improved. For instance the combination of geometric method and KL/FM refinement that will lead to fast and high quality decomposition. Or multilevel techniques perform better when executing LND or spectral method on the original graph and improving it through KL/FM algorithm.

Furthermore here we provide a graphical display in table 1 and 2 concerning qualitative comparison of static graph partitioning schemes comparing the abovementioned features. Also table 3 provides a comparison between common and well known techniques regards adaptive schemes.

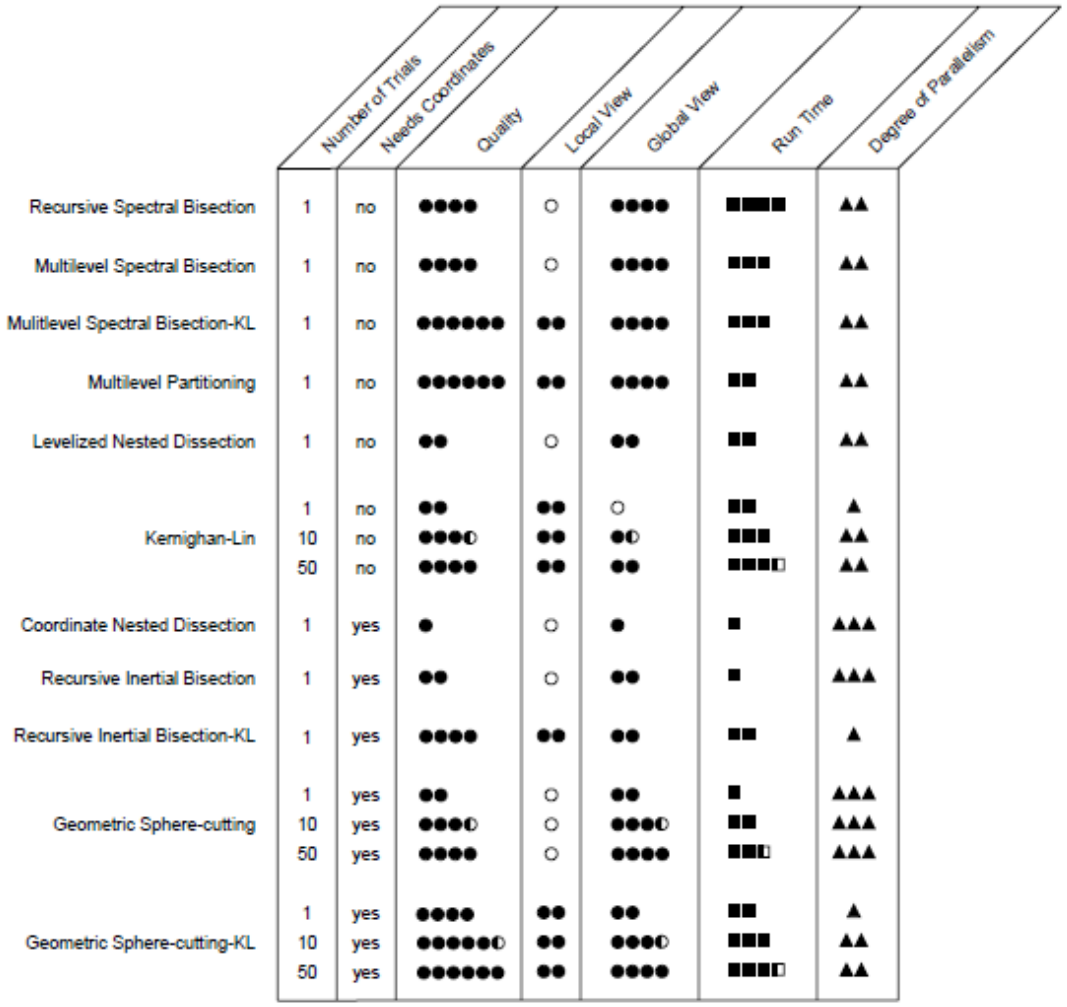


Table 1: Graph partitioning schemes rated based on run time, degree of parallelism and related characteristics

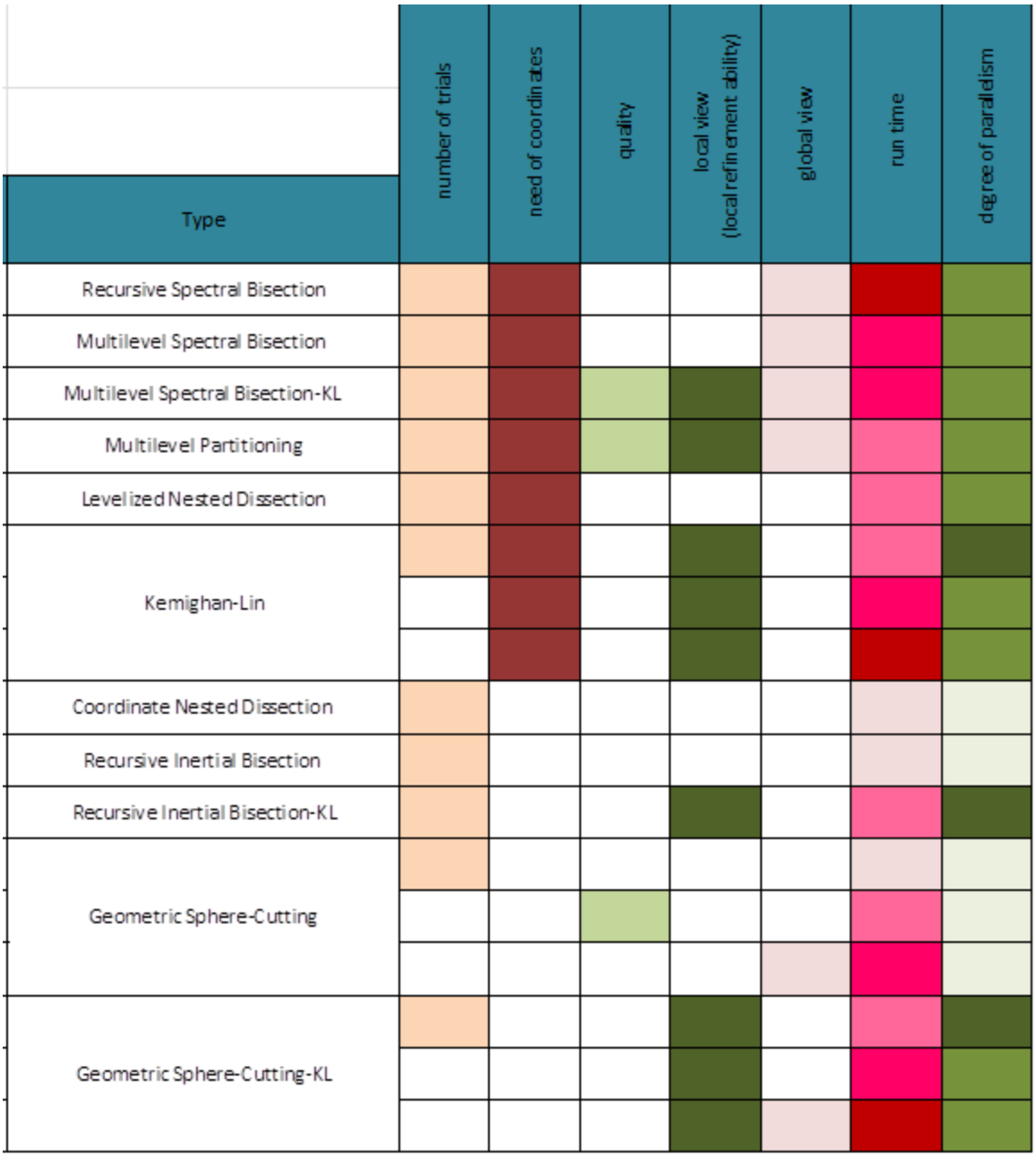


Table 2: Qualitative comparison of graph partitioning schemes

The first column shows the number of trials required to find a good quality partition. Here uncolored cells indicate the necessity of more than one trial. Colored cells in the second column show the need to know the coordinates in order to solve the problem. Colored cells in the third column indicate high-quality partitioning. The quality of different schemes is assessed by

taking global and local view into account. The next two columns provide information in this regard; where colored cells in both columns indicate their ability in maintaining such requirement. Consequently as shown in table 1, algorithms that have colored cells for both characteristics provide higher quality decomposition compared to the rest. Next column provides information on algorithm's runtime. Lighter shades imply faster schemes with little execution time. For instance rows 9, 10 and12 have the lowest computational time compared to the rest. Finally the last column indicating the level of parallelizability where darker shade implies that the method is highly sequential and incapable of parallelism.

As illustrated in table (1) we can conclude that multilevel graphing is one of the good types of partition among all since

- Its number of trials is low.
- There's no need to collect coordinate information associated with the nodes.
- It maintains high quality edge-cut compared to other methods.
- It has the ability to provide both local and global view.
- It executes on a somewhat moderate run time.
- Its capability in exploiting parallelism effectively.

As shown in table 3 diffusion based schemes are known to be very promising as the quality is high when the imbalance happens globally and scratch remap scheme is known to be practical where imbalance occurs locally.

Adaptive schemes	Quality	Speed	Amount of data movement
Diffusion based repartitioners	high	slow	low
Scratch-Remap repartitioners	High for local imbalanced areas Low for globally imbalanced areas	slow	high
Cut and paste repartitioning method	low	fast	low

Table 3: A summary of adaptive schemes

Limitations of graph partitioning approaches

- The edge-cut metric is neither an accurate model for inter processor communication costs incurred by parallel processing nor the total communication volume.
- Min-cut formulation is effective for the well-shaped meshes but not for non-uniformed meshes.
- (Catalyurek and Aykanat 1999)Developed a hyper graph partitioning formulation able to run on more general cases.
- Strength and weaknesses of the scheme are as follows
    - Able to minimize communications volume. Nevertheless the start-up time or the time required for the processor with the most communication remains the same.
    - Decreases the inter-processor communication costs for graphs of non-uniform degree. However this does not apply for graphs of uniform degrees.
    - The standard graph partitioning scheme is only capable of modeling square and symmetric sparse matrices.

- Solving linear systems, least squares problems, and linear programs require rectangular and unsymmetrical matrices. For this matter we can address the problem effectively through schemes such as bipartite graph partitioning and multi-constraint graph partitioning. Minimizing the edge-cut of a partitioning doesn't guaranty the numerical scalability of an iterative method.

- Numerical scalability is a measurement tool indicating the performance of a graph decomposition scheme for parallel computation. Numerical scalability promise to keep the convergence rate of the iterative procedure constant regardless of the number of subsets, processors or the size of the network. More specifically computational resources required to solve a problem of a given size are related to the problem size in linear way.
- Numerical scalability is maintained if only subsets have low mean aspect ratio.
- Walshaw, Cross, Diekmann, and Schlimback's scheme sustain a low mean aspect ratio but lack minimum edge-cut property.

Architecture modeling limitations

In the case of parallel computational models, we take advantage of traditional graph partitions on the assumption that a flat and homogeneous structure is our ultimate objective.

In the matter of Meta –computing environments where estimating the type of the machines and the exact number of processors for execution is indecisive until the execution time; Heterogeneous and hierarchical architectures have come to attention.

Software packages

Currently, software packages are available for both static and dynamic graphs. The purpose of using such packages is to save effort and time of the organization as well as enabling a

comparative assessment of different schemes when applied in the system. Table 4 presents available software packages

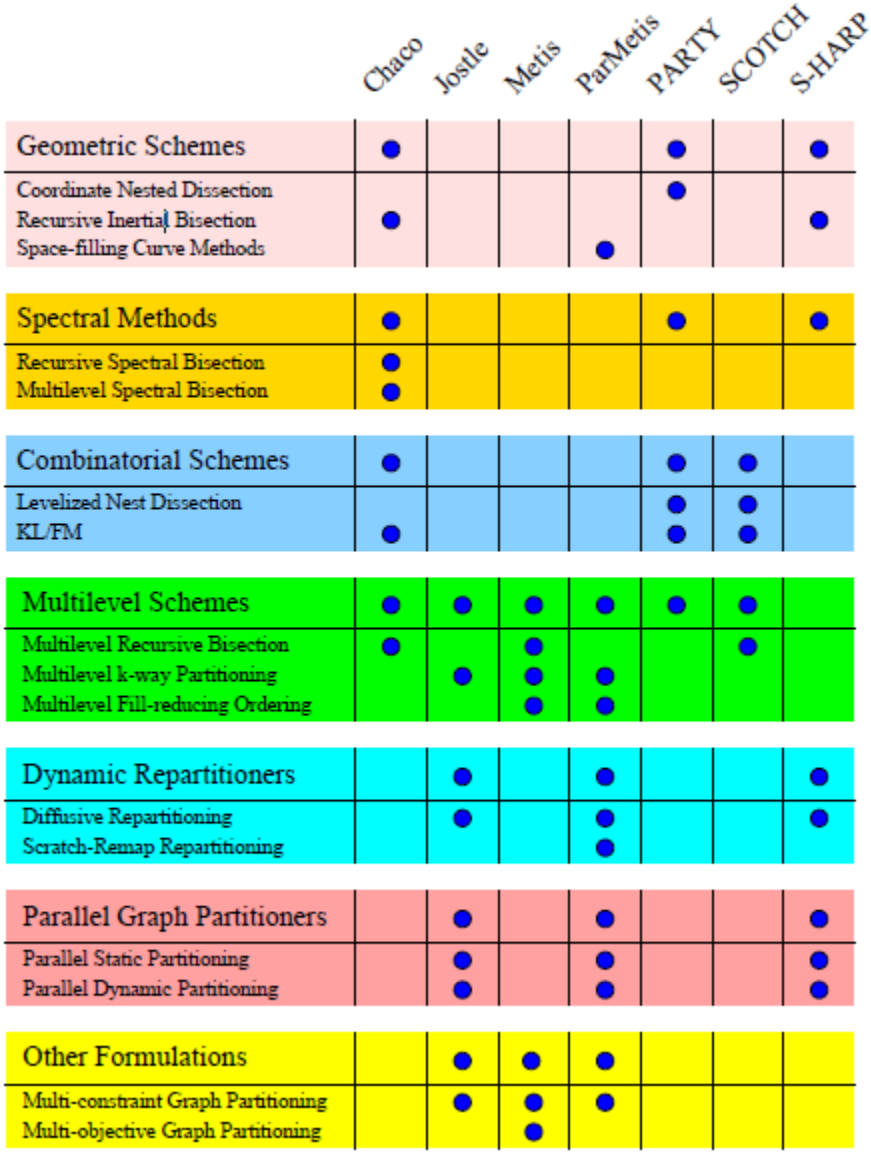


Table 4: Available software packages

3.3. Graph partitioning under disruption

Traditional models presented in the literature review assume that the network never fails to perform and its properties (nodes and edges) are defectless; while in real life supply chain networks, especially large scale networks that are expanded globally can face irregularities in their activities due to the occurrence of unexpected events such as economic crises, intentional attacks such as war and labor strikes, natural disasters such as hurricane and unseen weather conditions, power failure and machine breakdowns ,shortage of parts and materials ,quality rejection and corruption in transportation system, etc. These factors contribute to indirect and direct financial loss throughout the system, unreliability in performance; influencing the operational environment and inflicting damage to the system. However, some possible threats with high probability are anticipated and identified in advance by studying past records and their historical frequency. These set of threats can be dealt with resource concentration in potential areas as well as fortifying assets towards first, damage resistance and second, low degree of severity in the case of disaster. Nevertheless the effect of an event is very relevant to the point where it occurs as critical nodes and cardinal links known as bottlenecks in a network happen to create the greatest damage compared to non-critical ones. Therefore besides identifying potential areas vulnerable to certain threats it is also in the system's interest to identify critical elements and develop a link/ node protection scheme to increase their tolerance to shock and additionally provide comparable alternatives for extreme events. Nevertheless, in practice, it is not possible to completely rule out the occurrence of an unseen event thus, depending on the magnitude of the event the normal flow of the system is interrupted followed by its resulting consequences.

In the sense of network decomposition which is the scope of this research; a well-structured topology has a major role not only on the practicality of the resulting partitions but also the system's functionality under a certain level of damage and its ability to quickly replace or repair a broken link /node and bounce back. Thus models such as hierarchical, random and heterogeneous structures intend to return different results and will react differently to disaster.

In this study, other than our initial objective that is partitioning a network into independently manageable groups as well as re-clustering the network under any type of disruption, we aim to take out our suggested methods while sustaining connectivity between clusters through a set of commands. It is important to understand that a critical element in a network is not necessarily critical in graph partitioning, meaning that a node and its related links may be positioned in a critical state when clustered regardless of its state in the network. That is why maintaining resiliency in the actual bottlenecks as well as virtual bottlenecks of the network comes to attention as it is enables the system to proceed activates and services in spite of breakdown in entities. In order to avoid confusion for the reader we will refer to these virtual bottlenecks as damage sensitive elements. A damage sensitive node by definition is a node that contains a pair in more than one cluster and whose removal will lead to disjoint components and significantly degrades the performance of the network. Damage sensitive links are simply ones who exits or enters a damage sensitive node.

Contributing towards a better understanding of the importance of such investment in the system a comparative study is gathered and we will demonstrate both conditions where the method re-clusters the disrupted network with and without system control and protection of damage sensitive elements.

But before going any further studying post-actions in an actual disrupted network also referred to as “corrective actions”; we will take time addressing the essential components required in the underlying structure of a network in favor of a resilient and reliable network.

To start with, Resiliency has been a controversial topic as some studies insist on it to be solely relatable to post actions that are done for the system to go back to its pre-event state after an incident takes place; while others define it as combination of pre and post actions implemented in order to withstand shock and maintain performance meanwhile recovering from shock.

(Riskviews: Commentary on Risk and ERM 2013) Posted an article titled as “Five components of resilience -- robustness, redundancy, resourcefulness, response and recovery”. The post was "adapted from the “WEF Global Risks 2013 Report”

The state of resiliency in an organization is determined through the first three components mentioned in the title. Robustness, Redundancy and Resourcefulness are known as resilience capacity indicators and should be designed in the infrastructure of a system so that the inherent resilience capabilities of an organisation are assessed by them.

The article also states that Response and Recovery are two components known to evaluate the performance of a resilient system in the face of crisis and are subject to risk, event and time. More specifically Response and Recovery are considered as calibers for resilience characteristics. Here we take time to define each along with the attributes associated with them.

"Robustness is a factor that indicates reliability and is when the system is composed of standardized units and is equipped with fail-safes and firewalls at its vulnerable areas also critical points in order to efficiently deal with crisis instead of blocking the flow of information and stopping system’s progress. In other words a potential damage in one section is less likely to spread and expand around with pre-actions such as:

- Regular update and evaluation on system's health condition.
- Equipping system with mechanisms that are specifically designed to localize the impact of a problem and prevent unexpected attacks spreading to other sections.
- Utilizing adaptive decision-making models help system to continue its operations in times of crisis.

That is to say, robustness provides consistency in operations with respect to the nature of the business, the scale of operations performed in the organization and the system's response to a disconnection between sections.

"Redundancy comes with investing in on an alternative plan including a diverse range of policies, strategies and services. Network redundancy requires spare capacity and diversity in routes from an origin to its destination. Contrary to the general belief a redundant system brings back the initial investment and will save the organization in the long term and shouldn't be considered a dead loss. Most efficient attributes concerning this characteristic is as follows:

- Redundancy of critical infrastructure: to pinpoint critical components and create identical copies for them in the case of disruption.
- Diversity of strategy and solution: to develop alternative solutions for certain operations will allow the system to continue its activities though disrupted. In brief a balanced combination of diversity and redundancy is an important factor for efficiency.

"Resourcefulness is about asset and property optimization. It concerns system flexibility and adjustment to the aftermath of disruption, adapting to new circumstances and turning a negative situation into a positive form. This requires certain attributes namely:

- Testing both system's response plan and the individual's reaction to crisis in simulated exercises are necessary to maintain system's ability to resolve problems in real-time.
- Building trust within the network structure starting from the staff and working environment up to the consumer and providers, developing a self-organized system determined to solve problem in the face of unpredicted challenges.
- Self-organizing attitude: a skill that basically arises in the event of an abnormal situation requiring factors such as social and human capital[2], a strong social network and organizational structure, effective communication and an open attitude to exchange of information ideas and experiences.
- Establishing an atmosphere enhanced with creativity and innovation which is influenced by the state of the networks backup resources and the defined set of rules and disciplines.

"Response is the readiness of the system to react immediately in order to withstand the extent of damage. In other words the ability to notice and take action on any abnormal behavior by quickly sharing relevant information to related units through useful methods of gathering and distributing information. Attributes that can strongly minimize response time are

- Conducting an effective communication by building mutual trust. This will accelerate the flow of information in crisis and will ensure a two-way cooperation. Communication can also become more germen by familiarizing participants to informal communication channels so that they will be more coordinate when interacting in the time of crisis.

[2] Social capital: The networks of relationships among people who live and work in a particular society, enabling that society to function effectively.
Human capital: The skills, knowledge, and experience possessed by an individual or population, viewed in terms of their value or cost to an organization or country.

- Global participation among all unities is required so that potential factors creating crisis are eliminated as much as possible.

Recovery plays a great role in system's resiliency and comes with flexibility and tolerance.

Recovery is for the system to move back to some point of its normal behavior and adjust itself to the new situation after an incident. This component works by assessing systems strategies in transmitting necessary data so that decision makers are well informed to take forward actions required for network's adjustment to the new circumstance. Recovery is characterized with attributes such as

- Putting horizon scanning techniques in to use will not only brings gaps and inabilities into attention but also help detecting potential opportunities to enhance system performance.
- Building up a responsive feedback mechanism that empowers an organization with the ability to capture lessons learnt and translate information gained from horizon-scanning into practical, effective solutions.

Furthermore, in order to complete an efficient performance thoroughly, operators should regularly monitor the status of the network to detect any probable abnormality such as traffic conditions, link damage and security failure caused by network equipment deficiency or failure in satisfying basic agreed upon transmissions. In addition some effective inference techniques are necessary to acquire information for further network assessment. This in turn has a major role in preventing potential disruptive events such as natural disasters or unexpected attacks from happening.

The abovementioned characteristics are necessary to maintain an ideal system. Nevertheless based on our foregoing discussion we have acknowledged that no matter how hands-on and anticipated a system operates, there are some unseen flaws or accidents that will unfortunately cause disruption. However previous studies in this regard, as we will discuss in details in this section, have mainly focused on strategy adjustment and flexibility in order to decrease the chances of accidents. One of our goals in this thesis, as previously stated, is to create a set of corrective actions capable of resolving the occurrence of unpredicted incidents in the concept of graph partitioning and decomposition. That is re-clustering the network into a new set of clusters while satisfying node and link weight constraints.

A disrupted network by definition is when a network fails to accomplish an end to end communication. In particular, a challenged network is the consequence of an external disturbance that leads to changes in network topology where both links (communication paths) and nodes (entities) can encounter trouble performing their original service. In terms of network decomposition, the extent and impact of an event can be limited to one cluster and cause damage among intra-connected nodes or may exceed and spread throughout other clusters and affect inter-connections creating partial/complete disruption in the network. Previous works in various fields of science have been assigned to Predicting and diagnosing abnormalities in the system and taking action in advance to avoid significant disaster that can substantially demolish the system's performance and operation. Subsequently, Network vulnerability assessment and evolutionary methods have been widely studied towards understanding the nature of the network and its strengths and weaknesses in favor of formulating preventive actions mainly to increase networks resilience when facing disaster. This happens to be more evident in complex and noisy transportation networks where network planning and risk management is vital for

network functionality. (Snyder, et al. 2005) (Inference of Network-Service Disruption upon Natural Disasters n.d.)

In this regard (Erjongmanee, et al. 2008) provides an application capable of inferring network-service disruption upon natural disasters using sensory measurements and human inputs as well as analyzing the disruption caused by natural disasters. (Huang, et al. 2007) Suggest a network-wide analysis in routing information in favor of detecting and identifying network disruption. For this matter it applies a multivariate analysis technique on dynamic routing information and demonstrates how this technique can detect every reported disruption on nodes/ links within the network with a low percentage of deception. It investigates the type of disruption and the scenario that has created disruption through analyzing both network-wide static configuration and its dynamic routing updates. (Dinh, Xuan, et al. 2009) Believe that the pre-active assessment over network vulnerability in the light of connectivity matrix is essential to aspects such as design and maintenance of any infrastructure network such as communication, commercial, and social networks. It investigates a measure called pairwise connectivity that seeks to discover critical nodes/links and safeguard them against destructive case scenarios where the overall network connectivity is at risk. More specifically it generates this vulnerability problem as a new graph-theoretical optimization problem called β-disruptor, whose removal brings in the highest damage of the global pairwise connectivity by destroying a set of interacting nodes or connections that can cause a whole network breakdown.

(Snyder, et al. 2005) Holds account for additional investments in planning and designing the network .It concedes that depending on the possible financial resources, planners need to take risk of disruption into consideration by identifying and eliminating all threatening elements

prior to system failure due to difficulty in changing a strategic decision quickly or prescribing an efficient remedy for after destruction.

(Sterbenz, et al. 2012) Represent an analytical framework with the purpose of evaluating network resilience based on a two-dimensional state space, namely operational state and service delivered. It provides a flexible framework for an n-level hierarchical, modular structure and simulates the problem by a model called KU-CSM. (Hua, Ding and Shao 2008) Suggests a multi-criteria model for a periodic balanced partition that takes into account a pareto partition for a large-scale network via constructional partitioning and cooperative searching. (Drid, et al. 2010) Handles the problem of survivability via data banking and backup resources and solves network complexity and scalability by building a protection- based solution on a well-established p-cycle concept. (Dinh, Xuan, et al. 2009) Investigate a measure called pairwise connectivity. It solves the problem by generating a new graph-theoretical optimization called β-disruptor that finds a set of critical nodes and links, whose removal will results in the maximum change in the global pairwise connectivity. It claims that building a good communication among connected nodes could reduce potential disruption and preserve the infrastructure of the network.

Chapter 4:

Solution Approach

In this chapter we present our proposed solution for each set of the problem stated in chapter 2. We translate the actual problem into a node graph; as it provides a more coherent view of the problem and can be easily and rapidly constructed. Figure 7 looks at the solution methodology from a flow chart view. The first step would be to define the input graph. In our work, our modeled graph has a directed structure with a positive integer number assigned to each edge denoting weight. Nodes represent entities and links indicate their relationship according to the characteristics and nature of the problem. In the second stage the original graph is disrupted by means of various disruption scenarios. In the third step we apply our proposed methods and in the final step 4, our new re-clustered graph is generated.

In the following subsections we discuss our scenarios regarding disruption followed by a descriptive explanation of each one of the proposed techniques.

4.1. Scenario generation for disruption modeling

Identifying basic requirements are essential when developing a scenario based framework. In the case of disruption our model must determine the state of disruption. In this sense, it is necessary to detect whether we are dealing with a temporary or a permanent disruption. Note that here monetary consequences are excluded. Further the assessment of the nature of disruption is important as each one leads to a different solution plan. For this reason, network disruption is introduced as occurring in nodes or in the connection between nodes. Furthermore

the type of a disrupted link must also be distinguished as it can either be internal or external to the subgroup (intra-connection and inter-connection).

Given these complexities, in this study; we have defined three different categories for an operational network status namely complete disruption, partial disruption and no disruption. A complete disruption occurs when any of the elements of the network is permanently damaged and is no more available. Partial disruption is when system disruption is temporary and the failure has not led to complete loss of component parts. For instance some parts may shut down after system sees an event due to lost access to data but will come back into service once an alternative source and in some cases a transportation link is found. In other words, the link or node failure is repairable within a timeframe. Finally, no disruption is where our static graph applies and no node or link disruption takes place.

We generate nine possible scenarios considering the state of the network; namely:

1. No Disruption in the Network
2. Complete Disruption in Links
3. Complete Disruption in Nodes
4. Complete Disruption in Nodes and Links
5. Partially Disrupted Links
6. Partially Disrupted Nodes
7. Partially Disrupted Links and Nodes
8. Complete Disruption in Nodes and Partially disrupted Links:
9. Complete Disruption in Links and Partially Disrupted Nodes:

Each case has been individually studied and investigated by a specified percentage of disruption, considering any more than 40% disrupted nodes as a complete disruption. Nevertheless, in accord with accomplishing our purpose that is handling disruption in the most efficient way possible the best strategy would be to consider and signalize areas of vulnerability in favor of building up the infrastructure with key factors of resiliency. Therefore as discussed earlier we will be demonstrating each scenario for each method and its subclass, that is in consideration of the check and control step.

4.1.1. Scenarios description

Based on the information and previously discussed requirements, our proposed models define two set of disruptions namely complete and partial disruption. The algorithms work on the following principles.

As for an interrupted component when completely stated; network refinement is done by updating the corresponding elements of connectivity matrix *x* to zero. Accordingly, when link disruption occurs, broken links are solely updated to 0. On the other hand if node disruption occurs, all links associated with the broken node are taken out and updated to 0. Next step is to apply each one of our algorithms to the updated network and obtain a new partitioning scheme for the network.

The mechanism of the algorithm in a partial disruption can be briefly explained as follows. The weighted matrix (on a scale of 1 to 10) is taken into account and troubled links /nodes are explored and accordingly their value is adjusted to a greater number indicating delay. $HRP1^{+}$ and $HRP2^{+}$ are then applied to the updated matrix. More specifically when the program is exploring for set of nodes to combine the low weight elements are selected before high weight elements signifying extra time and ultimately resulting in network refinement.

In the case of our illustrated numerical example, we have considered adding a fixed quantity specified as 5 units to any misbehaved element. However the model is capable of assigning random values to each troubled component with respect to its constraints.

Additionally as mentioned previously since the algorithm randomly disrupts the network there is always a chance of selecting a damage sensitive component; therefore we have developed the program to identify nodes that are critical in the ultimate set of clusters and provide an alternative that functions just as the original component. More specifically if the attacked component is damage sensitive the program returns one link of higher priority back into the network and updates matrix *x* accordingly and then $\mathrm{HRP1}^{+}$ and $\mathrm{HRP2}^{+}$ are applied to the updated matrix.

Here we will explain each scenario along with the corresponding algorithm.

1. **No disruption in the network:**

This scenario indicates that system is steady and no disruption has occurred, therefore $\mathrm{HRP1}^{+}$ and $\mathrm{HRP2}^{+}$ result in the original partitioning without disruption.

2. **Complete disruption in links:**

This scenario indicates that system has encountered a permanent damage in the connections; hence the corresponding algorithm is used:

Get *n*

Find all the links in [*X*]

Randomly disrupt *n*% of the links

**Check *cric*

Update [*X*]

Define [*W*] equal to [*X*]

Get [*X*], [*W*] (input)

Run $HRP1^+$ or $HRP2^+$ (($HRP1^+$ *control* or $HRP2^+control$))

**Note that check *cric* command is only executed when the check and control step is included in the program and the extended version of our methods are running.

3. **Complete disruption in nodes:**

This scenario implies that system has encountered a permanent damage in nodes; thus the corresponding algorithm is used:

Get *n*

Randomly select *n*% of the nodes (*N*):

Disrupt all related links to the chosen nodes in [*X*]

**Check *cric*

Update [*X*]

Define [*W*] equal to [*X*]

Get [*X*], [*W*](input)

Run $HRP1^+$ or $HRP2^+$(($HRP1^+$ *control* or $HRP2^+control$))

**Note that check *cric* command is only executed when the check and control step is included in the program and the extended version of our methods are running.

4. **Complete disruption in nodes and links:**

This scenario shows permanent damage in both links and nodes and the corresponding algorithmic solution is as follows:

Get *n*

Find all the links in [*X*]

Randomly disrupt *n*% of the links

**Check *cric*

Update [*X*]

Randomly select *n*% of the nodes (*N*):

Disrupt all related links to the chosen nodes in [*X*]

Update [*X*]

Define [*W*] equal to the updated [*X*]

Get [*X*], [*W*] (input)

Run HRP1$^+$ or HRP2$^+$((HRP1$^+$ *control* or HRP2$^+$$control$))

**Note that check *cric* command is only executed when the check and control step is included in the program and the extended version of our methods are running.

5. **Partially disrupted links:**

This scenario implies that system has encountered a repairable damage in links and allegedly, tasks are accomplished with delay; hence the corresponding algorithm is presented as follows:

Define [*W*] equal to [*X*]

Find all the links in [*X*]

Randomly select *n*% of the links

**Check *cric*

Find the corresponding arrays for the selected links in [*W*]

Set the value of the weight equivalent to the value of disruption in the links

Update [*W*]

Get [*X*], [*W*](input)

Run HRP1$^+$ or HRP2$^+$((HRP1$^+$ *control* or HRP2$^+$$control$))

**Note that check *cric* command is only executed when the check and control step is included in the program and the extended version of our methods are running.

6. **Partially disrupted nodes:**

This scenario denotes temporary damage in nodes and the corresponding algorithm used is

presented as follows:

Define [W] equal to [X]

Randomly select n% of the nodes (N)

**Check *cric*

Find the correspondent arrays for the selected nodes in [W]

Set the value of the weight equivalent to the value of disruption in the nodes

Update [W]

Get [X], [W] (input)

Run HRP1$^+$ or HRP2$^+$((HRP1$^+$ *control* or HRP2$^+$$control$))

**Note that check *cric* command is only executed when the check and control step is included in the program and the extended version of our methods are running.

7. **Partially disrupted links and nodes:**

This scenario shows that system is facing a repairable damage in both links and nodes. Thus, there's communication and computation delay. Subsequently, the corresponding algorithm used is presented as follows:

Define [W] equal to [X]

Find all the links in [X]

Randomly select n% of the links

**Check *cric*

Find the correspondent arrays for the selected links in [W]

Set the value of the weight equivalent to the value of disruption in the links

Update [W]

Randomly select n% of the nodes (N)

Find the correspondent arrays for the selected nodes in [W]

Set the value of the weight equivalent to the value of disruption in the nodes

Update [*W*]

Get [*X*], [*W*] (input)

Run HRP1$^+$ or HRP2$^+$((HRP1$^+$ *control* or HRP2$^+$$control$))

**Note that check *cric* command is only executed when the check and control step is included in the program and the extended version of our methods are running.

8. **Complete disruption in nodes and partially disrupted links:**

This scenario implies that system is dealing with a permanent damage in some nodes; meanwhile links are facing a temporary conflict. HRP1$^+$ and HRP2$^+$contribute to the problem by the following algorithm:

Get *n*

Randomly select *n*% of the nodes (*N*):

Disrupt all related links to the chosen nodes in [*X*]

**Check *cric*

Update [*X*]

Define [*W*] equal to [*X*]

Find all links

Randomly select *n*% of the links in [*X*]

Find the corresponding arrays for the selected links in [*W*]

Set the weight value equivalent to the disruption value of the links

Update [*W*]

Get [*X*], [*W*] (input)

Run HRP1$^+$ or HRP2$^+$((HRP1$^+$ *control* or HRP2$^+$$control$))

**Note that check *cric* command is only executed when the check and control step is included in the program and the extended version of our methods are running.

9. Complete disruption in links and partially disrupted nodes:

This scenario indicates that system is facing a permanent damage in links while some nodes are temporarily dysfunctional. HRP1$^+$ and HRP2$^+$contribute to the problem by the following algorithm:

Get *n*

Find all the links in [*X*]

Randomly disrupt *n*% of the links

**Check *cric*

Update [*X*]

Define [*W*] equal to [*X*]

Randomly select *n*% of the nodes (*N*)

Find the correspondent arrays for the selected nodes in [*W*]

Set the weight valuer equivalent to the disruption value of the links

Update [*W*]

Get [*X*], [*W*] (input)

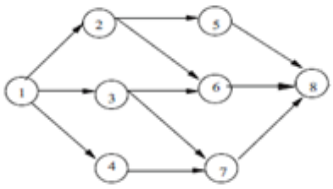
Run HRP1$^+$ or HRP2$^+$((HRP1$^+$ *control* or HRP2$^+$$control$))

**Note that check *cric* command is only executed when the check and control step is included in the program and the extended version of our methods are running.

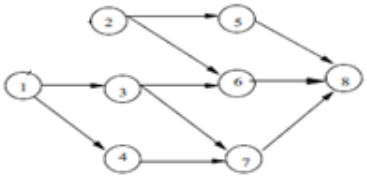
4.2. Scenario generation

We have considered a network of 8 nodes and 11 links shown in figure (10) to demonstrate the type and level of disruption. Figure (A) shows a network in its original state pre-event, figure

(B) and (C) display a complete link and node failure respectively. In figure (B) link 1-2 is taken out due to disruption, while figure (C) illustrates a situation where node 4 has been attacked therefore associated links with node 4, link 1-4 and 4-7 are also removed. Figure (D) and (E) present partial disruption. In figure (D) link1-2 is dashed representing a delay while figure (E) is showing a temporary damage in node 4 resulting delay in its associated links, 1-4 and 4-7, as well.

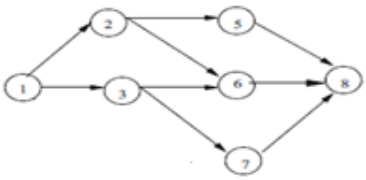


(A)



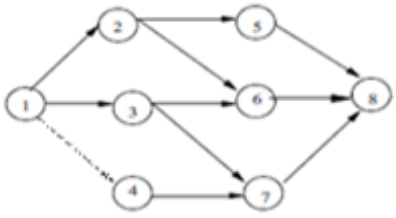
B

Link 1 to 2 is removed



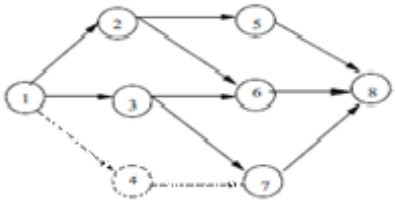
C

Node 4 and its associated links are removed



D

Link 1 to 4 is dashed due to partial disruption



E

Node 4 and its associated links are dashed

Figure 10: A network of 8 nodes and 11 links

In the next section, we will separately define the proposed methods and their extension. In each subsection, we will brief the reader with the procedure as well as providing the related pseudo-code to our model. Further, our previously discussed scenarios have been implemented separately displaying results at the 5 percentage of disruption probability level. We have also seen a stepwise computational procedure embodied in each scenario for when the network is under 5% disruption. In the following subsections we will demonstrate each one of our basic models as well as their extended versions provided in relative Tables.

4.3. Proposed partitioning approaches

Here we report on an improved version of HRP1 and HRP2 algorithms namely HRP1^+ standing for Hierarchical Recursive Progression1^+ and HRP2^+ an abbreviation for Hierarchical recursive progression2^+. (Awasthi, et al. 2009) . HRP1 and HRP2 are two heuristic approaches that fall into the combinatorial class algorithms and are performed on undirected and unweighted static graphs with the objective of decomposing a large network into subgroups of limited size while minimizing the inter-connection between these divided groups and maximizing the intra-connection so that the workflow among subgroups is reduced.

HRP1^+ and HRP2^+ are two extended version of the previously established algorithms capable of performing on positively weighted directed graphs. The techniques also attempt to reduce unnecessary communication between clusters while increasing data distribution and communication inside each cluster, in favor of operational efficiency and cost management. In addition they are modified to overcome unexpected incidents and changes in the underlying graph due to disruption by two adaptive computation approaches namely “complete failure update “and “partial failure update”. More specifically we have embodied a scenario based

model, discussed in the previous subsection, for the second stage of the execution that performs re-clustering on the disrupted network.

4.3.1. HRP1$^+$ Partitioning Technique

1. Description

The HRP1$^+$ technique is a cluster-based approach to graph decomposition that is capable of dealing with delay and disruption.The algorithm is based on the minimization of the minimum density criteria and is generated as follows:

HRP1$^+$ algorithm starts off with a single node in each subset. Therefore, the initial number of subgroups is equal to the total number of nodes in the network. HRP1$^+$ attempts to find a pair of nodes that can be combined as one by exploring all connected pairs with a size less or equal to *W* and a density greater than or equal to *Q*. A node-set with the lowest density among all other possible combinations is subsequently selected. Nevertheless the crucial point in finding the best result is the appropriate choice in defining *W* and *Q* for the studied network. (Awasthi, et al. 2009) . HRP1$^+$ algorithm is a recursive method and ends when each subset satisfies all constraints. In other words, completion is achieved when all subsets have a size less than or equal to *W* and/or a density lesser than or equal to *Q*.

2. Pseudo-code

The following inputs are the main variables of HRP1$^+$:

N: total number of nodes

i,j: connected nodes in the network

w_{temp} *(i):* total number of nodes in a subgroup

q_{temp} *(i)* : subset's density in each instance

Q: maximum density allowed in a subgroup (user-defined)

W: maximum nodes in a subgroup (user-defined)

C (r_k): total external connections in a subgroup defined by external nodes

card (r_k): cardinality

q (r_k) = *C* (r_k)/ *card* (r_k): externally connected nodes over cardinality

neigh_opt_1: (nop1), *neigh_opt_2: (nop2)*, are temporary variable set to achieve the optimum combination.

r_n: A vector of size *n* where components represent connected nodes to node *i*

The algorithm starts with *N=K, K* defines the number of clusters at each instant. In other words, the number of subsets in round one is equal to the total number of nodes.

A high level scope of our method comes as follow:

Input:

K: number of clusters

x: a matrix of links(dynamic property)

W and *Q*; user defined variables

Output:

Final set of generated clusters

1. Initialize *K* cluster
2. While termination condition is not satisfied do
3. Assign nodes to the clusters based on density constraint and node count limit in each cluster
4. Update cluster
5. End while

6. Detect damage sensitive elements (when applying HRP1$^+$ control)
7. Run disruption scenarios

A low-level stepwise code of our algorithm is as follows:

0: n=1

1. While (max (*card*) <= *W* || max (*q*) >= *Q*)
2. *K* = length(*x*);
   2.1. For *i=1: K* calculate
      2.1.1. r_n equals *all links with one end in node i* and then *c(i)* equals *length(find(r_n is not equal to 0));*
      2.1.2. *q (i)=c(i)/card(i);*
      2.1.3. q_{temp}*(i)=q(i);*
      2.1.4. End For
   2.2. For all *i* from 1 to *K*-1
      2.2.1. For all *j* from *i*+1 *to K*
      2.2.2. If ((*x(i,j)* equal to 1 or *x(j,i)* equal to 1) and (*card(i)+card(j)*) less than or equal to *W* and *q(i)* greater than *Q* and *q(j)* greater than *Q*)
         2.2.2.1. q_{temp} *(i)*= merging (*i,j,x,card*);
         2.2.2.2. End If
      2.2.3. If (q_{temp} *(i)* less than or equal to q_{opt} *(i)*) and (w_{temp} *(i)* greater than or equal to *w(i,j)*)
         2.2.3.1. *nop (i)* = result *(j,* 1*);*
         2.2.3.2. *nop2 (i) = j;*

2.2.3.3. End If

2.2.4. End For

2.2.5. If (q_{opt} (i) less than q_{opt} max)

2.2.5.1. $nop1_2$ =*i*

2.2.5.2. *nop2 =nop (i);*

2.2.5.3. $nop2_2$ = *nop2 (i);*

2.2.5.4. End If

2.3. End For

3. If ($nop1_2$ equals to 0 or $nop2_2$equals to 0) terminate algorithm

Merge and cardinality update

4. $card_{new}(nop1_2) = card_{new}(nop1_2) + card_{new}(nop2_2)$;

4.1. $card = card_{new}$;

5. *n*=*n*+1 and go to 1

3. Numerical Example

We have considered a network of 24 nodes and 30 links as shown in figure (11) to investigate all scenarios. For this hypothetical network our user defined parameter *W* is set to 8 and *Q* is set to 0.125.

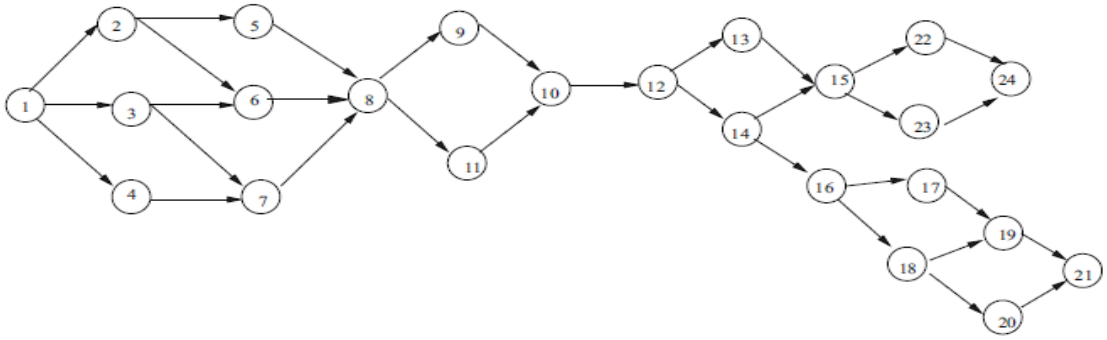


Figure 11: Network of 24 nodes and 30 links

Our network has been decomposed into 4 clusters shown in table 4.1.

No Disruption in the Network				
Clusters				**q**
1,4,7,3,6,2,5,8	9,10,11	12,14,16,17,18,20,21,19	13,15,22,24,23	.125,.6667,.25,.2

Table 4.1: No node disruption – No link disruption

The step wise procedure of computation for a network without disruption is shown in table 4.2.

The table displays the best three options of node consolidation at each iteration of the program. The algorithm chooses the node pair with the lowest density among the other options. For instance in row three of table 4.2, we see that the options for combinations are to

1) Unite node 1 and node 4

2) Unite node 16 to the previous combination that is the cluster containing node 18, 20 and 21

3) Merge node 19 with18, 20 and 21 that is the combination from the previous step

In this step since the W constraint is maintained it simply chooses node 19 over other choices as it allows a smaller value for Q.

Network with no disruption					
Number of iteration	1st best option	2nd best option	3rd best option	selected	*q*
1	1-4	20-21		20-21	1
2	1-4	18		18 joins 20-21	0.6667
3	1-4	16	19	19 joins 18-20-21	0.5
4	1-4	16	17	17 joins 18-20-21-19	0.2
5	1-4	16	-	16 joins 17-18-20-21-19	0.1667
6	1-4	14	-	14 joins 16-17-18-20-21-19	0.2857
7	1-4	12	-	12 joins 14-16-17-18-20-21-19	0.3750
(cluster 1)	12-14-16-17-18-20-21-19				0.2500
8	1-4	22-24	-	22-24	1
9	1-4	15	23	23 joins 22-24	0.3333
10	1-4	13-15	15	15 joins 22-24-23	0.333
11	1-4	13	-	13 joins 15-22-24-23	0.5
(cluster 2)	13-15-22-24-23				0.2000
12	1-4	-	-	1-4	1.5
13	7	-	-	7 joins 1-4	1
14	3	-	-	3 joins 1-4-7	0.75
15	6	-	-	6 joins 1-4-7-3	0.4
16	2	-	-	2 joins 1-4-7-3-6	0.333
17	5	-	-	5 joins 1-4-7-3-6-2	0.25
18	8	-	-	8 joins 1-4-7-3-6-2-5	0.1429
(cluster 3)	1-4-7-3-6-2-5-8				0.1250
19	9-10	-	-	9-10	1.5
20	11			11 joins 9-10	1
(cluster 4)	9-10-11				0.6667

Table 4.2: No node disruption – No link disruption

Table 4.3 shows results of the network with complete failure in links at the 5 percentage of disruption probability level.

Scenario 2: Complete disruption in links					
Number of iteration	1st best option	2nd best option	3rd best option	selected	*q*
1	1-4	17-19	20-21	20-21	1
2	1-4	17-19	18	18 joins 20-21	1
3	1-4	16	19	19 joins 18-20-21	0.75
4	1-4	16	17	17 joins 18-20-21-19	0.4
5	1-4	16	-	16 joins 17-18-20-21-19	0.3333
6	1-4	14	-	14 joins 16-17-18-20-21-19	0.4857
7	1-4	12	-	12 joins 14-16-17-18-20-21-19	0.5
(cluster 1)	12-14-16-17-18-20-21-19				0.2500
8	1-4	22-24	-	22-24	1.5
9	1-4	15	23	23 joins 22-24	0.6667
10	1-4	13-15	15	15 joins 22-24-23	0.75
11	1-4	13	-	13 joins 15-22-24-23	0.4
(cluster 2)	13-15-22-24-23				0.2000
12	1-4	-	-	1-4	2
13	7	-	-	7 joins 1-4	1.3333
14	3	-	-	3 joins 1-4-7	1
15	6	-	-	6 joins 1-4-7-3	0.6
16	2	-	-	2 joins 1-4-7-3-6	0.5
17	5	-	-	5 joins 1-4-7-3-6-2	0.2857
18	8	-	-	8 joins 1-4-7-3-6-2-5	0.3750
(cluster 3)	1-4-7-3-6-2-5-8				0.1250
19	9-10	-	-	9-10	2
20	11			11 joins 9-10	1
(cluster 4)	9-10-11				0.6667

Table 4.3: Clustering under 5% disruption

Table 4.4 shows results of the network with complete failure in nodes at the 5 percentage of disruption probability level.

Scenario 3: Complete disruption in nodes					
Number of iteration	1st best option	2nd best option	3rd best option	selected	*q*
1	1-4	20-21	23-24	23-24	1
2	1-4	15		15 joins 23-24	1
3	1-4	13		13 joins 15-23-24	0.75
4	1-4	12		12 joins 13-15-23-24	0.6
5	1-4	14		14 joins 12-13-15-23-24	0.5
6	1-4	16		16 joins 12-13-15-23-24-14	0.57
7	1-4	17		17 joins 12-13-15-23-24-14-16	0.5
(cluster 1)	12-13-15-23-24-14-16-17				0.2500
8	1-4	20-21	-	20-21	1.5
9	1-4	18		18 joins 20-21	1
10	1-4	19		19 joins 18-20-21	0.5
(cluster 2)	18-20-21-19				0.2000
11	1-4	-	-	1-4	2
12	7	-	-	7 joins 1-4	1.3333
13	3	-	-	3 joins 1-4-7	1
14	6	-	-	6 joins 1-4-7-3	0.6
15	2	-	-	2 joins 1-4-7-3-6	0.5
16	5	-	-	5 joins 1-4-7-3-6-2	0.2857
17	8	-	-	8 joins 1-4-7-3-6-2-5	0.3750
(cluster 3)	1-4-7-3-6-2-5-8				0.1250
18	9-10	-	-	9-10	2
19	11			11 joins 9-10	1
(cluster 4)	9-10-11				0.6667
(cluster 5)	22				0

Table 4.4: Clustering under 5% disruption

Table 4.5 provides the results of a scenario where the network is challenged with complete failure in nodes and links at the 5 percentage of disruption probability level.

Scenario 4: Complete disruption in links and nodes					
Number of iteration	1st best option	2nd best option	3rd best option	Selected	*q*
1	1-3	17-19	20-21	20-21	1
2	1-3	17-19	18	18 joins 20-21	1
3	1-3	16	19	19 joins 18-20-21	0.75
4	1-3	16	17	17 joins 18-20-21-19	0.4
5	1-3	16	-	16 joins 17-18-20-21-19	0.3333
6	1-3	14	-	14 joins 16-17-18-20-21-19	0.4286
7	1-3	12	-	12 joins 14-16-17-18-20-21-19	0.5
(cluster 1)	12-14-16-17-18-20-21-19				0.2500
8	1-3	22-24	-	22-24	1.5
9	1-3	15	23	23 joins 22-24	0.6667
10	1-3	13-15	15	15 joins 22-24-23	0.75
11	1-3	13	-	13 joins 15-22-24-23	0.4
(cluster 2)	13-15-22-24-23				0.2000
12	1-3	-	-	1-3	2
13	7	-	-	7 joins 1-3	1.3333
14	6	-	-	6 joins 1-3-7	0.75
15	2	-	-	2 joins 1-3-7-6	0.6
16	5	-	-	5 joins 1-3-7-6-2	0.3333
17	8	-	-	8 joins 1-3-7-6-2-5	0.4286
18	11	-	-	11 joins 1-3-7-6-2-5-8	0.3750
(cluster 3)	1-3-7-6-2-5-8-11				0.1250
19	9-10	-	-	9-10	1.5
(cluster 4)	9-10				1
(cluster 5)	4				0

Table 4.5: Clustering under 5% disruption

Table 4.6 shows results of the network with partial failure in links at the 5 percentage of disruption probability level.

Scenario 5: Partially disrupted links					
Number of iteration	1st best option	2nd best option	3rd best option	Selected	*q*
1	1-4	20-21		20-21	1.5
2	1-4	18		18 joins 20-21	1
3	1-4	16	19	19 joins 18-20-21	0.75
4	1-4	16	17	17 joins 18-20-21-19	0.4
5	1-4	16	-	16 joins 17-18-20-21-19	0.3333
6	1-4	14	-	14 joins 16-17-18-20-21-19	0.4286
7	1-4	12	-	12 joins 14-16-17-18-20-21-19	0.5
(cluster 1)	12-14-16-17-18-20-21-19				0.2500
8	1-4	22-24	-	22-24	1.5
9	1-4	15	23	23 joins 22-24	0.6667
10	1-4	13-15	15	15 joins 22-24-23	0.75
11	1-4	13	-	13 joins 15-22-24-23	0.4
(cluster 2)	13-15-22-24-23				0.2000
12	1-4	-	-	1-4	2
13	7	-	-	7 joins 1-4	1.3333
14	3	-	-	3 joins 1-4-7	1
15	6	-	-	6 joins 1-4-7-3	0.6
16	2	-	-	2 joins 1-4-7-3-6	0.5
17	5	-	-	5 joins 1-4-7-3-6-2	0.2857
18	8	-	-	8 joins 1-4-7-3-6-2-5	0.3750
(cluster 3)	1-4-7-3-6-2-5-8				0.1250
19	9-10	-	-	9-10	2
20	11			11 joins 9-10	1
(cluster 4)	9-10-11				0.6667

Table 4.6: Clustering under 5% disruption

Table 4.7 provides the results of a scenario where the network is challenged with partial failure in nodes at the 5 percentage of disruption probability level.

Scenario 6: Partially disrupted links						
Number of iteration	1st best option	2nd best option	3rd best option	4th best option	selected	*q*
1	1-3	2-5	20-21	-	20-21	1.5
2	1-3	2-5	18	-	18 joins 20-21	1
3	1-3	2-5	16	19	19 joins 18-20-21	0.75
4	1-3	2-5	16	17	17 joins 18-20-21-19	0.4
5	1-3	2-5	16	-	16 joins 17-18-20-21-19	0.3333
6	1-3	2-5	14	-	14 joins 16-17-18-20-21-19	0.4286
7	1-3	2-5	12	-	12 joins 14-16-17-18-20-21-19	0.5
(cluster 1)	12-14-16-17-18-20-21-19					0.2500
8	1-3	2-5	22-24	-	22-24	1.5
9	1-3	2-5	15	23	23 joins 22-24	0.6667
10	1-3	2-5	13-15	15	15 joins 22-24-23	0.75
11	1-3	2-5	13	-	13 joins 15-22-24-23	0.4
(cluster 2)	13-15-22-24-23					0.2000
12	1-3	2-5	-	-	2-5	2
13	1-2	6	-	-	6 joins 2-5	1.3333
14	1	-	-	-	1 joins 2-5-6	1
15	3	-	-	-	3 joins 1-2-5-6	0.8
16	7	-	-	-	7 joins 1-2-5-6-3	0.5
17	8	-	-	-	8 joins 1-2-5-6-3-7	0.5714
18	11	-	-	-	11 joins 1-2-5-6-3-7-8	0.5
(cluster 3)	1-2-5-6-3-7-8-11					0.2500
19	9-10	-	-	-	9-10	1.5
(cluster 4)	9-10					1
(cluster 5)	4					1

Table 4.7: Clustering under 5% disruption

Table 4.8 provides the results of a scenario for the network with partial failure in nodes and links at the 5 percentage of disruption probability level.

Scenario 7: Partially disrupted links and nodes						
Number of iteration	1st best option	2nd best option	3rd best option	4th best Option	selected	*q*
1	1-3	2-5	20-21	-	20-21	1.5
2	1-3	2-5	18	-	18 joins 20-21	1
3	1-3	2-5	16	19	19 joins 18-20-21	0.75
4	1-3	2-5	16	17	17 joins 18-20-21-19	0.4
5	1-3	2-5	16	-	16 joins 17-18-20-21-19	0.3333
6	1-3	2-5	14	-	14 joins 16-17-18-20-21-19	0.4286
7	1-3	2-5	12	-	12 joins 14-16-17-18-20-21-19	0.5
(cluster 1)	12-14-16-17-18-20-21-19					0.2500
8	1-3	2-5	22-24	-	22-24	1.5
9	1-3	2-5	15	23	23 joins 22-24	0.6667
10	1-3	2-5	13-15	15	15 joins 22-24-23	0.75
11	1-3	2-5	13	-	13 joins 15-22-24-23	0.4
(cluster 2)	13-15-22-24-23					0.2000
12	1-3	2-5	-	-	2-5	2
13	1-2	6	-	-	6 joins 2-5	1.3333
14	1	-	-	-	1 joins 2-5-6	1
15	3	-	-	-	3 joins 1-2-5-6	0.8
16	7	-	-	-	7 joins 1-2-5-6-3	0.5
17	8	-	-	-	8 joins 1-2-5-6-3-7	0.5714
18	11	-	-	-	11 joins 1-2-5-6-3-7-8	0.5
(cluster 3)	1-2-5-6-3-7-8-11					0.2500
19	9-10	-	-	-	9-10	1.5
(cluster 4)	9-10					1
(cluster 5)	4					1

Table 4.8: Clustering under 5% disruption

Table 4.9 provides the results of a scenario where the network is challenged with partial failure in links and complete disruption in nodes at the 5 percentage of disruption probability level.

Scenario 8: Complete disruption in nodes and partially disrupted links					
Number of iteration	1st best option	2nd best option	3rd best option	selected	*q*
1	1-3	20-21	-	20-21	1
2	1-3	18	-	18 joins 20-21	1
3	1-3	16	19	19 joins 18-20-21	0.75
4	1-3	16	17	17 joins 18-20-21-19	0.4
5	1-3	16	-	16 joins 17-18-20-21-19	0.3333
6	1-3	14	-	14 joins 16-17-18-20-21-19	0.4286
7	1-3	12	-	12 joins 14-16-17-18-20-21-19	0.5
(cluster 1)	12-14-16-17-18-20-21-19				0.2500
8	1-3	22-24	-	22-24	1.5
9	1-3	15	23	23 joins 22-24	0.6667
10	1-3	13-15	15	15 joins 22-24-23	0.75
11	1-3	13	-	13 joins 15-22-24-23	0.4
(cluster 2)	13-15-22-24-23				0.2000
12	1-3	-	-	1-3	2
13	7	-	-	7 joins 1-3	1.3333
14	6	-	-	6 joins 1-3-7	0.75
15	2	-	-	2 joins 1-3-7-6	0.6
16	5	-	-	5 joins 1-3-7-6-2	0.3333
17	8	-	-	8 joins 1-3-7-6-2-5	0.4286
18	11	-	-	11 joins 1-3-7-6-2-5-8	0.3750
(cluster 3)	1-3-7-6-2-5-8-11				0.1250
19	9-10	-	-	9-10	1.5
(cluster 4)	9-10				1
(cluster 5)	4				0

Table 4.9: Clustering under 5% disruption

Table 4.10 provides the results of a scenario for the network with partial failure in nodes and complete disruption in links at the 5 percentage of disruption probability level.

Scenario 9: Complete disruption in links and partially disrupted nodes					
Number of iteration	1st best option	2nd best option	3rd best option	selected	q
1	1-2	20-21	-	20-21	1.5
2	1-2	18	-	18 joins 20-21	1
3	1-2	16	19	19 joins 18-20-21	0.75
4	1-2	16	17	17 joins 18-20-21-19	0.4
5	1-2	16	-	16 joins 17-18-20-21-19	0.3333
6	1-2	14	-	14 joins 16-17-18-20-21-19	0.4286
7	1-2	12	-	12 joins 14-16-17-18-20-21-19	0.5
(cluster 1)	12-14-16-17-18-20-21-19				0.2500
8	1-2	22-24	-	22-24	1.5
9	1-2	15	23	23 joins 22-24	0.6667
10	1-2	13-15	15	15 joins 22-24-23	0.75
11	1-2	13	-	13 joins 15-22-24-23	0.4
(cluster 2)	13-15-22-24-23				0.2000
12	1-2	-	-	1-2	2
13	6	-	-	6 joins 1-2	1.3333
14	3	-	-	3 joins 1-2-6	1
15	7	-	-	7 joins 1-2-6-3	0.6
16	8	-	-	8 joins 1-2-6-3-7	0.83
17	5	-	-	5 joins 1-2-6-3-7-8	0.5714
18	11	-	-	11 joins 1-2-6-3-7-8-5	0.5
(cluster 3)	1-2-6-3-7-8-5-11				0.2500
19	9-10	-	-	9-10	1.5
(cluster 4)	9-10				1
(cluster 5)	4				1

Table 4.10: Clustering under 5% disruption

4. Properties

The main properties of algorithm HRP1^{+} and HRP1^{+}control are presented in detail in this section:

Property 1: The algorithms end after a certain number of iterations. This number is less than or equal to $N-1$.

The algorithm runs repetitively until all generated clusters reach to a size and density that satisfies the user-defined density Q and size W constraints. Accordingly by the end of the

runtime of the algorithm, if either *W* or *Q* or both are dissatisfied by the sub-networks produced, the algorithm terminates. Such kind of incident can occur soon after the first iteration (i.e. 1) or at the most in its last iteration (i.e. $N - 1$), therefore we can conclude that the number of iterations required for termination is less than or equal to $N - 1$. (Awasthi, et al. 2009)

Property 2: If r_1 and r_2 represent two sub-networks and $s = r_1 \cup r_2$. We denote *q(x)* as the density of a subnetwork *x*.

The total number of nodes connected to *s* is equal to the total number of nodes connected to r_2 minus r_1 plus the total number of exterior nodes connected to r_1 excluding r_2. Thus: $q(s) \leq$ Max $\{q(r_1), q(r_2)\}$. (Awasthi, et al. 2009)

Property 3: The computational complexity is known as an index of method evaluation. It represents the execution time required for the algorithm to solve the problem. Therefore, it is highly desirable for a method to reach an acceptable, near-optimal solution in a reasonable amount of time i.e. complexity. Furthermore, the computational time increases with the increase in density criterion *Q*. Here, the complexity of HRP1$^+$ is equal to $O(N^3)$. The algorithm firstly includes density criterion and secondly it explores sub-networks with maximum density as well as all connected pairs to find the lowest density of all other possible combinations in the network.

4.3.2. HRP2$^+$ Partitioning Technique

1. Description

HRP2$^+$ is also a cluster- based approach that runs recursively and is formulated specially to deal with system disruption and delay. It distinguishes from HRP1$^+$ in satisfying its density constraint. The objective of this model is to minimize the maximum density while HRP1$^+$ explores the network for pairs of nodes with density greater than or equal to *Q*.

HRP2$^+$ identifies all possible unions between connected subsets in favor of integrating pair of subsets that correspond with its two user-defined criteria: size (*W*) and density (*Q*).Therefore combinations with a size less than *W* and a density greater than *Q* are credible and sorted in order to define and select the highest and the next highest density subset whose union leads to a subset with an acceptable size and density.

Since HRP2$^+$ is an iterative approach it continues to combine subsets until the ultimate subsets respectively contain a number of nodes and a density less than or equal to *W* and *Q*. Note that the algorithm works on the assumption that size restriction has priority over density constraint and must be satisfied before its density verification. This means in the case of conflict between two constraints size wins over density, therefore the algorithm ends assigning nodes to a sub-network that has reached the maximum size regardless of the density condition.

2. Pseudo-code

Main variables are as follows:

N: total number of nodes

K: size of the input matrix that is the number of clusters at step zero

i,j: connected nodes in the network

Q: maximum density allowed in a subgroup (user-defined)

W: maximum nodes in a subgroup (user-defined)

S(*i*) number of successors

P(*i*) number of predecessors

q (i)= S(*i*)+ *P*(*i*)*:*total external connections in a subgroup defined by external nodes

ls list of successors

lp list of predecessors

nb :total number of nodes in each subset that's initially one

F_q *= q (i)/nb:* externally connected nodes over cardinality

w Destiny of the new cluster in each iteration.

i_1 and j_1 are temporary variable set to determine the optimum combination.

r_n: A vector of size *n* where components represent connected nodes to node *i*

The algorithm starts with *N=K,* the number of subsets in the first iteration is equal to the number of nodes in the network.

A high level scope of HRP2$^+$ comes as follows:

Input:

K: number of clusters

x: matrix of links(dynamic property)

W and *Q*; user defined variables

Output:

Final set of generated clusters

1. Initialize *K* cluster
2. While termination condition is not satisfied do

3. Compute *S* (*i*) and *P* (*i*)
4. Create *ls* and *lp*
5. Select *i1* based on density constraint and node count limit in each cluster
6. Search for *r* from *ls* and *lp* with respect to user defined constraints
7. Assign nodes to the clusters
8. Update clusters
9. End while
10. Detect damage sensitive elements (when applying HRP2^{+} control)
11. Run disruption scenarios

A low-level stepwise code of our Algorithm is as follow:

1. For *i*=1:*K*
   1.1. For *j*=1:*S(i)*
      1.1.1. *r*= *ls* (*i,j*);
      1.1.2. [*w*]= merging2 (*i,r,x,nb*);
      1.1.3. If (*w* > *X*) and (*nb(i)+nb(r)* less than or equal to *W*) and (*w* greater than *Q*) and (*wx(i,r)* less than or equal to w_{temp}) then
         1.1.3.1. *X*=*w*;
         1.1.3.2. i_1=i;
         1.1.3.3. j_1=r;
         1.1.3.4. *cont* =1;
         1.1.3.5. End If
      1.1.4. End For
   1.2. For *j*=1:*P(i)*

1.2.1. r= lp (i,j);

1.2.2. [w]= merging2 (i,r,x,nb); merging function

1.2.3. If (w greater than X) and ($nb(i)+nb(r)$ less than or equal to W) and (w greater than Q) and ($wx(i,r)$ less than or equal to w_{temp})

1.2.3.1. X=w;

1.2.3.2. i_1= I;

1.2.3.3. j_1= r;

1.2.3.4. *cont*=1;

1.2.3.5. End If

1.2.4. End for

2. End for
3. If *cont* equals to 0

   3.1. break;

   3.2. End If
4. If j_1 equals to -1

   4.1. break;

   4.2. End If

3. Numerical Example

We have applied HRP2$^+$ on the same hypothetical network studied for when applying HRP1$^+$. Our user defined parameters remain the same. The resulting clusters are shown in table 4.11.

No Disruption in the Network							
clusters							**Q**
1 2 6 8 4 11 5	3 7	9 10 12 14 15 16 18 19	13	17	20 21	22 24 23	0.4286 1 0.8750 2 2 0.5000 0.3333

Table 4.11: No node disruption – No link disruption

The step wise procedure of computation for a network without disruption is shown in table 4.12. The table displays all possible options of node consolidation at iteration while satisfying our constraints of the program. The algorithm chooses the node pair with the highest density (w) among the other options. For instance in row eight of table 4.12 we see that the options for combinations are to:

1) Unite subset 3, 7 with subset containing nodes 1, 2, 6 and 8

2) Unite node 6 to the previous combination that is the cluster containing node 1, 2, 6 and 8

3) Unite subset 10, 12 with subset containing nodes 14 and 15 that is the combination from step 2

4) Merge node 16 with subset containing nodes 14 and 15

In this step since the W constraint still maintains it simply chooses the last option over other choices as it allows a greater value for Q.

Scenario 1: Network with no disruption							
Number of iteration	1st best option	2nd best option	3rd best option	4th best option	5th best option	selected	*w*
1	1-2	5-8	6-8	-	-	6-8	3
2	1-2	2 and 6-8	14-15	-	-	14-15	2.5
3	1-2	2 and 6-8	-	-	-	2 joins 6-8	2.33
4	1 and 2-6-8	-	-	-	-	1 joins 2-6-8	2
5	3 and 1-2-6-8	4 and 1-2-6-8	7 and 1-2-6-8	3-7	-	3-7	2
6	3-7 and 1-2-6-8	4 and 1-2-6-8	9-10	10-12	-	10-12	2
7	3-7 and 1-2-6-8	4 and 1-2-6-8	10-12 and 14-15	14-15 and 16	-	16 joins 14-15	2
8	3-7 and 1-2-6-8	4 and 1-2-6-8	10-12 and 14-15-16	18 and 14-15-16	18-19	18 -19	2
9	3-7 and 1-2-6-8	4 and 1-2-6-8	10-12 and 14-15-16	-	-	10-12 joins 14-15-16	1.60
10	3-7 and 1-2-6-8	4 and 1-2-6-8	10-12-14-15-16 and 18-19	-	-	10-12-14-15-16 joins 18-19	1.4286
11	3-7 and 1-2-6-8	4 and 1-2-6-8	-	-	-	4 joins 1-2-6-8	1.40
12	3-7 and 1-2-6-8-4	5 and 1-2-6-8-4	9 and 1-2-6-8-4	9 and 10-12-14-15-16-18-19	-	9 joins 10-12-14-15-16-18-19	1.25
13	3-7 and 1-2-6-8-4	5 and 1-2-6-8-4	11 and 1-2-6-8-4	-	-	11 joins 1-2-6-8-4	1
14	3-7 and 1-2-6-8-4-11	5 and 1-2-6-8-4-11	20-21	-	-	20-21	1
15	3-7 and 1-2-6-8-4-11	5 and 1-2-6-8-4-11	22-24	-	-	22-24	1
16	3-7 and 1-2-6-8-4-11	5 and 1-2-6-8-4-11	23 and 22-24	-	-	23 joins 22-24	0.6667
17	3-7 and 1-2-6-8-4-11	5 and 1-2-6-8-4-11	-	-	-	5 joins 1-2-6-8-4-11	0.4286
(cluster 1)	1 2 6 8 4 11 5						0.4286
(cluster 2)	3 7						1
(cluster 3)	9 10 12 14 15 16 18 19						0.8750
(cluster 4)	13						2
(cluster 5)	17						2
(cluster 6)	20 21						0.5
(cluster 7)	22 24 23						0.3333

Table 4.12: No node disruption – No link disruption

Table 4.13 provides the results of a scenario where the network deals with complete failure in links at the 5 percentage of disruption probability level.

Scenario 2: Complete disruption in links							
Number of iteration	1st best option	2nd best option	3rd best option	4th best option	5th best option	selected	*w*
1	1-2	5-8	6-8	-	-	6-8	3
2	1-2	2 and 6-8	-	-	-	2 joins 6-8	2.33
3	1 and 2-6-8	-	-	-	-	1 joins 2-6-8	2
4	3 and 1-2-6-8	4 and 1-2-6-8	7 and 1-2-6-8	3-7	-	3-7	2
5	3-7 and 1-2-6-8	4 and 1-2-6-8	9-10	10-12	-	10-12	2
6	3-7 and 1-2-6-8	4 and 1-2-6-8	10-12 and 14	14-15	-	14-15	2
7	3-7 and 1-2-6-8	4 and 1-2-6-8	10-12 and 14-15	16 and 14-15	16-18	16-18	2
8	3-7 and 1-2-6-8	4 and 1-2-6-8	10-12 and 14-15	19 and16-18	-	19 joins16-18	1.6667
9	3-7 and 1-2-6-8	4 and 1-2-6-8	10-12 and 14-15	-	-	10-12joins 14-15	1.5
10	3-7 and 1-2-6-8	4 and 1-2-6-8	-	-	-	4 joins 1-2-6-8	1.40
11	3-7 and 1-2-6-8-4	5 and 1-2-6-8-4	9 and 1-2-6-8-4	9 and 10-12-14-15	10-12-14-15 and 16-18-19	10-12-14-15 joins 16-18-19	1.2857
12	3-7 and 1-2-6-8-4	5 and 1-2-6-8-4	9 and 1-2-6-8-4	9 and 10-12-14-15-16-18-19	-	9 joins 10-12-14-15-16-18-19	1.1250
13	3-7 and 1-2-6-8-4	5 and 1-2-6-8-4	11 and 1-2-6-8-4	-	-	11 joins 1-2-6-8-4	1
14	3-7 and 1-2-6-8-4-11	5 and 1-2-6-8-4-11	20-21	-	-	20-21	1
15	3-7 and 1-2-6-8-4-11	5 and 1-2-6-8-4-11	22-24	-	-	22-24	1
16	3-7 and 1-2-6-8-4-11	5 and 1-2-6-8-4-11	-	-	-	5 joins 1-2-6-8-4-11	0.4286
17	23 and 22-24	-	-	-	-	23 joins 22-24	0.3333
(cluster 1)	1 2 6 8 4 11 5						0.4286
(cluster 2)	3 7						1
(cluster 3)	9 10 12 14 15 16 18 19						0.8750
(cluster 4)	13						2
(cluster 5)	17						2
(cluster 6)	20 21						0.5
(cluster 7)	22 24 23						0.3333

Table 4.13: Clustering under 5% disruption

Table 4.14 provides the results of a scenario where the network deals with complete failure in nodes at the 5 percentage of disruption probability level.

Scenario 3: Complete disruption in nodes							
Number of iteration	1st best option	2nd best option	3rd best option	4th best option	5th best option	selected	*w*
1	1-2	5-8	6-8	-	-	6-8	3
2	1-2	2 and 6-8	-	-	-	2 joins 6-8	2.33
3	1 and 2-6-8	-	-	-	-	1 joins2-6-8	2
4	3 and 1-2-6-8	4 and 1-2-6-8	7 and 1-2-6-8	3-7	-	3-7	2
5	3-7 and 1-2-6-8	4 and 1-2-6-8	9-10	10-12	-	10-12	2
6	3-7 and 1-2-6-8	4 and 1-2-6-8	13-15	-	-	13-15	2
7	3-7 and 1-2-6-8	4 and 1-2-6-8	10-12 and 13-15	-	-	10-12 joins 13-15	2
8	3-7 and 1-2-6-8	4 and 1-2-6-8	18-19	-	-	18-19	1.5
9	3-7 and 1-2-6-8	4 and 1-2-6-8	-	-	-	4 joins 1-2-6-8	1.5
10	3-7 and 1-2-6-8-4	5 and 1-2-6-8-4	9 and 1-2-6-8-4	9 and 10-12-13-15	-	9 joins 10-12-13-15	1.20
11	3-7 and 1-2-6-8-4	5 and 1-2-6-8-4	11 and 1-2-6-8-4	-		11 joins 1-2-6-8-4	1
12	3-7 and 1-2-6-8-4-11	5 and 1-2-6-8-4-11	9-10-12-13-15 and 14	9-10-12-13-15 and 22	20 and 18-19	20 joins 18-19	1
13	3-7 and 1-2-6-8-4-11	5 and 1-2-6-8-4-11	9-10-12-13-15 and 14	9-10-12-13-15 and 22	22-24	22-24	1
14	3-7 and 1-2-6-8-4-11	5 and 1-2-6-8-4-11	9-10-12-13-15 and 14	9-10-12-13-15 and 22-24	9-10-12-13-15 and 23	23 joins 9-10-12-13-15	0.8333
15	3-7 and 1-2-6-8-4-11	5 and 1-2-6-8-4-11	-	-	-	5 joins 1-2-6-8-4-11	0.4286
16	14 and 9-10-12-13-15-23	9-10-12-13-15 and 22-24	-	-	-	9-10-12-13-15 joins 22-24	0.3750
17	17 and 18-19-20	-	-	-	-	17 joins 18-19-20	0.25
(cluster 1)	1 2 6 8 4 11 5						0.4286
(cluster 2)	3 7						1
(cluster 3)	9 10 12 13 15 23 22 24						0.3750
(cluster 4)	14						2
(cluster 5)	16						0
(cluster 6)	17 18 19 20						0.2500
(cluster 7)	21						1

Table 4.14: Clustering under 5% disruption

Table 4.15 shows results of the network with complete failure in both links and nodes at the 5 percentage of disruption probability level.

Scenario 4: Complete disruption in links and nodes					
Number of iteration	1st best option	2nd best option	3rd best option	selected	*w*
1	1-2	5-8	-	5-8	2.5
2	1-2	2 and 5-8	-	2 joins 5-8	2
3	1 and 2-5-8	10-12	-	10-12	2
4	1 and 2-5-8	10-12and 14	14-15	14-15	2
5	1 and 2-5-8	14-15 and 16	16-18	16-18	2
6	1 and 2-5-8	19 and 16-18	-	19 joins 16-18	1.6667
7	1 and 2-5-8	-	-	1 joins 2-5-8	1.5
8	1-2-5-8 and 4	9and 10-12	10-12and 14-15	10-12 joins 14-15	1.5
9	1-2-5-8 and 4	10-12-14-15 and 16-18-19	-	10-12-14-15 joins 16-18-19	1.2857
10	1-2-5-8 and 4	-	-	4 joins 1-2-5-8	1.2000
11	1-2-5-8-4 and 6	1-2-5-8-4 and 9	9 and10-12-14-15-16-18-19	9 joins 10-12-14-15-16-18-19	1.1250
12	1-2-5-8-4 and 6	1-2-5-8-4 and 11	-	11 joins1-2-5-8-4	1
13	1-2-5-8-4-11 and 6	20-21	-	20-21	1
14	1-2-5-8-4-11 and 6	22-24	-	22-24	1
15	1-2-5-8-4-11 and 6	-	-	6 joins1-2-5-8-4-11	0.4286
16	1-2-5-8-4-11-6 and 7	23 and 22-24	-	23 joins 22-24	0.3333
17	1-2-5-8-4-11-6 and 7	-	-	7 joins 1-2-5-8-4-11-6	0.1250
(cluster 1)	1 2 5 8 4 11 6 7				0.4286
(cluster 2)	3				1
(cluster 3)	9 10 12 14 15 16 18 19				0.3750
(cluster 4)	13				2
(cluster 5)	17				0
(cluster 6)	20 21				0.2500
(cluster 7)	22 24 23				1

Table 4.15: Clustering under 5% disruption

Table 4.16 shows results of the network with partial disruption in links at 5 percentage of disruption probability level.

Scenario 5: Partially disrupted links							
Number of iteration	1st best option	2nd best option	3rd best option	4th best option	5th best option	selected	*w*
1	1-2	5-8	6-8	-	-	6-8	3
2	1-2	2 and 6-8	14-15	-	-	14-15	2.5
3	1-2	2 and 6-8	-	-	-	2 joins 6-8	2.33
4	1 and 2-6-8	-	-	-	-	1 joins 2-6-8	2
5	3 and 1-2-6-8	4 and 1-2-6-8	7 and 1-2-6-8	3-7	-	3-7	2
6	3-7 and 1-2-6-8	4 and 1-2-6-8	9-10	10-12	-	10-12	2
7	3-7 and 1-2-6-8	4 and 1-2-6-8	10-12 and 14-15	14-15 and 16	-	16 joins 14-15	2
8	3-7 and 1-2-6-8	4 and 1-2-6-8	10-12 and 14-15-16	18 and 14-15-16	18-19	18 -19	2
9	3-7 and 1-2-6-8	4 and 1-2-6-8	10-12 and 14-15-16	-	-	10-12 joins 14-15-16	1.60
10	3-7 and 1-2-6-8	4 and 1-2-6-8	10-12-14-15-16 and 18-19	-	-	10-12-14-15-16 joins 18-19	1.4286
11	3-7 and 1-2-6-8	4 and 1-2-6-8	-	-	-	4 joins 1-2-6-8	1.40
12	3-7 and 1-2-6-8-4	5 and 1-2-6-8-4	9 and 1-2-6-8-4	9 and 10-12-14-15-16-18-19	-	9 joins 10-12-14-15-16-18-19	1.25
13	3-7 and 1-2-6-8-4	5 and 1-2-6-8-4	11 and 1-2-6-8-4	-	-	11 joins 1-2-6-8-4	1
14	3-7 and 1-2-6-8-4-11	5 and 1-2-6-8-4-11	20-21	-	-	20-21	1
15	3-7 and 1-2-6-8-4-11	5 and 1-2-6-8-4-11	22-24	-	-	22-24	1
16	3-7 and 1-2-6-8-4-11	5 and 1-2-6-8-4-11	23 and 22-24	-	-	23 joins 22-24	0.6667
17	3-7 and 1-2-6-8-4-11	5 and 1-2-6-8-4-11	-	-	-	5 joins 1-2-6-8-4-11	0.4286
(cluster 1)	1 2 6 8 4 11 5						0.4286
(cluster 2)	3 7						1
(cluster 3)	9 10 12 14 15 16 18 19						0.8750
(cluster 4)	13						2
(cluster 5)	17						2
(cluster 6)	20 21						0.5
(cluster 7)	22 24 23						0.3333

Table 4.16: Clustering under 5% disruption

Table 4.17 provides the results of a scenario where the network with partial failure in nodes at the 5 percentage of disruption probability level.

Scenario 6: Partially disrupted nodes							
Number of iteration	1st best option	2nd best option	3rd best option	4th best option	5th best option	selected	*w*
1	1-2	5-8	6-8	-	-	6-8	3
2	1-2	2 and 6-8	14-15	-	-	14-15	2.5000
3	1-2	2 and 6-8	-	-	-	2 joins 6-8	2.3333
4	1 and 2-6-8	-	-	-	-	1 joins 2-6-8	2
5	3 and 1-2-6-8	9 and 1-2-6-8	7 and 1-2-6-8	3-7	-	3-7	2
6	3-7 and 1-2-6-8	9 and 1-2-6-8	9-10	10-12	-	10-12	2
7	3-7 and 1-2-6-8	9 and 1-2-6-8	10-12 and 14-15	14-15 and 16	-	14-15 joins 16	2
8	3-7 and 1-2-6-8	9 and 1-2-6-8	10-12 and 14-15-16	14-15-16 and 18	18-19	18-19	2
9	3-7 and 1-2-6-8	9 and 1-2-6-8	10-12 and 14-15-16	-	-	10-12 joins 14-15-16	1.6000
10	3-7 and 1-2-6-8	9 and 1-2-6-8	19 and 10-12-14-15-16	-	-	19 joins 10-12-14-15-16	1.4286
11	3-7 and 1-2-6-8	9 and 1-2-6-8	-	-	-	9 joins 1-2-6-8	1.4000
12	3-7 and 1-2-6-8-9	1-2-6-8-9 and 11	10-12-14-15-16-19 and 20	-	-	10-12-14-15-16-19 joins 20	1.2500
13	3-7 and 1-2-6-8-9	11 and 1-2-6-8-9	-	-	-	11 joins 1-2-6-8-9	1.1667
14	3-7 and 1-2-6-8-9-11	3-7 and 4	-	-	-	4 joins 3-7	1
15	1-2-6-8-9-11 and 5	22-24	-	-	-	22-24	1
16	1-2-6-8-9-11 and 5	23 and 22-24	-	-	-	23 joins 22-24	0.6667
17	1-2-6-8-9-11 and 5	-	-	-	-	1-2-6-8-9-11 joins 5	0.4286
(cluster 1)	3 7 4						0.6667
(cluster 2)	5 1 2 6 8 9 11						0.4286
(cluster 3)	10 12 14 15 16 18 19 20						0.8750
(cluster 4)	13						2
(cluster 5)	17						2
(cluster 6)	21						1
(cluster 7)	22 24 23						0.3333

Table 4.17: Clustering under 5% disruption

Table 4.18 provides the results of a scenario where the network deals with partial failure in both links and nodes at the 5 percentage of disruption probability level.

Scenario 7: Partially disrupted nodes and links							
Number of iteration	1st best option	2nd best option	3rd best option	4th best option	5th best option	selected	*w*
1	1-2	5-8	6-8	-	-	6-8	3
2	1-2	2 and 6-8	14-15	-	-	14-15	2.5000
3	1-2	2 and 6-8	-	-	-	2 joins 6-8	2.3333
4	1 and 2-6-8	-	-	-	-	1 joins 2-6-8	2
5	3 and 1-2-6-8	9 and 1-2-6-8	7 and 1-2-6-8	3-7	-	3-7	2
6	3-7 and 1-2-6-8	9 and 1-2-6-8	9-10	10-12	-	10-12	2
7	3-7 and 1-2-6-8	9 and 1-2-6-8	10-12 and 14-15	14-15 and 16	-	14-15 joins 16	2
8	3-7 and 1-2-6-8	9 and 1-2-6-8	10-12 and 14-15-16	14-15-16 and 18	18-19	18-19	2
9	3-7 and 1-2-6-8	9 and 1-2-6-8	10-12 and 14-15-16	-	-	10-12 joins 14-15-16	1.6000
10	3-7 and 1-2-6-8	9 and 1-2-6-8	19 and 10-12-14-15-16	-	-	19 joins 10-12-14-15-16	1.4286
11	3-7 and 1-2-6-8	9 and 1-2-6-8	-	-	-	9 joins 1-2-6-8	1.4000
12	3-7 and 1-2-6-8-9	1-2-6-8-9 and 11	10-12-14-15-16-19 and 20	-	-	10-12-14-15-16-19 joins 20	1.2500
13	3-7 and 1-2-6-8-9	11 and 1-2-6-8-9	-	-	-	11 joins 1-2-6-8-9	1.1667
14	3-7 and 1-2-6-8-9-11	3-7 and 4	-	-	-	4 joins 3-7	1
15	1-2-6-8-9-11 and 5	22-24	-	-	-	22-24	1
16	1-2-6-8-9-11 and 5	23 and 22-24	-	-	-	23 joins 22-24	0.6667
17	1-2-6-8-9-11 and 5	-	-	-	-	1-2-6-8-9-11 joins 5	0.4286
(cluster 1)	3 7 4						0.6667
(cluster 2)	5 1 2 6 8 9 11						0.4286
(cluster 3)	10 12 14 15 16 18 19 20						0.8750
(cluster 4)	13						2
(cluster 5)	17						2
(cluster 6)	21						1
(cluster 7)	22 24 23						0.3333

Table 4.18: Clustering under 5% disruption

Table 4.19 shows results of the network with partial disruption in links and complete disruption in nodes at 5 percentage of disruption probability level.

Scenario 8: Complete disruption in nodes and partially disrupted links						
Number of iteration	1st best option	2nd best option	3rd best option	4th best option	Selected	*w*
1	1-2	5-8	-	-	5-8	2.5
2	1-2	2 and 5-8	14-15	-	14-15	2.5
3	1-2	2 and 5-8	-	-	2 joins 5-8	2
4	1 and 2-5-8	10-12	-	-	10-12	2
5	1 and 2-5-8	10-12and 14-15	14-15 and 16	-	14-15 joins 16	2
6	1 and 2-5-8	10-12and 14-15-16	14-15-16 and 18	18-19	18-19	2
7	1 and 2-5-8	10-12and 14-15-16	-	-	10-12 joins 14-15-16	1.6000
8	1 and 2-5-8	-	-	-	1 joins 2-5-8	1.5
9	1-2-5-8 and 4	9and 10-12-14-15-16	10-12-14-15-16 and 18-19	-	10-12-14-15-16 joins 18-19	1.4286
10	1-2-5-8 and 4	9 and 10-12-14-15-16-18-19	-	-	9 joins 10-12-14-15-16-18-19	1.2500
11	1-2-5-8 and 4	-	-	-	4 joins 1-2-5-8	1.2000
12	1-2-5-8-4 and 6	1-2-5-8-4 and 11	-	-	1-2-5-8-4 joins 11	1
13	1-2-5-8-4-11 and 6	20-21	-	-	20-21	1
14	1-2-5-8-4-11 and 6	22-24	-	-	22-24	1
15	1-2-5-8-4-11 and 6	23 and 22-24	-	-	23 joins 22-24	0.6667
16	1-2-5-8-4-11 and 6	-	-	-	1-2-5-8-4-11 joins 6	0.4286
17	1-2-5-8-4-11-6 and 7	-	-	-	7 joins 1-2-5-8-4-11-6	0.1250
(cluster 1)	1 2 5 8 4 11 6 7					0.1250
(cluster 2)	3					0
(cluster 3)	9 10 12 14 15 16 18 19					0.8750
(cluster 4)	13					2
(cluster 5)	17					2
(cluster 6)	20 21					0.5
(cluster 7)	22 24 23					0.3333

Table 4.19: Clustering under 5% disruption

Table 4.20 shows results of the network with partial disruption in nodes and complete disruption in links at the 5 percentage of disruption probability level.

Scenario 9: Complete disruption in links and partially disrupted nodes							
Number of iteration	1st best option	2nd best option	3rd best option	4th best option	-	selected	*w*
1	1-2	2-6	5-8	6-8	-	6-8	3
2	1-2	2 and 6-8	14-15	-	-	14-15	2.5
3	1-2	2 and 6-8	-	-	-	2 joins 6-8	2.3333
4	1 and 2-6-8	3 and 2-6-8	-	-	-	3 joins 2-6-8	2
5	1 and 2-6-8-3	9 and 2-6-8-3	9-10	10-12	-	10-12	2
6	1 and 2-6-8-3	9 and 2-6-8-3	10-12 and 14-15	16 and 14-15	-	16 joins 14-15	2
7	1 and 2-6-8-3	9 and 2-6-8-3	10-12 and 14-15-16	14-15-16 and 18	18-19	18-19	2
8	1 and 2-6-8-3	9 and 2-6-8-3	10-12 and 14-15-16	-	-	10-12 joins 14-15-16	1.6000
9	1 and 2-6-8-3	9 and 2-6-8-3	10-12-14-15-16 and 18-19	-	-	10-12-14-15-16 joins 18-19	1.4286
10	1 and 2-6-8-3	9 and 2-6-8-3	-	-	-	9 joins 2-6-8-3	1.4000
11	1 and 2-6-8-3-9	11 and 2-6-8-3-9	10-12-14-15-16-18-19 and 20	-	-	10-12-14-15-16-18-19 joins 20	1.2500
12	1 and 2-6-8-3-9	11 and 2-6-8-3-9	-	-	-	11 joins 2-6-8-3-9	1.1667
13	1 and 2-6-8-3-9-11	4-7	-	-	-	4-7	1
14	1 and 2-6-8-3-9-11	22-24	-	-	-	22-24	1
15	1 and 2-6-8-3-9-11	-	-	-	-	1 joins 2-6-8-3-9-11	0.7143
16	5 and 1-2-6-8-3-9-11	23 and 22-24	-	-	-	23 joins 22-24	0.6667
17	5 and 1-2-6-8-3-9-11	-	-	-	-	5 joins 1-2-6-8-3-9-11	0.3750
(cluster 1)	4 7						1
(cluster 2)	5 1 2 6 8 3 9 11						0.3750
(cluster 3)	10 12 14 15 16 18 19 20						0.8750
(cluster 4)	13						2
(cluster 5)	17						2
(cluster 6)	21						1
(cluster 7)	22 24 23						0.3333

Table 4.20: Clustering under 5% disruption

4. Properties

The properties of algorithm HRP2$^+$ are described as follows:

1) The algorithm ends after a certain number of iterations that is equal to or less than $N - 1$.
2) Subsets must be connected with at least one link. This means the mentioned link must have its head in one subset and its end in another.
3) All subsets must have an acceptable number of nodes that is smaller than or equal to the user defined *W*.
4) The density of the subset which is formulized as the number of external links divided by the size of the subset (cardinality) must finally reach an amount less than or equal to *Q*.
5) The resulting subset in each round of the program must not only satisfy size (*W*) constraint but also its density must be lower than the maximum density between the selected subsets.

   If r_1 and r_2 represent two sub-networks and $s = r_1 \cup r_2$. We denote *w(x)* as the density of a subnetwork *x*.

   The total number of nodes connected to *s* is equal to the total number of nodes connected to r_2 minus r_1 plus the total number of exterior nodes connected to r_1 excluding r_2. Thus: $w(s) \leq \text{Max } \{w(r_1), w(r_2)\}$. (Awasthi, et al. 2009)
6) The computational time increases with the increase in density criterion *Q*. Here, the complexity of HRP2$^+$ is equal to $O(N^2)$ as the algorithm firstly includes density criterion and secondly it explores sub-networks with maximum density as well as all connected pairs to find the lowest density of all other possible combinations in the network.

4.3.3. HRP1$^+$and HRP2$^+$ with Control

1. Description

We maintain that since adapting the infrastructure of a network involves changing strategic decisions, it is more desirable to make additional investment in measures that help building a fortified system and increase network resiliency especially for critical as well as damage sensitive elements. Therefore in addition to our initial implementation, here; since the decomposed network is facing a randomly generated disruption; damage sensitive elements are also exposed to risk. Thus, in order to ensure flow of information between clusters; we improve our model by sustaining connectivity among ultimate clusters. That is why we have introduced a check and control operator to the system with the purpose of identifying and flagging the damage sensitive nodes/links in favor of vaccinating them with system reliability and resiliency. Hence once detected, we immune them by means of intense monitoring and inspection to prevent all possible chances of faulty behavior or breakdown. Further depending on the scale of the network we may also capacitate damage sensitive elements with an alternative that is supposed to function as the original element in the case of failure. That is they are established to carry out the assigned task according to the new circumstances. Nevertheless to achieve a practical solution for any real-time network certain constraints namely time and cost must be considered and satisfied in order to guarantee the viability of our solution approach.

In this subsection we have examined our improved version of HRP1$^+$ and HRP2$^+$ on the same example. That is, we show the impact of system resiliency in a network, executing our graph partitioning approach while including our control and check step in each scenario

2. Pseudo-code

Control/check function after clustering the network for both $HRP1^+$ and $HRP2^+$:

List of employed variables:

ls list of successors

ch is an array representing a subset of successor nodes (dynamic property)

cric an array of damage sensitive nodes

p_r, p_c variables representing subset of successor nodes in the clusters

q_r, q_c variables representing subset of successor nodes in the damage sensitive vector

A low-level stepwise code of our algorithm is as follows:

Function [*cric,ls*]= control(*x,result*)

1. For *i*=1:*K*

1.1. *ls*(i,1:length(find(x(i,:)==1)))=find(x(i,:)= =1);

1.2. End for

2. For *i*=1: *M*

2.1. For *j*=1: *N*

2.2. If *result* (*i,j*) is not equal to 0

2.2.1. $temp_x$= result (*i,j*);

2.2.2. *ch* = ls($temp_x$,:);

2.3. For *k*=1: length (*ch*)

2.3.1. If *ch* (*k*) is not equal to 0

2.3.1.1. [p_r p_c] =find (result==*ch* (*k*));

2.3.2. If p_r is not equal to *i*

2.3.2.1. [q_r q_c] =find (*cric*= =$temp_x$);

2.3.3. If is empty (q_r)

2.3.3.1. *cric*=[*cric*; $temp_x$];

2.3.3.2. End if

2.3.2.2 End if

2.3.1.2. End if

2.4. End for

3. $temp_x$=[];

4. *ch*=[];

2.2.3. End if

2.5. End for

5. End for

In brief, the control function simply searches the *ls* matrix and projects the connections into our result matrix in order to determine whether the connection has an end in more than one cluster or not. Thus its role is to investigate for pairs of nodes that are assigned to two clusters. The next step would be to maintain one link between successors if the node is identified as a damage sensitive element.

3. Numerical Example

1. Numerical Example of HRP1$^+$ with Control

In this subsection, we apply HRP1$^+$ with control on our predefined network. The stepwise partitioning procedure of our intact network is shown in Table 4.21. In this scenario as can be seen there is no difference between the results from HRP1$^+$ and HRP1$^+$ with Control but the mere fact that the execution time of the program is longer since it requires computing and flagging damage sensitive nodes.

Network with no disruption					
Number of iteration	1st best option	2nd best option	3rd best option	selected	*q*
1	1-4	20-21		20-21	1
2	1-4	18		18 joins 20-21	0.6667
3	1-4	16	19	19 joins 18-20-21	0.5
4	1-4	16	17	17 joins 18-20-21-19	0.2
5	1-4	16	-	16 joins 17-18-20-21-19	0.1667
6	1-4	14	-	14 joins 16-17-18-20-21-19	0.2857
7	1-4	12	-	12 joins 14-16-17-18-20-21-19	0.3750
(cluster 1)	12-14-16-17-18-20-21-19				0.2500
8	1-4	22-24	-	22-24	1
9	1-4	15	23	23 joins 22-24	0.3333
10	1-4	13-15	15	15 joins 22-24-23	0.333
11	1-4	13	-	13 joins 15-22-24-23	0.5
(cluster 2)	13-15-22-24-23				0.2000
12	1-4	-	-	1-4	1.5
13	7	-	-	7 joins 1-4	1
14	3	-	-	3 joins 1-4-7	0.75
15	6	-	-	6 joins 1-4-7-3	0.4
16	2	-	-	2 joins 1-4-7-3-6	0.333
17	5	-	-	5 joins 1-4-7-3-6-2	0.25
18	8	-	-	8 joins 1-4-7-3-6-2-5	0.1429
(cluster 3)	1-4-7-3-6-2-5-8				0.1250
19	9-10	-	-	9-10	1.5
20	11			11 joins 9-10	1
(cluster 4)	9-10-11				0.6667

Table 4.21: No node disruption – No link disruption

Table 4.22 shows results of the network with complete disruption in links while protecting damage sensitive elements at the 5 percentage of disruption probability level.

Scenario 2: Complete disruption in links					
Number of iteration	1st best option	2nd best option	3rd best option	selected	*q*
1	1-4	16-17	-	16-17	1.5
2	1-4	19	-	19 joins 16-17	1
3	1-4	21	-	21 joins 16-17-19	0.75
4	1-4	20	-	20 joins 16-17-19-21	0.4
5	1-4	18	-	18 joins16-17-19-21-20	0.1667
(cluster 1)	16-17-18-20-21-19				0
6	1-4	22-24	-	22-24	1.5
7	1-4	15 and 22-24	23 and 22-24	23 joins 22-24	0.6667
8	1-4	15 and 22-24-23	-	15 joins 22-24-23	0. 7500
9	1-4	13	-	13 joins 15-22-24-23	0.6000
10	1-4	12	14	14 joins13-15-22-24-23	0.3333
11	1-4	12	-	12 joins 13-15-22-24-23-14	0.2857
(cluster 2)	12-13-15-22-24-23-14				0.1429
12	1-4	-	-	1-4	2
13	7	-	-	7 joins 1-4	1.3333
14	3	-	-	3 joins 1-4-7	1
15	6	-	-	6 joins 1-4-7-3	0.6000
16	2	-	-	2 joins 1-4-7-3-6	0.5000
17	5	-	-	5 joins 1-4-7-3-6-2	0.2857
18	8	-	-	8 joins 1-4-7-3-6-2-5	0.3750
(cluster 3)	1-4-7-3-6-2-5-8				0.1250
19	9-10	-	-	9-10	2
20	11	-	-	11 joins 9-10	1
(cluster 4)	9-10-11				0.6667

Table 4.22: Clustering under 5% disruption

Table 4.23 shows results of the network with complete disruption in nodes while protecting damage sensitive elements at the 5 percentage of disruption probability level.

Scenario 3: Complete disruption in nodes					
Number of iteration	1st best option	2nd best option	3rd best option	Selected	*q*
1	1-4	-	-	1-4	1.5
2	7	-	-	7 joins 1-4	1
3	2	-	-	2 joins 1-4-7	1
4	6	-	-	6 joins 1-4-7-2	0.6
5	5	-	-	5 joins 1-4-7-2-6	0.333
6	8	-	-	8 joins 1-4-7-2-6-5	0.4286
7	11	-	-	11 joins 1-4-7-2-6-5-8	0.3750
(cluster 1)	1-4-7-2-6-5-8-11				0.1250
8	9-10	-	-	9-10	1.5
9	12	-	-	12 joins 9-10	1.333
10	13	-	-	13 joins 9-10-12	1
11	14	-	-	14 joins 9-10-12-13	0.8
12	16	-	-	16 joins 9-10-12-13-14	0.8333
13	17	-	-	17 joins 9-10-12-13-14-16	0.7143
14	19	-	-	19 joins 9-10-12-13-14-16-17	0.6250
(cluster 2)	9-10-12-13-14-16-17-19				0.3750
15	15-23	18-20	-	18-20	1.5
16	15-23	21 and 18-20	-	21 joins 18-20	0.6667
(cluster 3)	18-20-21				0.3333
17	15-23	22-24	-	22-24	1.5
18	15-23	23 and 22-24	-	23 joins 22-24	0.6667
19	15	15 and 22-24-23	-	15 joins 22-24-23	0.5
(cluster 4)	15-22-24-23				0.2500
(cluster 5)	3				0

Table 4.23: Clustering under 5% disruption

Table 4.24 shows a step wise clustering result of the network with complete disruption in both nodes and links while protecting damage sensitive elements at the 5 percentage of disruption probability level.

Scenario 4: Complete disruption in links and nodes					
Number of iteration	1st best option	2nd best option	3rd best option	selected	*q*
1	1-4	-	-	1-4	1.5
2	3	-	-	3 joins 1-4	1
3	6	-	-	6 joins 1-4-3	0.75
4	2	-	-	2 joins 1-4-3-6	0.6
5	5	-	-	5 joins 1-4-3-6-2	0.333
6	8	-	-	8 joins 1-4-3-6-2-5	0.4286
7	11	-	-	11 joins 1-4-3-6-2-5-8	0.3750
(cluster 1)	1-4-3-6-2-5-8-11				0.1250
8	9-10	-	-	9-10	1.5
9	12	-	-	12 joins 9-10	1.333
10	14	-	-	14 joins 9-10-12	1
11	13	-	-	13 joins 9-10-12-14	0.6
12	15	-	-	15 joins 9-10-12-14-13	0.6667
13	23	-	-	23 joins 9-10-12-14-13-15	0.5714
14	24	-	-	24 joins 9-10-12-14-13-15-23	0.3750
(cluster 2)	9-10-12-14-13-15-23-24				0.2500
15	16-17	-	-	16-17	1.5
16	19	-	-	19 joins 16-17	1
17	21	-	-	21 joins 16-17-19	0.75
18	20	-	-	20 joins 16-17-19-21	0.4
19	18	-	-	18 joins 16-17-19-21-20	0.1667
(cluster 3)	16-17-19-21-20-18				1
(cluster 4)	7				0
(cluster 5)	22				0

Table 4.24: Clustering under 5% disruption

Table 4.25 shows a step wise clustering result of the network with partial disruption in its links while protecting damage sensitive elements at the 5 percentage of disruption probability level.

Scenario 5: Partially disrupted links					
Number of iteration	1st best option	2nd best option	3rd best option	selected	*q*
1	1-4	20-21	-	20-21	1.5
2	1-4	18 and 20-21	-	18 joins 20-21	1
3	1-4	16	19	19 joins 18-20-21	0.75
4	1-4	16	17	17 joins 18-20-21-19	0.4
5	1-4	16	-	16 joins 17-18-20-21-19	0.333
6	1-4	14	-	14 joins 16-17-18-20-21-19	0.4286
7	1-4	12	-	12 joins 14-16-17-18-20-21-19	0.5
(cluster 1)	12-14-16-17-18-20-21-19				0.2500
8	1-4	22-24	-	22-24	1.5
9	1-4	15	23	23 joins 22-24	0.6667
10	1-4	13-15	15	15 joins 22-24-23	0.75
11	1-4	13	-	13 joins 15-22-24-23	0.4
(cluster 2)	13-15-22-24-23				0.2000
12	1-4	-	-	1-4	2
13	7	-	-	7 joins 1-4	1.333
14	3	-	-	3 joins 1-4-7	1
15	6	-	-	6 joins 1-4-7-3	0.6
16	2	-	-	2 joins 1-4-7-3-6	0.5
17	5	-	-	5 joins 1-4-7-3-6-2	0.2857
18	8	-	-	8 joins 1-4-7-3-6-2-5	0.3750
(cluster 3)	1-4-7-3-6-2-5-8				0.1250
19	9-10	-	-	9-10	2
20	11	-	-	11 joins 9-10	1
(cluster 4)	9-10-11				0.6667

Table 4.25: Clustering under 5% disruption

Table 4.26 shows a step wise clustering result of the network with partial disruption in the nodes while protecting damage sensitive elements at the 5 percentage of disruption probability level.

Scenario 6: Partially disrupted links					
Number of iteration	1st best option	2nd best option	3rd best option	selected	*q*
1	1-4	20-21	-	20-21	1.5
2	1-4	18	-	18 joins 20-21	1
3	1-4	16	19	19 joins 18-20-21	0.75
4	1-4	16	17	17 joins 18-20-21-19	0.4
5	1-4	16	-	16 joins 17-18-20-21-19	0.333
6	1-4	14	-	14 joins 17-18-20-21-19	0.4286
7	1-4	12	-	12 joins 14-17-18-20-21-19	0.5
(cluster 1)	12-14-17-18-20-21-19				0.2500
8	1-4	22-24	-	22-24	1.5
9	1-4	15	23	23 joins 22-24	0.6667
10	1-4	13-15	15	15 joins 22-24-23	0.75
11	1-4	13	-	13 joins 15-22-24-23	0.4
(cluster 2)	13-15-22-24-23				0.2
12	1-4	-	-	1-4	2
13	3	-	-	3 joins 1-4	1.333
14	6	-	-	6 joins 1-4-3	1
15	2	-	-	2 joins 1-4-3-6	0.8
16	5	-	-	5 joins 1-4-3-6-2	0.5
17	8	-	-	8 joins 1-4-3-6-2-5	0.5714
18	11	-	-	11 joins 1-4-3-6-2-5-8	0.5
(cluster 3)	1-4-3-6-2-5-8-11				0.2500
19	9-10	-	-	9-10	1.5
(cluster 4)	9-10				1
(cluster 5)	7				1

Table 4.26: Clustering under 5% disruption

Table 4.27 shows a step wise clustering result of the network with partial disruption in both links and nodes while protecting damage sensitive elements at the 5 percentage of disruption probability level.

Scenario 7: Partially disrupted links and nodes					
Number of iteration	1st best option	2nd best option	3rd best option	selected	*q*
1	1-4	20-21	-	20-21	1.5
2	1-4	18	-	18 joins 20-21	1
3	1-4	16	19	19 joins 18-20-21	0.75
4	1-4	16	17	17 joins 18-20-21-19	0.4
5	1-4	16	-	16 joins 17-18-20-21-19	0.333
6	1-4	14	-	14 joins 17-18-20-21-19	0.4286
7	1-4	12	-	12 joins 14-17-18-20-21-19	0.5
(cluster 1)	12-14-17-18-20-21-19				0.2500
8	1-4	22-24	-	22-24	1.5
9	1-4	15	23	23 joins 22-24	0.6667
10	1-4	13-15	15	15 joins 22-24-23	0.75
11	1-4	13	-	13 joins 15-22-24-23	0.4
(cluster 2)	13-15-22-24-23				0.2
12	1-4	-	-	1-4	2
13	3	-	-	3 joins 1-4	1.333
14	6	-	-	6 joins 1-4-3	1
15	2	-	-	2 joins 1-4-3-6	0.8
16	5	-	-	5 joins 1-4-3-6-2	0.5
17	8	-	-	8 joins 1-4-3-6-2-5	0.5714
18	11	-	-	11 joins 1-4-3-6-2-5-8	0.5
(cluster 3)	1-4-3-6-2-5-8-11				0.2500
19	9-10	-	-	9-10	1.5
(cluster 4)	9-10				1
(cluster 5)	7				1

Table 4.27: Clustering under 5% disruption

Table 4.28 shows a step wise clustering result of the network with partial disruption in its links and complete disruption in nodes while protecting damage sensitive elements at the 5 percentage of disruption probability level.

Scenario 8: Complete disruption in nodes and partially disrupted links				
Number of iteration	1st best option	2nd best option	selected	*q*
1	1-4	-	1-4	1.5
2	3	-	3 joins 1-4	1
3	6	-	6 joins 1-4-3	0.75
4	2	-	2 joins 1-4-3-6	0.6
5	5	-	5 joins 1-4-3-6-2	0.3333
6	8	-	8 joins 1-4-3-6-2-5	0.4286
7	11	-	11 joins 1-4-3-6-2-5-8	0.3750
(cluster 1)	1-4-3-6-2-5-8-11			0.1250
8	9-10	-	9-10	1.5
9	12	-	12 joins 9-10	1.3333
10	13	-	13 joins 9-10-12	1
11	14	-	14 joins 9-10-12-13	0.8
12	16	-	16 joins 9-10-12-13-14	0.8333
13	17	-	17 joins 9-10-12-13-14-16	0.7143
14	19	-	19 joins 9-10-12-13-14-16-17	0.6250
(cluster 2)	9-10-12-13-14-16-17-19			0.3750
15	15-23	18-20	18-20	0.6667
16	15-23	21	21 joins 18-20	1.5
(cluster 3)	18-20-21			0.3333
17	15-23	22-24	22-24	1.5
18	15	23	22-24-23	0.6667
19	15	-	15 joins 22-24-23	0.5
(cluster 4)	15-22-24-23			0.2500

Table 4.28: Clustering under 5% disruption

Table 4.29 shows a step wise clustering result of the network with partial disruption in nodes and complete disruption in links while protecting damage sensitive elements at the 5 percentage of disruption probability level.

Scenario 9: Complete disruption in links and partially disrupted nodes					
Number of iteration	1st best option	2nd best option	3rd best option	selected	q
1	1-4	20-21	-	20-21	1.5
2	1-4	18	-	18 joins 20-21	1
3	1-4	16	19	19 joins 18-20-21	0.75
4	1-4	16	17	17 joins 18-20-21-19	0.4
5	1-4	16	-	16 joins 17-18-20-21-19	0.3333
6	1-4	14	-	14 joins 16-17-18-20-21-19	0.4286
7	1-4	12	-	12 joins 14-16-17-18-20-21-19	0.5000
(cluster 1)	12-14-16-17-18-20-21-19				0.2500
8	1-4	22-24	-	22-24	1.5
9	1-4	15	23	23 joins 22-24	0.6667
10	1-4	13-15	15	15 joins 22-24-23	0.75
11	1-4	13	-	13 joins 15-22-24-23	0.4
(cluster 2)	13-15-22-24-23				0.2000
12	1-4	-	-	1-4	2
13	3	-	-	3 joins 1-4	1.3333
14	6	-	-	6 joins1-4-3	1
15	2	-	-	2 joins 1-4-3-6	0.8000
16	5	-	-	5 joins 1-4-3-6-2	0.5000
17	8	-	-	8 joins 1-4-3-6-2-5	0.5714
18	11	-	-	11 joins 1-4-3-6-2-5-8	0.5
(cluster 3)	1-4-3-6-2-5-8-11				0.2500
19	9-10	-	-	9-10	1.5
(cluster 4)	9-10				1
(cluster 5)	7				1

Table 4.29: Clustering under 5% disruption

2. Numerical Example of HRP2$^+$ with Control

In this subsection we apply HRP2$^+$ with control on our predefined network. The stepwise partitioning procedure of our intact network is shown in Table 4.30. In this scenario as can be seen there is no difference between the results from HRP2$^+$ and HRP2$^+$ with Control but the fact that the execution time of the program is longer since it requires computing and flagging damage sensitive nodes.

Scenario 1:Network with no disruption							
Number of iteration	1st best option	2nd best option	3rd best option	4th best option	5th best option	selected	*w*
1	1-2	5-8	6-8	-	-	6-8	3
2	1-2	2 and 6-8	14-15	-	-	14-15	2.5
3	1-2	2 and 6-8	-	-	-	2 joins 6-8	2.33
4	1 and 2-6-8	-	-	-	-	1 joins 2-6-8	2
5	3 and 1-2-6-8	4 and 1-2-6-8	7 and 1-2-6-8	3-7	-	3-7	2
6	3-7 and 1-2-6-8	4 and 1-2-6-8	9-10	10-12	-	10-12	2
7	3-7 and 1-2-6-8	4 and 1-2-6-8	10-12 and 14-15	14-15 and 16	-	16 joins 14-15	2
8	3-7 and 1-2-6-8	4 and 1-2-6-8	10-12 and 14-15-16	18 and 14-15-16	18-19	18 -19	2
9	3-7 and 1-2-6-8	4 and 1-2-6-8	10-12 and 14-15-16	-	-	10-12 joins 14-15-16	1.60
10	3-7 and 1-2-6-8	4 and 1-2-6-8	10-12-14-15-16 and 18-19	-	-	10-12-14-15-16 joins 18-19	1.4286
11	3-7 and 1-2-6-8	4 and 1-2-6-8	-	-	-	4 joins 1-2-6-8	1.40
12	3-7 and 1-2-6-8-4	5 and 1-2-6-8-4	9 and 1-2-6-8-4	9 and 10-12-14-15-16-18-19	-	9 joins 10-12-14-15-16-18-19	1.25
13	3-7 and 1-2-6-8-4	5 and 1-2-6-8-4	11 and 1-2-6-8-4	-	-	11 joins 1-2-6-8-4	1
14	3-7 and 1-2-6-8-4-11	5 and 1-2-6-8-4-11	20-21	-	-	20-21	1
15	3-7 and 1-2-6-8-4-11	5 and 1-2-6-8-4-11	22-24	-	-	22-24	1
16	3-7 and 1-2-6-8-4-11	5 and 1-2-6-8-4-11	23 and 22-24	-	-	23 joins 22-24	0.6667
17	3-7 and 1-2-6-8-4-11	5 and 1-2-6-8-4-11	-	-	-	5 joins 1-2-6-8-4-11	0.4286
(cluster 1)	1 2 6 8 4 11 5						0.4286
(cluster 2)	3 7						1
(cluster 3)	9 10 12 14 15 16 18 19						0.8750
(cluster 4)	13						2
(cluster 5)	17						2
(cluster 6)	20 21						0.5
(cluster 7)	22 24 23						0.3333

Table 4.30: Clustering under 5% disruption

DS*	1	8	4	11	3	7	12	15	16	18	19	13	17

DS (Damage Sensitive elements)

Table 4.31 shows a step wise clustering result of the network with complete disruption in links while protecting damage sensitive elements at the 5 percentage of disruption probability level.

Scenario 2: Complete disruption in links							
Number of iteration	1st best option	2nd best option	3rd best option	4th best option	5th best option	selected	*w*
1	1-2	5-8	6-8	-	-	6-8	3
2	1-2	2 and 6-8	14-15	-	-	14-15	2.5
3	1-2	2 and 6-8				2 joins 6-8	2.33
4	1 and 2-6-8	-	-	-	-	1 joins 2-6-8	2
5	3 and 1-2-6-8	4 and 1-2-6-8	7 and 1-2-6-8	3-7	-	3-7	2
6	3-7 and 1-2-6-8	4 and 1-2-6-8	9-10	10-12	-	10-12	2
7	3-7 and 1-2-6-8	4 and 1-2-6-8	10-12 and 14-15	16 and 14-15	-	16 joins 14-15	2
8	3-7 and 1-2-6-8	4 and 1-2-6-8	10-12 and 14-15-16	-	-	10-12 joins 14-15-16	1.60
9	3-7 and 1-2-6-8	4 and 1-2-6-8	17-19	-	-	17-19	1.5
10	3-7 and 1-2-6-8	4 and 1-2-6-8	-	-	-	4 joins 1-2-6-8	1.40
11	3-7 and 1-2-6-8-4	5 and 1-2-6-8-4	9 and 1-2-6-8-4	9 and 10-12-14-15-16	-	9 joins 10-12-14-15-16	1.333
12	3-7 and 1-2-6-8-4	5 and 1-2-6-8-4	11 and 1-2-6-8-4	17-19 and 9-10-12-14-15-16-	18 and 9-10-12-14-15-16	18 joins 9-10-12-14-15-16	1.1429
13	3-7 and 1-2-6-8-4	5 and 1-2-6-8-4	11 and 1-2-6-8-4	-	-	11 joins 1-2-6-8-4	1
14	3-7 and 1-2-6-8-4-11	5 and 1-2-6-8-4-11	9-10-12-14-15-16-18 and 13	9-10-12-14-15-16-18 and 22	22-24	22-24	1
16	3-7 and 1-2-6-8-4-11	5 and 1-2-6-8-4-11	9-10-12-14-15-16-18 and 13	9-10-12-14-15-16-18 and 23	-	9-10-12-14-15-16-18 joins 23	0.75

Table 4.31: Clustering under 5% disruption

Scenario 2: Complete disruption in links							
Number of iteration	1st best option	2nd best option	3rd best option	4th best option	5th best option	selected	*w*
17	3-7 and 1-2-6-8-4-11	5 and 1-2-6-8-4-11	17-19 and 21	-	-	17-19 joins 21	0.6667
18	3-7 and 1-2-6-8-4-11	5 and 1-2-6-8-4-11	-	-	-	5 joins 1-2-6-8-4-11	0.4286
19	17-19-21 and 20	-	-	-	-	17-19-21 joins 20	0.2500
(cluster 1)	1 2 6 8 4 11 5						0.4286
(cluster 2)	3 7						1
(cluster 3)	9 10 12 14 15 16 18 23						0.6250
(cluster 4)	13						2
(cluster 5)	17 19 21 20						0.25
(cluster 6)	22 24						0.5

Table 4.31: Clustering under 5% disruption

Table 4.32 shows a step wise clustering result of the network with complete disruption in nodes while protecting damage sensitive elements at the 5 percentage of disruption probability level.

Scenario 3: Complete disruption in nodes							
Number of iteration	1st best option	2nd best option	3rd best option	4th best option	5th best option	selected	*w*
1	1-2	2-6	5-8	6-8	-	6-8	3
2	1-2	2 and 6-8	14-15	-	-	14-15	2.5
3	1-2	2 and 6-8	-	-	-	2 joins 6-8	2.33
4	1 and 2-6-8	3 and 2-6-8	-	-	-	3 joins2-6-8	2
5	1 and 2-6-8-3	9 and 2-6-8-3	9-10	10-12	-	10-12	2
6	1 and 2-6-8-3	9 and 2-6-8-3	10-12 and 14-15	14-15 and 16	-	14-15 joins 16	2
7	1 and 2-6-8-3	9 and 2-6-8-3	10-12 and 14-15-16	18 and 14-15-16	18-19	18-19	2
8	1 and 2-6-8-3	9 and 2-6-8-3	10-12 and 14-15-16	-	-	10-12 joins 14-15-16	1.60
9	1 and 2-6-8-3	9 and 2-6-8-3	10-12-14-15-16 and 18-19	-	-	10-12-14-15-16 joins 18-19	1.4286

Table 4.32: Clustering under 5% disruption

Scenario 3: Complete disruption in nodes							
Number of iteration	1st best option	2nd best option	3rd best option	4th best option	5th best option	selected	*w*
10	1 and 2-6-8-3	9 and 2-6-8-3	-	-	-	9 joins 2-6-8-3	1.40
11	1 and 2-6-8-3-9	2-6-8-3-9 and 11	20 and 10-12-14-15-16-18-19	-	-	20 joins 10-12-14-15-16-18-19	1.25
12	1 and 2-6-8-3-9	2-6-8-3-9 and 11	-	-	-	2-6-8-3-9 joins 11	1.1667
13	1 and 2-6-8-3-9-11	4-7	-	-	-	4-7	1
14	1 and 1-2-6-8-3-9-11	22-24	-	-	-	22-24	1
15	1 and 2-6-8-3-9-11	-		-	-	1 joins 2-6-8-3-9-11	0.7143
16	1-2-6-8-3-9-11and 5	22-24 and 23	-	-	-	22-24 joins 23	0.6667
17	1-2-6-8-3-9-11and 5	-	-	-	-	1-2-6-8-3-9-11joins 5	0.3750
(cluster 1)	1 2 6 8 3 9 11 5						0.3750
(cluster 2)	4 7						1
(cluster 3)	10 12 14 15 16 18 19 20						0.8750
(cluster 4)	13						2
(cluster 5)	17						2
(cluster 6)	21						1
(cluster 7)	22 24 23						0.3333

Table 4.32: Clustering under 5% disruption

Table 4.33 shows a step wise clustering result of the network with complete disruption in both nodes and links while protecting damage sensitive elements at the 5 percentage of disruption probability level.

Scenario 4: Complete disruption in links and nodes							
Number of iteration	1st best option	2nd best option	3rd best option	4th best option	5th best option	selected	*w*
1	1-2	6-8	-	-	-	6-8	2.5
2	1-2	14-15	-	-	-	14-15	2.5
3	1-2	-	-	-	-	1-2	2
4	1-2 and 3	1-2 and 6-8	3 and 6-8	-	-	3 joins 6-8	2
5	1-2 and 3-6-8	3-6-8 and 9	4-7	10-12	-	10-12	2
6	1-2 and 3-6-8	3-6-8 and 9	4-7	10-12 and 14-15	14-15 and 16	14-15 joins 16	2
7	1-2 and 3-6-8	3-6-8 and 9	4-7	10-12 and 14-15-16	-	10-12 joins 14-15-16	1.60
8	1-2 and 3-6-8	3-6-8 and 9	4-7	-	-	4-7	1.50
9	1-2 and 3-6-8	3-6-8 and 9	10-12-14-15-16 and 9	17-19	-	17-19	1.5
10	1-2 and 3-6-8	3-6-8 and 9	10-12-14-15-16 and 9	-	-	10-12-14-15-16 joins 9	1.333
11	1-2 and 3-6-8	3-6-8 and 5	-	-	-	3-6-8 joins 5	1.25
12	1-2 and 3-6-8-5	1-2 and 4-7	9-10-12-14-15-16 and 13	9-10-12-14-15-16 and 17-19	9-10-12-14-15-16 and18	9-10-12-14-15-16 joins18	1.1429
13	1-2 and 3-6-8-5	1-2 and 4-7	9-10-12-14-15-16-18 and 22	22-24	-	22-24	1
14	1-2 and 3-6-8-5	1-2 and 4-7	9-10-12-14-15-16-18 and 23	-	-	9-10-12-14-15-16-18 joins 23	0.8750
15	1-2 and 3-6-8-5	1-2 and 4-7	-	-	-	1-2 joins 4-7	0.75
16	1-2 and 3-6-8-5	17-19 and 21	-	-	-	17-19 joins 21	0.6667
17	1-2 and 3-6-8-5	17-19-21and 20	-	-	-	17-19- 21 joins 20	0.25
18	1-2 and 3-6-8-5	-	-	-	-	1-2 joins 3-6-8-5	0.125
(cluster 1)	1 2 4 7 3 6 8 5						0.125
(cluster 2)	9 10 12 14 15 16 18 23						0.75
(cluster 3)	11						1
(cluster 4)	13						2
(cluster 5)	17 19 21 20						0.25
(cluster 7)	22 24						0.5

Table 4.33: Clustering under 5% disruption

Table 4.34 shows a step wise clustering result of the network with partial disruption in links while protecting damage sensitive elements at the 5 percentage of disruption probability level.

Scenario 5: Partially disrupted links							
Number of iteration	1st best option	2nd best option	3rd best option	4th best option	5th best option	selected	*w*
1	1-2	5-8	6-8	-	-	6-8	3
2	1-2	2 and 6-8	14-15	-	-	14-15	2.5
3	1-2	2 and 6-8	-	-	-	2 joins 6-8	2.33
4	1 and 2-6-8	-	-	-	-	1 joins 2-6-8	2
5	3 and 1-2-6-8	4 and 1-2-6-8	7 and 1-2-6-8	3-7	-	3-7	2
6	3-7 and 1-2-6-8	4 and 1-2-6-8	9-10	10-12	-	10-12	2
7	3-7 and 1-2-6-8	4 and 1-2-6-8	10-12 and 14-15	14-15 and 16	-	16 joins 14-15	2
8	3-7 and 1-2-6-8	4 and 1-2-6-8	10-12 and 14-15-16	18 and 14-15-16	18-19	18 -19	2
9	3-7 and 1-2-6-8	4 and 1-2-6-8	10-12 and 14-15-16	-	-	10-12 joins 14-15-16	1.60
10	3-7 and 1-2-6-8	4 and 1-2-6-8	10-12-14-15-16 and 18-19	-	-	10-12-14-15-16 joins 18-19	1.4286
11	3-7 and 1-2-6-8	4 and 1-2-6-8	-	-	-	4 joins 1-2-6-8	1.40
12	3-7 and 1-2-6-8-4	5 and 1-2-6-8-4	9 and 1-2-6-8-4	9 and 10-12-14-15-16-18-19	-	9 joins 10-12-14-15-16-18-19	1.25
13	3-7 and 1-2-6-8-4	5 and 1-2-6-8-4	11 and 1-2-6-8-4	-	-	11 joins 1-2-6-8-4	1
14	3-7 and 1-2-6-8-4-11	5 and 1-2-6-8-4-11	20-21	-	-	20-21	1
15	3-7 and 1-2-6-8-4-11	5 and 1-2-6-8-4-11	22-24	-	-	22-24	1
16	3-7 and 1-2-6-8-4-11	5 and 1-2-6-8-4-11	23 and 22-24	-	-	23 joins 22-24	0.6667
17	3-7 and 1-2-6-8-4-11	5 and 1-2-6-8-4-11	-	-	-	5 joins 1-2-6-8-4-11	0.4286
(cluster 1)	1 2 6 8 4 11 5						0.4286
(cluster 2)	3 7						1
(cluster 3)	9 10 12 14 15 16 18 19						0.8750
(cluster 4)	13						2
(cluster 5)	17						2
(cluster 6)	20 21						0.5
(cluster 7)	22 24 23						0.3333

Table 4.34: Clustering under 5% disruption

Table 4.35 shows a step wise clustering result of the network with partial disruption in nodes while protecting damage sensitive elements at the 5 percentage of disruption probability level.

Scenario 6: Partially disrupted nodes							
Number of iteration	1st best option	2nd best option	3rd best option	4th best option	5th best option	selected	*w*
1	1-2	5-8	6-8	-	-	6-8	3
2	1-2	2 and 6-8	14-15	-	-	14-15	2.5000
3	1-2	2 and 6-8	-	-	-	2 joins 6-8	2.3333
4	1 and 2-6-8	-	-	-	-	1 joins 2-6-8	2
5	3 and 1-2-6-8	4 and 1-2-6-8	7 and 1-2-6-8	3-7	-	3-7	2
6	3-7 and 1-2-6-8	4 and 1-2-6-8	9-10	10-12	-	10-12	2
7	3-7 and 1-2-6-8	4 and 1-2-6-8	10-12 and 14-15	14-15 and 16	-	14-15 joins 16	2
8	3-7 and 1-2-6-8	4 and 1-2-6-8	10-12 and 14-15-16	14-15-16 and 18	18-19	18-19	2
9	3-7 and 1-2-6-8	4 and 1-2-6-8	10-12 and 14-15-16	-	-	10-12 joins 14-15-16	1.6000
10	3-7 and 1-2-6-8	4 and 1-2-6-8	18-19 and 10-12-14-15-16	-	-	18-19 joins 10-12-14-15-16	1.4286
11	3-7 and 1-2-6-8	4 and 1-2-6-8	-	-	-	4 joins 1-2-6-8	1.4000
12	3-7 and 1-2-6-8-4	5 and 1-2-6-8-4	9 and 1-2-6-8-4	9 and 10-12-14-15-16-18-16	-	9 joins 10-12-14-15-16-18-16	1.2500
13	3-7 and 1-2-6-8-4	5 and 1-2-6-8-4	11 and 1-2-6-8-4	-	-	11 joins 1-2-6-8-4	1
14	3-7 and 1-2-6-8-4-11	5 and 1-2-6-8-4-11	20-21	-	-	20-21	1
15	3-7 and 1-2-6-8-4-11	5 and 1-2-6-8-4-11	22-24	-	-	22-24	1
16	3-7 and 1-2-6-8-4-11	5 and 1-2-6-8-4-11	22-24 and 23	-	-	22-24 joins 23	0.6667
17	3-7 and 1-2-6-8-4-11	5 and 1-2-6-8-4-11	-	-	-	5 joins 1-2-6-8-4-11	0.4286
(cluster 1)	1 2 6 8 4 11 5						0.4286
(cluster 2)	3 7						1
(cluster 3)	9 10 12 14 15 16 18 19						0.8750
(cluster 4)	13						2
(cluster 5)	17						2
(cluster 6)	20 21						0.5
(cluster 7)	22 24 23						0.3333

Table 4.35: Clustering under 5% disruption

Table 4.36 shows a step wise clustering result of the network with partial disruption in both links and nodes while protecting damage sensitive elements at the 5 percentage of disruption probability level.

Scenario 7: Partially disrupted nodes and links							
Number of iteration	1st best option	2nd best option	3rd best option	4th best option	5th best option	selected	*w*
1	1-2	5-8	6-8	-	-	6-8	3
2	1-2	2 and 6-8	14-15	-	-	14-15	2.5000
3	1-2	2 and 6-8	-	-	-	2 joins 6-8	2.3333
4	1 and 2-6-8	-	-	-	-	1 joins 2-6-8	2
5	3 and 1-2-6-8	4 and 1-2-6-8	7 and 1-2-6-8	3-7	-	3-7	2
6	3-7 and 1-2-6-8	4 and 1-2-6-8	9-10	10-12	-	10-12	2
7	3-7 and 1-2-6-8	4 and 1-2-6-8	10-12 and 14-15	14-15 and 16	-	14-15 joins 16	2
8	3-7 and 1-2-6-8	4 and 1-2-6-8	10-12 and 14-15-16	14-15-16 and 18	18-19	18-19	2
9	3-7 and 1-2-6-8	4 and 1-2-6-8	10-12 and 14-15-16	-	-	10-12 joins 14-15-16	1.6000
10	3-7 and 1-2-6-8	4 and 1-2-6-8	18-19 and 10-12-14-15-16	-	-	18-19 joins 10-12-14-15-16	1.4286
11	3-7 and 1-2-6-8	4 and 1-2-6-8	-	-	-	4 joins 1-2-6-8	1.4000
12	3-7 and 1-2-6-8-4	5 and 1-2-6-8-4	9 and 1-2-6-8-4	9 and 10-12-14-15-16-18-16	-	9 joins 10-12-14-15-16-18-16	1.2500
13	3-7 and 1-2-6-8-4	5 and 1-2-6-8-4	11 and 1-2-6-8-4	-	-	11 joins 1-2-6-8-4	1.1667
14	3-7 and 1-2-6-8-4-11	5 and 1-2-6-8-4-11	20-21	-	-	20-21	1
15	3-7 and 1-2-6-8-4-11	5 and 1-2-6-8-4-11	22-24	-	-	22-24	1

Table 4.36: Clustering under 5% disruption

Scenario 7: Partially disrupted nodes and links							
Number of iteration	1st best option	2nd best option	3rd best option	4th best option	5th best option	selected	*w*
16	3-7 and 1-2-6-8-4-11	5 and 1-2-6-8-4-11	22-24 and 23	-	-	22-24 joins 23	0.6667
17	3-7 and 1-2-6-8-4-11	5 and 1-2-6-8-4-11	-	-	-	5 joins 1-2-6-8-4-11	0.4286
(cluster 1)	1 2 6 8 4 11 5						0.4286
(cluster 2)	3 7						1
(cluster 3)	9 10 12 14 15 16 18 19						0.8750
(cluster 4)	13						2
(cluster 5)	17						2
(cluster 6)	20 21						0.5
(cluster 7)	22 24 23						0.3333

Table 4.36: Clustering under 5% disruption

Table 4.37 shows a step wise clustering result of the network with partial disruption in links and complete disruption in nodes while protecting damage sensitive elements at the 5 percentage of disruption probability level.

Scenario 8: Complete Disruption in nodes and partially disrupted links								
Number of iteration	1st best option	2nd best option	3rd best option	4th best option	5th best option	6th best option	selected	*w*
1	1-2	6-8	-	-	-	-	6-8	2.5
2	1-2	14-15	-	-	-	-	14-15	2.5
3	1-2	-	-	-	-	-	1-2	2
4	3 and 1-2	6-8 and 1-2	3 and 6-8	-	-	-	3 joins 6-8	2
5	3-6-8 and 1-2	9 and 3-6-8	4-7	10-12	-	-	10-12	2
6	3-6-8 and 1-2	9 and 3-6-8	4-7	10-12 and 14-15	14-15 and 16	-	14-15 joins 16	2
7	3-6-8 and 1-2	9 and 3-6-8	4-7	10-12 and 14-15-16	14-15-16 and 18	18-19	18-19	2
8	3-6-8 and 1-2	9 and 3-6-8	4-7	10-12 and 14-15-16	-	-	10-12 and 14-15-16	1.60

Table 4.37: Clustering under 5% disruption

Scenario 8: Complete Disruption in nodes and partially disrupted links								
Number of iteration	1st best option	2nd best option	3rd best option	4th best option	5th best option	6th best option	selected	*w*
9	3-6-8 and 1-2	9 and 3-6-8	4-7	-	-	-	4-7	1.50
10	3-6-8 and 1-2	9 and 3-6-8	9 and 10-12-14-15-16	18-19 and 10-12-14-15-16	-	-	18-19 joins 10-12-14-15-16	1.4286
11	3-6-8 and 1-2	9 and 3-6-8	-	-	-	-	9 joins 3-6-8	1.25
12	3-6-8-9 and 1-2	20 and 10-12-14-15-16-18-19	-	-	-	-	20 joins 10-12-14-15-16-18-19	1.25
13	3-6-8-9 and 1-2	-	-	-	-	-	3-6-8-9 joins 1-2	1
14	1-2-3-6-8-9 and 4-7	1-2-3-6-8-9 and 5	22-24	-	-	-	22-24	1
15	1-2-3-6-8-9 and 4-7	1-2-3-6-8-9 and 5	23 and 22-24	-	-	-	23 joins 22-24	0.6667
16	1-2-3-6-8-9 and 4-7	1-2-3-6-8-9 and 5	-	-	-	-	1-2-3-6-8-9 joins 5	0.4286
(cluster 1)	1 2 3 6 8 9 5							0.4286
(cluster 2)	4 7							1
(cluster 3)	10 12 14 15 16 18 19 20							1
(cluster 4)	11							1
(cluster 5)	13							2
(cluster 6)	17							2
(cluster 7)	21							1
(cluster 8)	22 24 23							0.3333

Table 4.37: Clustering under 5% disruption

Table 4.38 shows a step wise clustering result of the network with partial disruption in nodes and complete disruption in links while protecting damage sensitive elements at the 5 percentage of disruption probability level.

Scenario 9: Complete disruption in links and partially disrupted nodes							
Number of iteration	1st best option	2nd best option	3rd best option	4th best option	5th best option	selected	*w*
1	1-2	5-8	6-8	-	-	6-8	3
2	1-2	2 and 6-8	14-15	-	-	14-15	2.5
3	1-2	2 and 6-8	-	-	-	2 joins 6-8	2.33
4	1 and 2-6-8	-	-	-	-	1 joins 2-6-8	2
5	3 and 1-2-6-8	4 and 1-2-6-8	7 and 1-2-6-8	3-7	-	3-7	2
6	3-7 and 1-2-6-8	4 and 1-2-6-8	9-10	10-12	-	10-12	2
7	3-7 and 1-2-6-8	4 and 1-2-6-8	10-12 and 14-15-16	14-15 and 16	-	14-15 joins 16	2
8	3-7 and 1-2-6-8	4 and 1-2-6-8	10-12 and 14-15-16	14-15-16 and 18	18-19	18-19	2
9	3-7 and 1-2-6-8	4 and 1-2-6-8	10-12 and 14-15-16	-	-	10-12 joins 14-15-16	1.6000
10	3-7 and 1-2-6-8	4 and 1-2-6-8	10-12-14-15-16 and 18-19	-	-	10-12-14-15-16 joins 18-19	1.4286
11	3-7 and 1-2-6-8	4 and 1-2-6-8	-	-	-	4 joins 1-2-6-8	1.40
12	3-7 and 1-2-6-8-4	5 and 1-2-6-8-4	9 and 1-2-6-8-4	10-12-14-15-16-18-19 and 9	-	10-12-14-15-16-18-19 joins 9	1.2500
13	3-7 and 1-2-6-8-4	5 and 1-2-6-8-4	11 and 1-2-6-8-4	-	-	11 joins 1-2-6-8-4	1
14	3-7 and 1-2-6-8-4-11	5 and 1-2-6-8-4-11	20-21	-	-	20-21	1
15	3-7 and 1-2-6-8-4-11	5 and 1-2-6-8-4-11	22-24	-	-	22-24	1
16	3-7 and 1-2-6-8-4-11	5 and 1-2-6-8-4-11	22-24 and 23	-	-	22-24 joins 23	0.6667
17	3-7 and 1-2-6-8-4-11	5 and 1-2-6-8-4-11	-	-	-	5 joins 1-2-6-8-4-11	0.4286
(cluster 1)	1 2 6 8 4 11 5						0.4286
(cluster 2)	3 7						1
(cluster 3)	9 10 12 14 15 16 18 19						0.8750
(cluster 4)	13						2
(cluster 5)	17						2
(cluster 6)	20 21						0.5
(cluster 7)	22 24 23						0.3333

Table 4.38: Clustering under 5% disruption

4. Properties

We refer the reader to properties provided on page 84 and 100 of this study.

4.4. Methods comparison

The results of our stepwise procedure reveal that HRP1$^+$ computes partitioning by assigning nodes to clusters one at a time. This means next cluster is not generated before the completion of the previous cluster. On the other hand, HRP2$^+$ is able to develop multiple clusters at the same time. Therefore it computes partitioning by searching for pair of nodes with the highest density without being constrained to one cluster.

Considering the results from our numerical example in the four previous subsections, we conclude that in general HRP1$^+$leads to less number of edge-cuts than HRP2$^+$ with or without the control step.

As a matter of fact the advantage of the control strep, as previously mentioned, is that damage sensitive elements are protected in favor of system resiliency. Therefore it maintains connectivity among our ultimate set of partitions in the occurrence of an event.

Chapter 5:

Numerical

Experimentation

In this chapter, two types of experiments are performed. First we will test our two suggested methods ($HRP1^{+}$, $HRP2^{+}$) and their extensions ($HRP1^{+}$control, $HRP2^{+}$control) on randomly generated networks. Second we test them on benchmark datasets available in literature. 5 different graphs selected from "UF Sparse Matrix Collection" are used. After experimental evaluation, a comparative analysis is performed. The four algorithms were coded in Mat lab.

5.1. Randomly generated graphs

Graph of 24 nodes and 30 links was generated at random and the four algorithms were applied. In this section we show results for the second scenario where network is dealing with complete disruption in links. Tables for other scenarios are available in appendix (I). It can be seen from the following tables that as the percentage of disruption increases the number of isolated nodes increases and subsets are unbalanced. However when applying HRP control methods the negativity of disruption leading to disconnected subsets is reduced. Nevertheless the system can maintain its functionality up to a certain level. Here, our first table, table 1.a shows partitioning results of the network with complete link disruption when applying $HRP1^{+}$while accounting for different percentages of disruption.

% of disruption	Complete disruption in links: HRP1^{+}												
	Clusters												q
5%	1,4,7,3,6,2,5,8			9,10,12,13,14,16,17,19				11	15,22,24,23		18,20,21		.25,.375,1,.25,.33
10 %	1,4,7,3,6,2,5,8			9,10,12,13,14,16,17,19				11	15,22,24,23		18,20,21		.25,.375,1,.25,.3333
15 %	1,2,5,6,3,7,4,8			9,10,12,13,14,16,17,19				11	15,22,24,23		18,20,21		.25,.38,1,.25,.33
20 %	1,4,7,3,6,8,11,9			2,5		10,12,14,16,17,13,19,18		15,22,24,23			20,21		.13,0,.38,.25,.5
30 %	1,3,6,7,4,8,11,9			2,5		10,12,14,16,17,13,19,18		15,22,24,23			20,21		.13,0,.38,.25,.5
40 %	1,2		3	4,7,8,5,11,10,12,13			6	9	14,16,17,19,21,20,18			15,22,24,23	0,0,.125,0,0,0,.25

Table 1.a: HRP1^{+} : No node disruption – Complete link disruption

Table 1.b shows partitioning results of the network with complete link disruption when applying HRP1^{+}control while keeping account for different percentages of disruption.

% of disruption	Complete disruption in links: HRP1^{+}control																										q
	Clusters																										
5 %	1,4,7,3,6,2,5,8					9,10,12,13,14,16,18,20				11	15,22,24,23					17		19,21									.1429,.5,1,.25,2,1
10 %	1,4,7,3,6,2,5				8	9,10,11				12,13,15,23,24,22,14					16			17,19,20,21,18									.1423,2,.6667,.2857,2,.2
15 %	1,2,5,6,4		3	7	11		20			8,9,10,12,14,16,17,13					15,22,24,23			18,19,21								.6,2,3,1,1,.5,.25,.3333
20 %	1	2,5,8,11,6,7,4			3	9	10			12,13,15,23,24,22					14	16		17,18,20,21,19								1,.5714,1,2,3,.1667,1,2,.2
30 %	1	2,5,6,3,8,9,11		4	7	10,12,14,15,23,24,22			13		16	17	18,19,21						20								2,.4286,2,2,.2857,0,3,2,1,1
40 %	1,3,6,2,8,11,7		4	5	9	10	12	13	14,15,22,23		16	17	18	19	20	21	24										.1429,0,1,1,2,3,1,.75,3,2,2,2,1,1,1

Table 1.b: HRP1^{+}control: No node disruption – Complete link disruption

Table 1.c shows partitioning results of the network with complete link disruption when applying HRP2^{+}while accounting for different percentages of disruption.

% of disruption	Complete disruption in links: HRP2^{+}	
	Clusters	q
5%	1 2 3 6 8 9 11 5 | 4 7 | 10 12 14 15 16 18 19 20 | 13 | 17 | 21 | 22 24 23	0.3750 1 0.8750 2 2 1 0.3333
10%	1 2 6 8 4 11 3 5 | 7 | 9 10 12 14 15 16 18 19 | 13 | 17 | 20 21 | 22 24 23	0.3750 2 0.75 2 1 0.50 0.3333
15%	1 2 4 7 5 | 3 6 8 11 | 9 10 12 14 15 16 18 22 | 13 | 17 | 19 20 21 | 23 | 24	0.40 0.75 0.8750 2 1 0.3333 1 1
20%	1 3 7 8 9 10 2 11 | 4 | 5 | 6 | 12 14 15 16 17 19 21 13 | 18 20 | 22 24 23	0.7500 2 1 2 0.3750 0.50 0.3333
30%	1 2 5 8 6 | 3 | 4 7 | 9 | 10 12 13 | 11 | 14 | 15 | 16 | 17 19 21 | 18 | 20 | 22 | 23 | 24	0.40 0 1 0 0.3333 0 1 0 0 0.3333 0 1 0 0 0
40%	1 3 6 8 9 10 4 7 | 2 5 | 11 | 12 14 16 18 19 20 21 17 | 13 | 15 23 24 | 22	0.3750 0.5000 1 0.1250 0 0.3333 1

Table 1.c: HRP2^{+} : No node disruption – Complete link disruption

Table 1.d shows partitioning results of the network with complete link disruption when applying HRP2^{+} control while accounting for different percentages of disruption.

% of disruption	Complete disruption in links: $HRP2^{+}$ control										
	Clusters										q
5%	1 2 6 8 4 11 5	3 7	9 10 12 14 15 16 18 19	1 3	17	20 21	22 24 23				0.4286 1 0.8750 2 20.50 0.3333
10%	1 2 6 8 4 9 10 12	3 7	5	1 1	13 15 22 24 14 23	16 18 19 20 21	17				0.8750 1 2 2 0.1667 0.40 2.0
15%	1 2 3 7 8 9 11 4	5	6	10 12 14 15 16 18 19 20	13	17	21	22	23	24	0.6250 2 2 1 2 2 1 1 1 0
20%	1 2 6 8 7 11 3 4	5	9 10 12 14 15 16 18 20	13	17 19	21	22 24 23				0.3750 2 0.50 2 0.50 1 0.3333
30%	1 2 6 8 7 9 3 4	5	10 12 14 15 16 17 23 22	1 1	1 3	18 19 21 20	2 4				0.25 2 0.6250 1 2 0.25 1
40%	1 2 4 7 5	3 6 8 11 9 10	12 13 15 23 14 16 24	17 19 20 21 18	2 2						0.4 0.5 0.2857 0.2 0

Table 1.d: $HRP2^{+}$ control: No node disruption – Complete link disruption

5.2. Benchmark dataset

5.2.1. Datasets

For benchmarking purposes 5 graphs were selected from a set of large and actively growing sparse matrices available on the University of Florida collection arising in real applications. Each network comes descriptively with classified information based on the user's requirement. Here we will display images of the networks and their graphical matrix as well as their basic data required in order to apply our proposed techniques.

Our first network, Dwt_66, consists of 66 nodes and 320 links

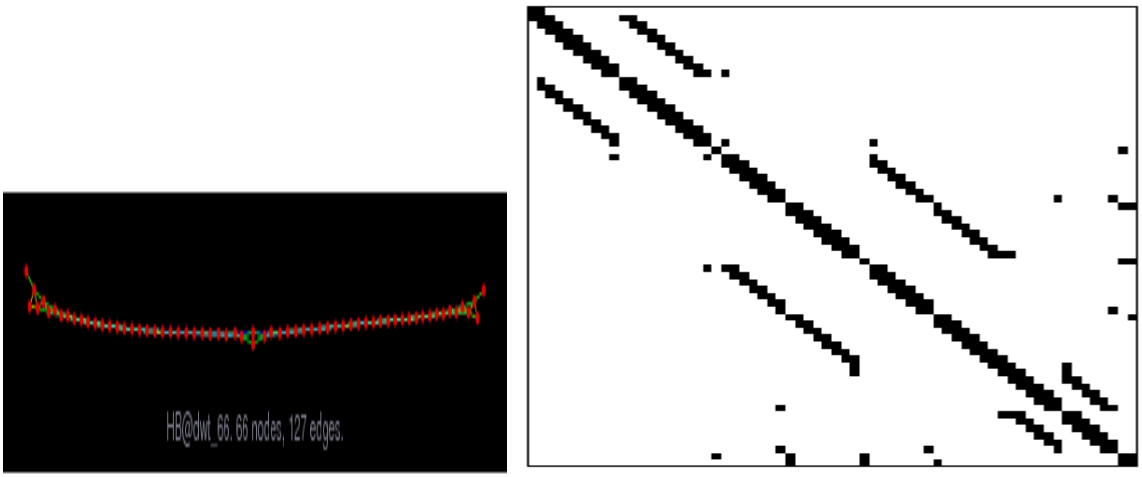


Figure 12: Dwt_66

Our second network, GD96_b, contains 111 nodes and 193 links:

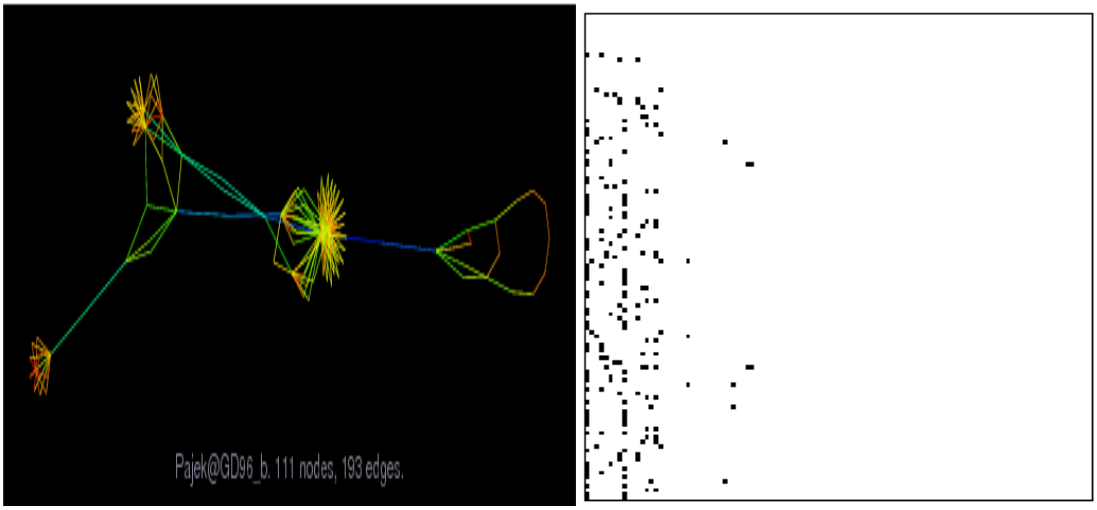


Figure 13: GD96_b

Our third network, Dwt_221,is a network of 221 nodes and 1629 arcs:

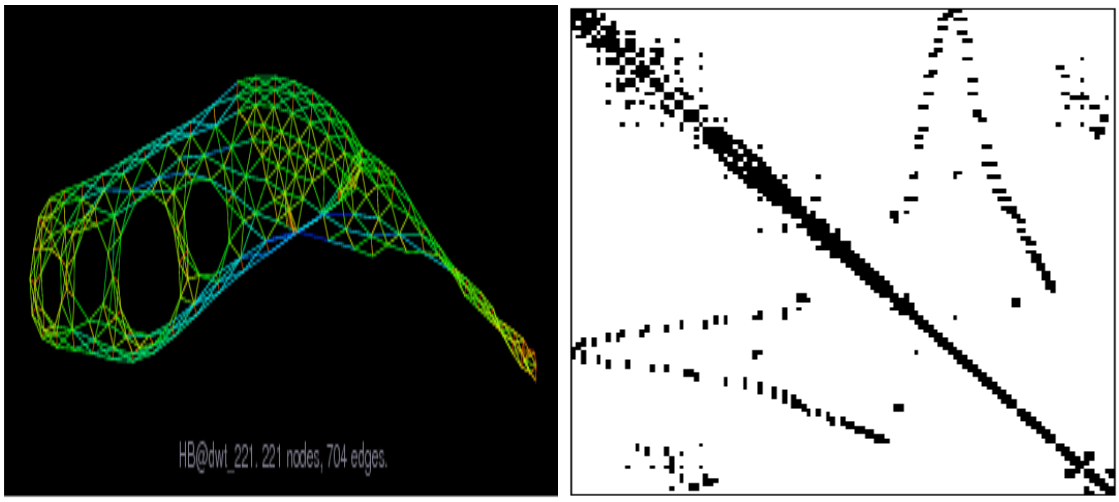


Figure 14: Dwt_221

The fourth network consists of 310 nodes and 2448 links:

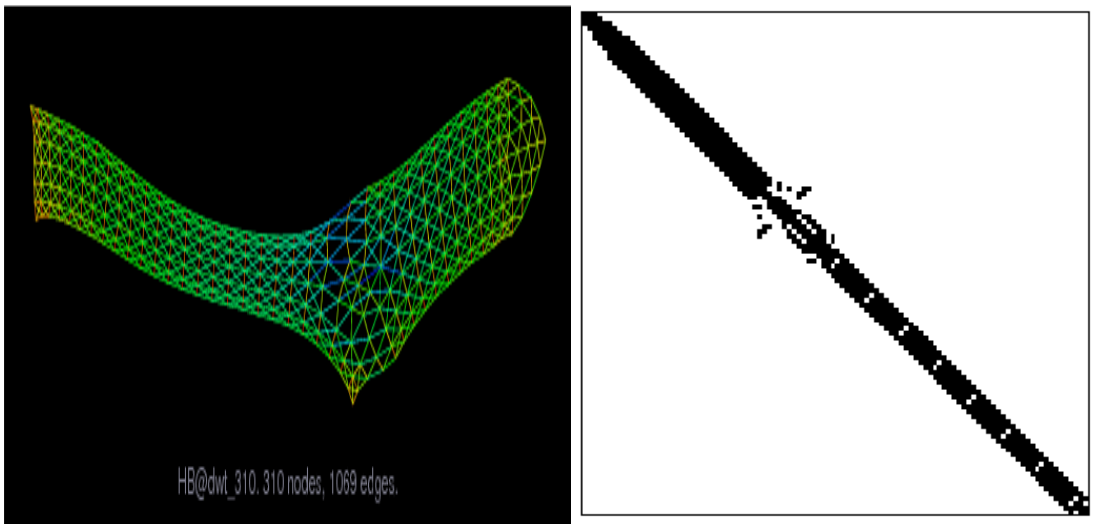


Figure 15: Dwt_310

Our last studied case, Dwt_419, is a network of 419 nodes and 3563links:

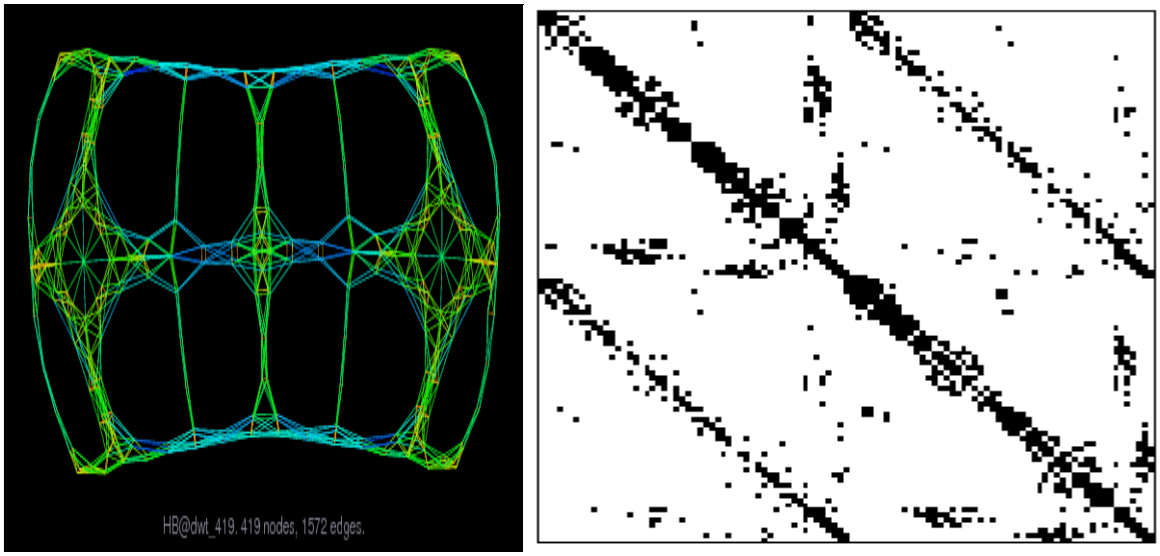


Figure 16: Dwt_419

Table below, specifies our input criteria, *W* and *Q*, for each network:

	Nodes	Links	*W*	*Q*
Network Name: dwt_66	66	320	9	0.125
Network Name: GD96_b	111	193	32	0.1
Network Name: dwt_221	221	1629	20	0.4
Network Name: dwt_310	310	2448	20	0.1
Network Name: dwt_419	419	3563	30	0.1

Table 5: Network properties

5.2.2. Results

The results of our experiments are contained in 5 separate tables (tables 5.3.1-5.3.5) with information on number of edge-cuts (EC), number of clusters(C) and their execution runtime (T) for different scenarios in each method. We investigate approaches when a 20 percentage of disruption in the network takes place. The tables are ordered by the magnitude of the network, keeping our user defined criteria constant in favor of solely comparing the efficiency of our methods and the impact of our control step in a real test-bed.

In table 5.3.1, HRP1$^+$ has a shorter run time period compared to HRP2$^+$. Looking at the results of scenario 1where the network is in its original state we find that the number of clusters obtained from HRP1$^+$ is lower than HRP2$^+$ and also lower number of edge cuts which means less communication between subsets. Considering the properties of network "dwt_66" we see that the network has been clustered evenly when applying HRP1$^+$ or HRP1$^+$control compared to the latter method.

Network Name: dwt_66	*HRP1*$^+$			*HRP1*$^+$CTRL			*HRP2*$^+$			*HRP2*$^+$CTRL		
	EC	T	C	EC	T	C	EC	T	C	EC	T	C
scenario 1	14	0.364269	8	14	0.499139	8	20	1.046194	11	20	1.175877	11
scenario 2	14	0.309937	8	14	0.320392	8	20	1.005525	11	24	0.964988	13
scenario 3	16	0.27981	11	10	0.278608	10	18	0.837592	15	24	0.840339	14
scenario 4	12	0.270214	12	12	0.262729	13	20	0.833734	16	24	0.821086	15
scenario 5	20	0.32052	9	18	0.311333	10	22	1.008775	11	20	0.996996	11
scenario 6	26	0.285595	12	28	0.298734	12	24	0.983296	13	18	1.005252	10
scenario 7	26	0.286848	12	34	0.284347	15	28	0.979564	14	18	0.988724	10
scenario 8	12	0.26412	12	18	0.261247	14	20	0.833907	16	24	0.83501	14
scenario 9	26	0.283433	12	28	0.314438	12	22	0.975434	12	18	1.003802	10

Table 5.3.1: Experimentation results for dwt_66

In table 5.3.2 the number of clusters is noticeably higher when applying HRP2$^+$or its extension. This implies that HRP1$^+$is a more practical and applicable scheme for networks with different properties.

Network Name:GD96_b	$HRP1^+$			$HRP1^+$CTRL			$HRP2^+$			$HRP2^+$CTRL		
	EC	T	C	EC	T	C	EC	T	C	EC	T	C
Senario1	7	1.23011	5	7	1.281785	5	79	1.263312	80	79	1.35105	80
Senario2	12	1.130326	11	9	1.139839	8	77	1.107098	80	79	1.002849	80
Senario3	6	0.717735	15	7	0.737094	16	73	1.085395	80	67	1.097984	72
Senario4	6	1.118549	14	8	1.0961	9	73	1.888434	80	64	1.091873	70
Senario5	8	1.184167	6	7	1.235409	5	79	0.969039	80	79	0.990126	80
Senario6	8	1.227042	6	7	1.172727	5	79	1.03994	80	79	0.989257	80
Senario7	8	1.219729	6	7	1.1671	5	79	0.985315	80	79	0.984069	80
Senario8	6	1.027051	11	7	1.130391	8	74	0.889942	80	67	1.075683	71
Senario9	9	1.142831	7	7	1.234013	5	79	0.971419	80	64	1.19959	65

Table 5.3.2: Experimentation results for GD96_b

In table 5.3.3 the runtime for HRP2$^+$ and its extension is considerably greater than HRP1$^+$. This implies that HRP1$^+$ is more time efficient. Also considering the properties of network "dwt_221" we see that the resulting subsets are more uniform when applying HRP1$^+$ or HRP1$^+$control compared to the latter method.

Network Name:dwt_221	$HRP1^+$			$HRP1^+$CTRL			$HRP2^+$			$HRP2^+$CTRL		
	EC	T	C	EC	T	C	EC	T	C	EC	T	C
Senario1	56	20.91748	14	56	22.09931	14	62	78.73637	16	62	79.44794	16
Senario2	56	20.68356	14	56	21.02643	14	72	80.08517	19	76	81.75892	18
Senario3	52	19.6509	18	60	20.60258	16	64	77.99255	23	69	77.76762	18
Senario4	68	20.0806	19	59	20.27924	16	58	75.00573	23	66	79.1284	19
Senario5	60	24.32281	15	60	20.9834	15	62	83.79591	16	62	82.33762	16
Senario6	70	20.72931	17	64	20.8376	16	72	78.32173	20	62	80.28499	16
Senario7	80	20.84777	19	68	20.92525	17	76	78.37606	18	62	80.05695	16
Senario8	58	19.74654	17	66	20.55445	16	60	75.54441	22	70	76.35257	18
Senario9	70	21.02887	17	64	20.70675	16	62	79.29462	18	62	81.04738	16

Table 5.3.3: Experimentation results for dwt_221

Table 5.3.4 shows a high computational time when applying HRP2$^+$ or HRP2$^+$control. Also the evenness of clusters obtained from HRP2$^+$ is lower than HRP1$^+$implying somewhat equal fraction of workload.

Network Name:dwt_310	$HRP1^+$			$HRP1^+$CTRL			$HRP2^+$			$HRP2^+$CTRL		
	EC	T	C	EC	T	C	EC	T	C	EC	T	C
Senario1	72	95.51849	16	72	89.07549	16	84	415.8814	20	84	462.427	20
Senario2	78	87.00209	17	72	90.55361	16	94	362.1133	22	88	427.1849	21
Senario3	74	84.95539	22	68	91.25897	16	92	334.4588	25	88	454.5952	21
Senario4	78	84.70753	22	70	92.1976	16	90	335.8227	25	84	447.1835	23
Senario5	88	86.09844	19	72	88.81553	16	84	347.3733	20	84	362.0947	20
Senario6	108	85.02189	23	72	94.30971	16	84	347.0726	20	90	356.6155	21
Senario7	98	87.07642	21	72	88.18089	16	80	347.1723	20	90	357.8399	21
Senario8	78	98.22717	22	69	88.84786	16	90	339.5932	25	96	353.2139	21
Senario9	105	100.5631	23	70	90.47033	16	96	345.2952	21	94	357.1828	21

Table 5.3.4: Experimentation results for dwt_310

Table 5.3.5 also shows how HRP1$^+$ obtain results in more reasonable amount of time .while returning balanced subset with less communication volume.

Network Name:dwt_419	$HRP1^+$			$HRP1^+$CTRL			$HRP2^+$			$HRP2^+$CTRL		
	EC	T	C	EC	T	C	EC	T	C	EC	T	C
Senario1	60	469.1994	15	60	404.3159	15	88	1481.506	22	88	1827.263	22
Senario2	60	470.9417	15	60	405.7485	15	88	1517.069	19	84	1631.449	19
Senario3	70	478.2006	20	60	410.5418	15	94	1809.647	25	76	1657.697	21
Senario4	64	474.5634	21	60	588.553	15	96	1780.667	23	96	1616.728	20
Senario5	72	498.2138	18	60	454.4045	15	88	2325.846	22	88	1830.702	20
Senario6	76	446.0309	20	60	453.7501	15	98	1778.197	20	88	1591.643	19
Senario7	92	422.7026	23	60	503.5079	15	98	1639.467	20	88	1533.68	19
Senario8	74	391.7177	24	66	429.9015	17	90	1594.045	25	86	1533.68	18
Senario9	76	421.5906	20	60	449.2944	15	86	1608.51	20	88	1551.944	17

Table 5.3.5: Experimentation results for dwt_419

Considering all the above tables in this section, we can see that as the network grows bigger the computational time for HRP2$^+$ with and without the control step is considerably longer than HRP1$^+$, as the algorithmic complexity of HRP2$^+$ is higher, therefore we can conclude that HRP1$^+$ runs faster. In terms of partitioning set for each method we notice that HRP1$^+$ returns smaller number of clusters compared to HRP2$^+$. As for the communication volume between clusters, we can see that HRP1$^+$ has created lower edge cuts meaning that the inter connection is lower and sub networks will be more independent when taking out their tasks. This is due to the density criteria used in HRP1$^+$ when combining subsets. Also the results reveal a balanced and uniform set of clusters when applying HRP1$^+$.Further, when considering the control step in our calculation, the evenness of our partitions increases even though the execution time would take longer compared to when we solely run HRP.

Chapter 6:
Future works and Conclusions

6.1. Conclusions

Graph partitioning is an optimization problem that divides a network into subsets of manageable size with less complexity and minimum interaction, with the aim of maximizing their independency as much as possible.

In this research, with respect to dynamic portioning, we investigated the applicability of four heuristic -based approaches for network partitioning called HRP1^{+} , HRP2^{+}, HRP1^{+} with control and HRP2^{+}with control for a large scale network. In addition, we extended our algorithms by integrating disruption modeling into our methods in favor of handling disruptive events by post corrective actions leading to system recovery and well performance meanwhile and after disruption. In this sense we provided a quick automated coping strategy in the existence of abnormalities that takes the given graph and returns the modified graph without the missing nodes, broken links and outdated elements. Our methods take the adapted graph as the input and produces partitioning. For this matter we re-clustered the network through two different approaches namely “complete failure update “and “partial failure update” based on damage severity. We considered nine possible scenarios specifying a possibility of combination of node and link disruption. Furthermore in order to ensure connectivity between the ultimate subsets we developed a control and checking step.

We demonstrated the necessity of the control and checking step by comparing the methods executed for both circumstances.

To fully understand the efficiency of our suggested techniques each case was studied and tested by various percentage of disruption on a randomly generated network. We as well tested our solution on several networks with different properties chosen from benchmark dataset in order to study its complexity and the time it takes to perform and resolve the problem.

From the observations we can conclude that HRP1^{+} approach achieves a better result for dense and large scale networks. Nevertheless considering the control step helps obtaining a practical result and improves system functionality by maintaining connectivity between the subsets.

However a model capable of managing and optimizing complex systems calls for a wide fundamental change in the perspective of the company. Not only logisticians and operators need to level up their envision with a broader insight of how a decision in one section can have a great impact on the whole system but must also acquire sufficient technological and analytical competence in order to firstly fulfill communication requirements and secondly understand and interpret data such that the system evaluation and improvement can ultimately pursuit its effective functionality. All in all we assume that the system has a well-defined educational agenda for training its personnel in the areas of skill and competency.

6.2. Future work

In this study we investigated and compared the two methods and their extensions with each other. We can also consider comparing our results with other established methods by applying them on the same network. Further since we set our criteria constant, we can generate different set of experimentations by alternatively changing size and density criteria in order to understand their influence on the evenness and communication volume.

In addition we can include an estimate restoration time that can be employed when practicing partial disruption, that is, the assigned value can be based on this estimate rather than considering a fixed value for all the troubled links.

References

Aftosmis, M. J., M. J. Berger, and S. M. Murman. "Applications of Space-Filling Curves to Cartesian Methods for CFD." 42ND Aerospace Sciences Meeting and Exhibit. National Center for Atmospheric Research, 2004.

Andersen, Reid, Fan Chung, and Kevin Lang. "Local Graph Partitioning using PageRank Vectors." San Diego: University of California, 2006, 20.

Anderson, William P., Hanna Maoh, and Charles Burke. "assessing risk and resilience for transportation infrastructure in canada ." Ontario: University of Windsor, 2011.

"army study guide." AR 711-7, 18. n.d.

Awasthi, Anjali, SS Chauhan, M Parent, and Jean-Marie Proth. "Algorithms for partitioning of large routing networks." Journal of the Operational Research Society, 2009: 1159–1167.

Bondy, J.A., and S.R Murty. "graph theory with applications." ontario,canada: department of combinatorics and optimization,university of waterloo, 1976.

Catalyurek, U., and C. Aykanat. "Hypergraph-partitioning based decomposition for parallel sparse-matrix vector multiplication." IEEE Transactions on Parallel and Distributed Systems, 1999: 673-693.

Chamberlain, Bradford L. "Graph Partitioning Algorithms for DistributingWorkloads of Parallel Computations." October 13, 1998.

Dinh, Thang N., Ying Xuan, My T. Thai, E.K. Park, and Taieb Znati. "On Approximation of New Optimization Methods for Assessing Network Vulnerability." Florida, 2009.

Drid, Hamza, Bernard Cousin, Miklos Molnar, and Nasir Ghani. "Graph Partitioning for Survivability in Multi-Domain Optical Networks." Rennes: University of Rennes, 2010.

Erjongmanee, Supaporn, Chuanyi Ji, Jere Stokely, and Neale Hightower. "Inference of Network-Service Disruption upon Natural Disasters." atlanta, 2008.

Gilbert, J. , G. Miller, and S. Teng. "Geometric mesh partitioning: Implementation and experiments." In Proceedings of International Parallel Processing Symposium, 1995: 418-427.

Gonina, Katya, Sayak Ray, and Bor-Yiing Su. "Graph Partitioning Implementation Strategy Pattern." Devender Jain, Sep 02, 2013.

h, Djidjev, and Gilbert J. "separators in graphs with negative and multiple vertex weights." Dep.of computer science,Rice University, 1994.

Hua, Zhihua, Yongsheng Ding, and Qing Shao. "Immune co-evolutionary algorithm based partition balancing optimization." Shanghai, 2008.

Huang, Yiyi, Nick Feamster, Anukool Lakhina, and Jun (Jim) Xu. "Diagnosing Network Disruptions with Network Wide Analysis." Georgia, 2007.

"Inference of Network-Service Disruption upon Natural Disasters." Atlanta, n.d.

Joint Publication 1-02. "The Dictionary of Military and Associated Terms,." Joint Publication 1-02. April 12, 2001.

Karypis, G., and V. Kumar. "Multilevel k-way partitioning scheme for irregular graphs." Journal of Parallel and Distributed Computing, 1998.

Karypis, G., and V. Kumar. "A fast and high quality multilevel scheme for partitioning irregular graphs." SIAM Journal, 1998: 359-392.

Karypis, G., and V. Kumar. "Multilevel algorithms for multi-constraint graph partitioning." 1998.

Madry, Aleksander. "Fast Approximation Algorithms for Cut-based Problems in Undirected Graphs." Massachusetts: Society for Industrial and Applied Mathematics, 2010.

Meyerhenke, Henning, Burkhard Monien, and Thomas Sauerwald. "A New Diffusion-based Multilevel Algorithm for Computing Graph Partitions of Very High Quality." Fuerstenallee,Paderborn: University of Paderborn, 2008.

Mieghem, P. Van, et al. "A Framework for Computing Topological Network Robustness." 2010.

Miller, Gary L., Shang Teng, William Thurston, and Stephen A. Vavasis. "Automatic mesh partitioning." In Graph Theory and Sparse Matrix Computation, by Alan George, John R. Gilbert and Joseph W. H. Liu, 57-84. 1993.

Oliker, L., and R. Biswas. "Parallel load balancing for adaptive unstructured meshes." Journal of Parallel and Distributed Computing, 1998: 150-177.

Riskviews: "Commentary on Risk and ERM ." january 24, 2013.

Roy, Sandip, Lesieutre, and Bernie. "Studies in Network Partitioning Based on Topological." Massachusetts: Massachusetts Institute of Technology, 2003.

Schloegel, K., G. Karpis, and V. Kumar. "A new algorithm for multi-objective graph partitioning." Euro-Par . 1999. 322-331.

Schloegel, K., G. Karypis, and V. Kumar. "Parallel multilevel algorithms for multi-constraint graph partitioning." Minnesota: Dept. of Computer Science and Engineering, Univ. of Minnesota, 1999.

Schloegel, Kirk, George Karypis, and Vipin Kumar. "Graph Partitioning for High Performance Scientific Simulations."Minneapolis: Army HPC Research Center, 2006.

Schloegel, Kirk, George Karypis, and Vipin Kumar. "Wavefront Diffusion and LMSR: Algorithms for Dynamic Repartitioning of Adaptive Meshes ." IEEE Transactions on Parallel and Distributed Systems, 1998.

Snyder, Lawrence V., Paola M. Scaparra, Mark S. Daskin, and Richard L.Church. "Planning for Disruptions in Supply Chain." New Orleans: tutorial, 2005.

Sterbenz, James P.G, Egemen K. Çetinkaya, mahmood A.Hameed, Abdul Jabbar, Shi Qian, and Justin P. Rohrer. "Evolution of network resilience survivability and disruption tolerance." Kansas: The University of Kansas, 2012.

Teresco, James D., Karen D. Devine, Flaherty, and Joseph E. "Partitioning and Dynamic Load Balancing for the Numerical Solution of Partial Differential Equations." In Numerical Solution of Partial Differential Equations on Parallel Computers, pp 55-88. Williamstown, MA, 01267, USA: Springer Berlin Heidelberg, 2006.

Appendix (I)

Numerical Experimentation

Table (continued)

- Applying the four methods namely HRP1^{+}, HRP2^{+}, HRP1^{+} control, and HRP2^{+}control methods on a network encountering complete disruption in nodes

% of disruption	Complete disruption in nodes: HRP1^{+}	
	Clusters	q
5%	1,4,7,3,6,2,5,8 | 9,10,11 | 12,14,16,17,19,20,21,13 | 15,22,24,23 | 18	.125,.667,.25,.25,0
10%	1 | 2,5,6,3,8,11,9,10 | 4 | 7 | 12,14,16,17,18,20,21,19 | 13,15,22,24,23	0,.125,0,0,.25,.2
15%	1,2,6 | 3 | 4 | 5 | 7 | 8 | 9,10,11,12,13,14,16,17 | 15,22,24,23 | 18,20,21,19	0,0,0,0,0,0,.25,.25,.25
20%	1,2,6,4,7 | 3 | 5 | 8 | 9 | 10,12,13,14,16,17,19,21 | 11 | 15,22,24,23 | 18,20	0,0,0,0,0,.25,0,.25,.5
30%	1 | 2,5,8,11,6,3 | 4 | 7 | 9 | 10 | 12,14,13,15,23,24,22 | 16 | 17,19,18,21,20	0,0,0,0,0,0,0,0,0
40%	1,2,6,5 | 3 | 4 | 7 | 8 | 9 | 10 | 11 | 12,14,16,17,19,21,15,13 | 18 | 20 | 22 | 23 | 24	0,0,0,0,0,0,0,0,0,0,0,0,0,0,0

Table 2.a: HRP1^{+} : No link disruption – Complete node disruption

% of disruption	Complete disruption in nodes: HRP1^{+}control	
	Clusters	q
5%	1 4 7 3 6 2 5 8 | 9 10 11 | 12 13 15 22 24 14 16 17 | 18 20 21 19 | 23	0.125 0.6667 0.25 0.25 0
10%	1 2 5 8 10 11 6 3 | 4 7 | 9 | 12 13 15 22 24 14 16 17 | 18 20 21 19 | 23	0.25 0.5 0 0.25 0.25 0
15%	1 4 3 6 2 5 8 11 | 7 | 9 10 12 13 | 14 15 | 16 17 19 21 20 18 | 22 24 | 23	0.1250 0 0.2500 0 0 0 0
20%	1 | 2 5 6 3 7 8 11 10 | 4 | 9 | 12 13 14 15 22 24 23 | 16 17 19 21 20 18	0 0.1250 0 0 0.1429 0
30%	1 3 7 8 5 11 9 10 | 2 | 4 | 6 | 12 14 15 23 16 18 17 19 | 1 3 | 2 0 | 21 | 2 2 | 2 4	0.1250 0 0 0 0.25 0 0 1 0 0
40%	1 3 7 6 8 5 11 | 2 | 4 | 9 | 10 12 13 | 14 15 | 16 17 19 18 20 | 21 | 22 24 23	All zeros

Table 2.b: HRP1^{+}control: No link disruption – Complete node disruption

% of disruption	Complete disruption in nodes: HRP2^{+}	
	Clusters	q
5%	1 3 7 8 9 11 5 4 | 2 | 6 | 10 12 14 15 16 18 19 20 | 13 | 17 | 21 | 22 24 23	0.3750 0 2 0.8750 2 2 1 0.3333
10%	1 2 6 8 9 10 4 3 | 5 | 7 | 1 1 | 12 | 13 14 15 16 18 19 20 23 | 17 | 21 | 22 24	0.5 2 0 2 0 0.5 2 1 0.5
15%	1 2 5 8 4 6 7 | 3 | 9 10 12 14 15 16 23 17 | 11 | 13 | 18 20 21 | 19 | 22 | 24	0.1429 0 0.6250 0 2 0.3333 0 0 1
20%	1 3 7 8 9 11 5 4 | 2 | 6 | 1 0 | 12 14 15 22 | 13 | 16 | 17 | 18 19 | 20 21 | 23 24	0.2500 0 2 0 0.25 0 0 0 0.50 0.50 0.50
30%	1 2 3 7 8 11 6 4 | 5 | 9 | 10 12 14 15 16 17 19 2 | 13 | 18 | 20 | 22 24 23	0.3750 2 0 0.75 2 1 1 0.3333
40%	1 | 2 3 6 8 11 10 12 14 | 4 | 5 | 7 | 9 | 13 | 15 | 16 18 19 20 21 | 17 | 22 | 23 | 2 4	0 0.50 0 0 2 0 1 0 0.2 0 0 0 0

Table 2.c: HRP2^{+} : No link disruption – Complete node disruption

% of disruption	Complete disruption in nodes: HRP2⁺ control												
	Clusters												q
5%	1 2 6 8 4 11 5	3 7	9 10 12 14 15 16 18 19	13	1 7	20 21	22	23 24					0.4286 1 0.8750 2 2 0.50 0 0.50
10%	1 2 6 8 4 11 3 5	7	9 10 12 14 15 16 18 19	13	17	20 21	22 24 23						0.3750 2 0.8750 2 2 0.50 0.3333
15%	1 2 6 8 9 10 11 5	3 7 4	12	13 15 22 23 14 16 17 19	18	20	21	2 4					0.25 0.6667 1 0.6250 2 0 1 1
20%	1 3 4	2 6 8 7 11 9 10	5	12 13 15 22 23 24	14	16 17 19 21 18	20						0.3333 0.1429 0 0.3333 2 0.20 1
30%	1 3 7 2 6 8 9 10	4	5	1 1	12 13 15 22 23 24	1 4	16 17 19 21	18	20				0.3750 2 0 1 0.1667 1 0.25 1 0
40%	1 3 2	4 7 8 11	5	6	9	10 12 14 15 16 17 19 22	13	18	20	21	23	24	0.3333 0.25 0 1 0 0.8750 2 2 0 1 0 1

Table 2.d: HRP2⁺ control: No link disruption – Complete node disruption

- Applying the four methods on a network dealing with complete disruption in both nodes and links.

% of disruption	*Complete Disruption in Nodes and Links:* HRP1⁺																
	Clusters																q
5%	1,2,5,6,3,7,4	8	9,10,11,12,13,14,16,17	15,22,24,23	18,20,21,19												0,0,.25,.25,.25
10 %	1,2,3,6,8,11,10,7	4	5	9	12,14,16,17,18,20,21,19	13,15,22,24,23											.25,1,0,0,.25,.2
15 %	1	2,6,3,7,4,8,9,11	10,12,14,16,18,20,21,19	13,15,23	5	17	2 2	24									0,.125,.125,0,0,0,0,0
20 %	1,3,6,2,8,7,9,10	4	5	11	12,13,15,23,24,22	14	16,18,19,21,20	17									.125,0,0,0,.1667,0,0,0
30 %	1	2	3	4,7,8,6,11,9,10	5	12,14,15,23,24,13,16,18	17,19,21,20	22									0,0,0,.1429,0,.25,.25,0
40 %	1,4,7	2	3	5,8,11	6	9	1 0	1 2	13	14	15	16	17	18,19,21,20	22,24	2 3	0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0

Table 3.a: HRP1⁺ : Complete node disruption – Complete link disruption

% of disruption	*Complete Disruption in Nodes and Links:* **HRP1$^+$control**												
	Clusters												**q**
5%	1 2 5 4 7 3 6 8	9 10 11	12 13 15 22 24 14 16 17	18 20 21 19	23								0.1250 0.6667 0.25 0.25 0
10 %	1 4 3 2 6 8 11 9	5	7	10	12 14 16 17 18 20 21 19	13 15 22 24 23							0.1250 0 0 2 0.25 0.2
15 %	1 4 7 3 6 2 5 8	9	10 12 13 15 14 23 24 22	11	16 17 18 20 21	19							0.1250 0 0 1 0 0
20 %	1 2 3 7 6 5	4	8 9 10 11 12 14 15 22	13	16 17 19 21 18	20	23 24						0 0 0.2500 0 0.2 0 0.5
30 %	1	2 5 6 8 7 4 9 10	3	11	12 14 13 15 23 22	16 17	18	1 9	20	21	24		0 0.1250 0 0 0.1667 0 0 0 0 0 0
40 %	1	2	3	4 7 8 9	5	6	10 12 13 15 14 16 18 17	11	19 20 21	2 2	23	24	0 0 0 0 0 0 0.1250 0 0.3333 0 0 0

Table 3.b: HRP1$^+$control: complete link disruption – Complete node disruption

% of disruption	*Complete Disruption in Nodes and Links:* **HRP2$^+$**																	
	Clusters																	**q**
5 %	1 2 4 7 5	3 6 8 9 10 12 14 15	11	13	16 17 19 21 20	1 8	22 24 23											0.4 1 2 2 0.2 0 0.3333
10 %	1 3 6 8 4 11 5 7	2	9 10 12 13 15 22 23 24	14	16 18 19 20 21	17												0.1250 0 0.1250 0 0.2 1
15 %	1	2 6 5	3	4	7	8	9	10 12 14 15 16 18 20 21	11	13	17	19	22 24	23				0 0.3333 1 0 0 0 1 1 1 2 1 1 0.50 1
20 %	1 3 6 8 9 10 12 14	2	4	5	7	11	13	15	16 17 19 21 18 20	22	23	24						0.75 1 0 1 2 1 0 0 0.1667 0 0 0
30 %	1 2 5 8 6	3	4 7	9	10 12 13	11	14	1 5	16	17 19 21	18	20	22	2 3	2 4			0.40 0 1 0 0.3333 0 1 0 0 0.3333 0 1 0 0 0
40 %	1	2	3	4	5 8 9 10 12 14 11 16	6	7	13	15	17	18	19	20	21	22	23	24	0 0 0 0 0.1250 1 0 0 0 0 0 0 1 1 1 0 1

Table 3.c: HRP2$^+$: complete link disruption – Complete node disruption

% of disruption	*Complete Disruption in Nodes and Links:* **HRP2$^+$ control**	
	Clusters	**q**
5%	1 2 6 8 3 9 11 5 | 4 7 | 10 12 14 15 16 18 19 20 | 13 | 17 | 21 | 22 24 23	0.3750 1.0000 0.8750 2.0000 2.0000 1.0000 0.3333
10 %	1 2 6 8 4 9 10 5 | 3 7 | 11 | 12 13 15 22 24 | 14 | 16 18 19 20 21 | 17 | 23	0.6250 1.0000 2.0000 0.2000 0 0.4000 2.0000 0
15 %	1 2 3 7 8 9 11 4 | 5 | 6 | 10 12 14 13 15 23 | 16 17 19 21 | 18 | 20 | 22 | 24	0.6250 2.0000 2.0000 0.1667 0.2500 1 0 0
20 %	1 | 2 | 3 | 4 7 | 5 | 6 8 9 10 12 14 15 16 | 11 | 13 | 17 18 19 21 | 20 | 22 24 23	1 1 1 0.50 1 1.1250 2 2 0.25 0 0.3333
30 %	1 2 6 8 7 9 3 4 | 5 | 10 | 11 | 12 13 15 22 23 | 1 4 | 16 17 19 20 21 | 18 | 24	0.2500 2.0000 1.0000 1.0000 0.2000 1.0000 0.2000 1.0000 0
40 %	1 2 6 8 4 11 5 10 | 3 | 7 | 9 | 12 13 15 | 14 | 16 17 19 21 | 18 | 20 | 22 | 23 | 24	0.50 2 2 0 0.3333 0 0.25 1 0 1 0 0

Table 3.d: HRP2$^+$ control: complete link disruption – Complete node disruption

- Applying the four methods on a network dealing with partial disruption in links

% of disruption	*Partially Disrupted Links:* **HRP1$^+$**				
	Clusters				**q**
5%	1,4,7,3,6,2,5,8	9,10,11	12,14,16,17,18,20,21,19	13,15,22,24,23	.125,.6667,.25,.2
10 %	1,4,7,3,6,2,5,8	9 | 10,11	12,14,16,17,18,20,21,19	13,15,22,24,23	.25,2,1.5,.25,.2
15 %	1,4,7,3,2,5,8,11	6 | 9,10,12,13	14,16,17,18,20,21,19,15	22,24,23	.25,1,.5,.25,.3333
20 %	1,4,7,3,6,2,5	8,11,9	10,12,13,15,23,24,22,14	16,17,19,20,21,18	.1429,.6667,.25,.1667
30 %	1,4,7,3,6,2	5 | 8,9,10,11	12,14,16,17,18,20,21,19	13,15 | 22,24,23	.3333,2,.75,.25,1,.3333
40 %	1,4,7,3,2,5	6,8,11,9,10 | 12,13,14,16,17,19	15 | 18 | 20,21	22 | 23,24	.1667,.4,.6667,3,2,1,2,1

Table 4.a: HRP1$^+$: No node disruption – Partial link disruption

% of disruption	*Partially Disrupted Links:* **$\mathbf{HRP1^{+}}$control**								
	Clusters								**q**
5%	1 4 7 3 6 2 5 8	9 10 11	12 14 16 17 18 20 21 19	13 15 22 24 23					0.1250 0.6667 0.2500 0.2000
10%	1 4 3 6 2 5 8 11	7	9 10	12 14 16 17 18 20 21 19	13 15 22 24 23				0.2500 1.0000 1.0000 0.2500 0.2000
15%	1 4 7 3 6 2 5 8	9 10 11	12 14 16 17 18 20 21 19	13 15 22 24 23					0.1250 0.6667 0.2500 0.2000
20%	1 2 5 6 8 7 11 9	3	4	10	12 14 16 18 20 21 17 13	15	19	22 24 23	0.3750 1 1 2 0.3750 2 1 0.3333
30%	1 2 5 6 3 7	4	8 11 9 10	12 14 16 17 13 15 23 24	18 20 21 19	22			0.3333 1 0.50 0.3750 0.2500 1
40%	1 2 5 3 7	4	6 8 11 9	10	12 14 16 17 19 20 21 18	13 15	22 24 23		0.40 1 0.50 2 0.25 1 0.3333

Table 4.b: $HRP1^{+}$control : No node disruption – Partial link disruption

% of disruption	*Partially Disrupted Links:* **$\mathbf{HRP2^{+}}$**							
	Clusters							**q**
5%	1 2 6 8 4 11 5	3 7	9 10 12 14 15 16 18 19	13	17	20 21	22 24 23	0.4286 1 0.8750 2 2 0.5 0.3333
10%	1 2 6 8 4 11 5	3 7	9 10 12 14 15 16 18 19	13	17	20 21	22 24 23	0.4286 1 0.8750 2 2 0.5 0.3333
15%	3 7	9 10 12 14 15 16 18 19	13	17	20 21	22 24 23		0.4286 1 0.8750 2 2 0.50 0.3333
20%	1 2 3 6 8 11 5	4 7	9 10 12 14 15 16 18 19	13	17	20 21	22 24 23	0.4286 1 0.8750 2 2 0.5 0.3333
30%	1 3 4 6 8 2 7 11	5	9 10 12 14 15 16 18 19	13	17	21 20	23 22 24	0.6667 0.4286 0.8750 2 2 1 0.3333
40%	2 6 8 7 11 4 1 3	5	9 10 12 14 15 16 19 18	13	17	21 20	23 22 24	0.3750 2 0.8750 2 2 0.5 0.3333

Table 4.c: $HRP2^{+}$: No node disruption – Partial link disruption

% of disruption	*Partially Disrupted Links:* HRP2⁺ control							
	Clusters							**q**
5%	1 2 6 8 4 11 5	3 7	9 10 12 14 15 16 18 19	13	17	20 21	22 24 23	0.4286 1 0.8750 2 2 0.5 0.3333
10%	1 2 6 8 4 11 5	3 7	9 10 12 14 15 16 18 19	13	17	20 21	22 24 23	0.4286 1 0.8750 2 2 0.5 0.3333
15%	1 2 6 8 4 11 5	3 7	9 10 12 14 15 16 18 19	13	17	20 21	22 24 23	0.4286 1 0.8750 2 2 0.5 0.3333
20%	1 2 6 8 4 11 5	3 7	9 10 12 14 15 16 18 19	13	17	20 21	22 24 23	0.4286 1 0.8750 2 2 0.5 0.3333
30%	1 2 6 8 4 11 5	3 7	9 10 12 14 15 16 18 19	13	17	20 21	22 24 23	0.4286 1 0.8750 2 2 0.5 0.3333
40%	1 2 6 8 4 11 5	3 7	9 10 12 14 15 16 18 19	13	17	20 21	22 24 23	0.4286 1 0.8750 2 2 0.5 0.3333

Table 4.d: HRP2⁺ control: No node disruption – Partial link disruption

- Applying the four methods on a network encountering partial disruption in nodes

% of disruption	*Partially Disrupted Nodes:* HRP1⁺												
	Clusters												**q**
5%	1,4,7,2,6,5,8,11	3	9,10	12,14,16,17,18,20,21,19	13,15,22,24,23								0.3750,1,1,.25,.2
10%	1,4,7,3,6,2,5,8	9	10,12,13,15,22,24,23,14	11	16	17,18,20,21,19							.25,2,.375,2,2,.2
15%	1,4,3,6,2,5,8,11	7	9,1012,13,14,16,17,19	15	18	20	21	22,24	23				.25,1,.5,3,2,2,2,1,2
20%	1	2	3,5,8,9,10,11,6,7	4	12,14,16,17,19,20,21,13	15,23,24	18	22					3,2,0.5,2,0.375,.6667,1,1
30%	1,3,7,6,2,8	4	5	9,10,11,12,13,14,16,15	17	18	19	20,21	22	23	24		.5,1,1,.625,2,3,3,1,2,2,2
40%	1	2,4,7,8,6,9	3	5	10	11	12	13,14,16,18,20,19,15,23	17	21	22	24	2,.8333,2,1,3,2,2,.625,1,1,2,2

Table 5.a: HRP1⁺ : No link disruption – Partial node disruption

% of disruption	*Partially Disrupted Nodes:* **HRP1$^+$control**										
	Clusters										**q**
5%	1 4 7 3 6 2 5 8	9 10 11	12 14 16 17 18 20 21 19	13 15	22 24 23						0.1250 0.6667 0.2500 1.0000 0.3333
10 %	1 4 3 6 2 8 11 9	5	7	1 0	12 14 16 17 18 20 21 19	13 15 22 24 23					0.3750 1 1 2 0.2500 0.2000
15 %	1 4 7 3 6 2 5 8	9	10 11	12 14 16 18 20 21 17 13	15 22 24 23	19					0.2500 2 1.50 0.3750 0.2500 1
20 %	1 2 5 6 3 7	4	8 11 9 10	12 14 15 22 24 23 13 16	17 19 18 21	20					0.3333 1 0.50 0.25 0.50 1
30 %	1	2 5 6 3 8 11 9	4 7	10 12 13 14 16 17 18 15	1 9	2 0	2 1	22	23	24	2 0.4286 1 0.6250 2 2 2 2 2 2
40 %	1 4 7 8 6 5 11 9	2	3	1 0	12 14 16 17 18 20 21 19	13 15	22	23	24		0.3750 1 1 2 0.25 1.50 2 2 2

Table 5.b: HRP1$^+$control : No link disruption – Partial node disruption

% of disruption	*Partially Disrupted Nodes:* **HRP2$^+$**							
	Clusters							**q**
5 %	1 2 6 8 4 11 5	3 7	9 10 12 14 15 16 18 19	1 3	1 7	21 20	22 24 23	0.4286 1 0.8750 2 2 0.5 0.3333
10 %	1 3 6 8 4 11 7	2 5	9 10 12 14 15 16 18 19	13	1 7	21 20	22 24 23	0.4286 1 0.8750 2 2 0.50 0.3333
15 %	3 7 4	5 2 8 6 1 9	11 12 14 15 10 16 18 19	13	1 7	21 20	23 22 24	0.6667 0.50 0.8750 2 2 0.50 0.3333
20 %	1 2 6 8 4 11 5	3 7	9 10 12 14 15 16 18 19	13	17	21 20	23 22 24	0.4286 1 0.8750 2 2 0.50 0.3333
30 %	3 7 4	5 1 2 6 8 9 11	10 12 14 15 16 18 19 20	1 3	17	21	22 24 23	0.6667 0.4286 0.875 2 2 1 0.3333
40 %	4 7 3	5 1 6 8 2 9 11	10 12 14 15 16 18 19 20	13	17	21	22 24 23	0.6667 0.4286 0.8750 2 2 1 0.3333

Table 5.c: HRP2$^+$: No link disruption – Partial node disruption

% of disruption	*Partially Disrupted Nodes:* $HRP2^{+}$ control							
	Clusters							q
5 %	1 2 6 8 4 11 5	3 7	9 10 12 14 15 16 18 19	13	1 7	20 21	22 24 23	0.4286 1 0.8750 2 2 0.5000 0.3333
10 %	1 2 6 8 4 11 5	3 7	9 10 12 14 15 16 18 19	13	17	20 21	22 24 23	0.4286 1 0.8750 2 2 0.5 0.3333
15 %	1 2 6 8 4 11 5	3 7	9 10 12 14 15 16 18 19	13	17	20 21	22 24 23	0.4286 1 0.8750 2 2 0.5 0.3333
20 %	1 2 6 8 4 11 5	3 7	9 10 12 14 15 16 18 19	13	17	20 21	22 24 23	0.4286 1 0.8750 2 2 0.50 0.3333
30 %	1 2 6 8 4 11 5	3 7	9 10 12 14 15 16 18 19	13	17	20 21	22 24 23	0.4286 1 0.8750 2 2 0.50 0.3333
40 %	1 2 6 8 4 11 5	3 7	9 10 12 14 15 16 18 19	13	17	20 21	22 24 23	0.4286 1 0.8750 2 2 0.5 0.3333

Table 5.d: HRP*2^{+} *control: No link disruption – Partial node disruption

- Applying the four methods on a network encountering partial disruption in both nodes and links

% of disruption	*Partially Disrupted Links and Nodes:* $HRP1^{+}$																		
	Clusters																		q
5%	1,4,7,3,2,5,8,11	6	9	10	12,14,16,17,18,20,21,19	13,15,22,24,23													.375,1,2,3,.25,.2
10 %	1	2,5,6,3,8,11,9	4,7	10,12,13,15,22,24,23,14	16	17,19,18,21,20													2,.4286,1,.25,2,.2
15 %	1	3	5	9	21	2,4,7,8,6	10,11,12,13,15,14,16,17	18,20,19	22,24,23										2,2,1,2,1,1,0.5,0.6667,0.3333
20 %	1	2,3,4,7,6	5	8	9,10,11	12,13,14	15	16	17	18,20,21	19	22,24	23						1,.6,2,3,.6667,1,3,3,2,.6667,2,1,2
30 %	1,4	2,5	3	6	7,8,9	10	11	12,13,15,22,24,23,14,16	17	18	19,20,21								1.5,1.5,3,3,2,3,2,.375,2,2,.6667
40 %	1,3,6	2	4	5	7	8,11,9	10	12	13	14	15,23,24	16	17	18	19	20	21	22	1.3333,2,2,2,3,1.3333,2,3,2,3,1,3,2,3,3,2,2,1

Table 6.a: HRP*1^{+} *: partial link disruption – Partial node disruption

% of disruption	Partially Disrupted Links and Nodes: HRP1^{+}control										
	Clusters										q
5%	1 4 7 3 6 2 5 8	9 10 11	12 14 16 17 18 20 21 19	13 15	22 24 23						0.1250 0.6667 0.2500 1 0.3333
10 %	1 4 3 6 2 8 11 9	5	7	1 0	12 14 16 17 18 20 21 19	13 15 22 24 23					0.3750 1 1 2 0.25 0.20
15 %	1 4 7 3 6 2 5 8	9	10 11	12 14 16 18 20 21 17 13	15 22 24 23	19					0.2500 2 1.50 0.3750 0.2500 1
20 %	1 2 5 6	3 7 4	8 9 10 11 12 13 14 16	15	17 19 18 21	20	22 24 23				0.50 0.6667 0.50 2 0.50 1 0.3333
30 %	1	2 5 6 3	4 7 8 11 9	10 12 13 14 16 17 15 23	1 8	1 9	20	21	22	2 4	2 0.50 0.60 0.6250 3 3 2 2 2 2
40 %	1	2	3	4 7 8 10 11 9 6 5	12 14 16 17 19 20 21 18	13 15	2 2	2 3	24		3 2 2 0.50 0.25 1.50 2 2 2

Table 6.b: HRP1^{+}control: partial link disruption – Partial node disruption

% of disruption	Partially Disrupted Links and Nodes: HRP2^{+}							
	Clusters							q
5 %	1 2 6 8 4 11 5	3 7	9 10 12 14 15 16 18 19	13	17	20 21	23 22 24	0.4286 1 0.8750 2 2 0.5 0.3333
10 %	1 3 6 8 4 11 7	2 5	9 10 12 14 15 16 18 19	13	17	21 20	23 22 24	0.4286 1 0.8750 2 2 0.5 0.3333
15 %	3 7 4	5 2 8 6 1 9	11 12 14 15 10 16 18 19	13	17	21 20	23 22 24	0.6667 0.50 0.8750 2 2 0.50 0.3333
20 %	1 2 3 6 8 11 5	4 7	9 10 12 14 15 16 19 18	13	17	21 20	23 22 24	0.4286 1 0.8750 2 2 0.5 0.3333
30 %	1 3 4 6 8 2 7 11	5	9 10 12 14 15 16 18 19	13	17	21 20	23 22 24	0.3750 2 0.8750 2 2 0.5 0.3333
40 %	4 7 3	5 1 6 8 2 9 11	10 12 14 15 16 18 19 20	13	17	21 20	23 22 24	0.6667 0.4286 0.8750 2 2 1 0.3333

Table 6.c: HRP2^{+} : partial link disruption – Partial node disruption

% of disruption	*Partially Disrupted Links and Nodes:* **HRP2⁺ control**	
	Clusters	**q**
5%	1 2 6 8 4 11 5 | 3 7 | 9 10 12 14 15 16 18 19 | 13 | 17 | 20 21 | 22 24 23	0.4286 1 0.8750 2 2 0.5 0.3333
10 %	1 2 6 8 4 11 5 | 3 7 | 9 10 12 14 15 16 18 19 | 13 | 17 | 20 21 | 22 24 23	0.4286 1 0.8750 2 2 0.5 0.3333
15 %	1 2 6 8 4 11 5 | 3 7 | 9 10 12 14 15 16 18 19 | 13 | 17 | 20 21 | 22 24 23	0.4286 10.8750 2 2 0.5 0.3333
20 %	1 2 6 8 4 11 5 | 3 7 | 9 10 12 14 15 16 18 19 | 13 | 17 | 20 21 | 22 24 23	0.4286 1 0.8750 22 0.5 0.3333
30 %	1 2 6 8 4 11 5 | 3 7 | 9 10 12 14 15 16 18 19 | 13 | 17 | 20 21 | 22 24 23	0.4286 1 0.8750 2 2 0.5 0.3333
40 %	1 2 6 8 4 11 5 | 3 7 | 9 10 12 14 15 16 18 19 | 13 | 17 | 20 21 | 22 24 23	0.4286 1 0.8750 2 2 0 0.3333

Table 6.d: HRP2⁺ control: partial link disruption – Partial node disruption

- Applying the four methods on a network dealing with partial disruption in links and complete failure in nodes

% of disruption	*Complete Disruption in Nodes and Partially disrupted Links:* **HRP1⁺**	
	Clusters	**q**
5%	1,4,7,3,6,2,5,8 | 9 | 10 | 11 | 12,13,14,16,17,19,21,20 | 15,22,24,23 | 18	0.25,1,0,1,.25,.25,1
10 %	1,4,7,3,2,8 | 5 | 6 | 9,10,12,13,15,23,24,22 | 11 | 14 | 16,17,19,21,20,18	.5,1,1,.125,0,0,0
15 %	1 | 2,3,6,3,7,4 | 8 | 9,10,11,12,14,15,22,24 | 13 | 16 | 17 | 18 | 19,21,20 | 23	1,.1667,0,.125,0,2,0,2,.333,0
20 %	1,4,2,6 | 3 | 5 | 7 | 8,9,10,11,12,14,16,17 | 13,15,23,22 | 18 | 19 | 20,21 | 24	.5,0,2,0,.375,.25,0,0,0,0
30 %	1,3,7,6,8,11,9 | 2 | 4 | 5 | 10 | 12,14 | 13 | 15 | 16 | 17,19,18 | 20 | 21 | 22 | 23 | 24	.4286,2,0,2,2,.5,0,0,0,.6667,2,2,0,0,0
40 %	1 | 2 | 3 | 4 | 5 | 6 | 7 | 8,11,9 | 12,13 | 14 | 15,23 | 16 17 | 18 | 19 | 20 | 21 | 22 | 24	2,0,2,1,0,2,0,.6667,2,1,0,1,.5,0,1,0,0,1,0

Table 7.a: HRP1⁺ : partial link disruption – complete node disruption

% of disruption	*Complete Disruption in Nodes and Partially disrupted Links:* **HRP1^{+}control**	
	Clusters	**q**
5%	1 4 7 3 6 2 5 8 | 9 10 11 | 12 13 15 22 24 14 16 17 | 18 20 21 19 | 23	0.1250 0.6667 0.2500 0.2500 0
10 %	1 4 3 2 6 8 11 9 | 5 | 7 | 1 0 | 12 14 16 17 18 20 21 19 | 13 15 22 24 23	0.1250 0 0 2.0000 0.2500 0.2000
15 %	1 4 7 3 6 2 5 8 | 9 | 10 12 13 15 14 23 24 22 | 1 1 | 16 17 18 20 21 | 19	0.2500 1.0000 0 1.0000 0 0
20 %	1 2 5 6 | 3 7 | 4 | 8 9 10 11 12 13 14 16 | 1 5 | 17 19 21 18 | 2 0 | 22 24 23	0.2500 0.5000 0 0.2500 2.0000 0.2500 0 0.3333
30 %	1 | 2 4 7 8 6 5 9 10 | 3 | 1 1 | 12 14 16 17 13 15 23 22 | 1 8 | 1 9 | 2 0 | 21 | 2 4	0 0.1250 0 0 0.2500 1.0000 0 0 0 0
40 %	1 | 2 | 3 | 4 8 7 9 | 5 | 6 | 10 12 13 15 14 16 17 19 | 1 1 | 1 8 | 2 0 2 1 | 2 2 | 2 3 | 2 4	1 0 0 0.50 0 0 0.3750 0 2 1 0 0 0

Table 7.b: HRP1^{+}control: partial link disruption – complete node disruption

% of disruption	*Complete Disruption in Nodes and Partially disrupted Links:* **HRP2^{+}**	
	Clusters	**q**
5%	1 2 6 8 4 11 5 | 3 7 | 9 10 12 14 15 16 23 13 | 17 19 21 20 | 18 | 22 24	0.4286 1 0.3750 0.2500 0 0.5
10 %	1 7 8 3 11 9 10 5 | 2 | 4 | 6 | 12 13 15 23 22 24 | 14 | 16 19 18 20 21 | 17	0.6250 0 2 2 0.1667 0 0.4 2
15 %	1 | 2 6 5 | 3 | 4 | 7 | 8 | 9 | 10 12 14 15 16 18 19 20 | 11 | 13 | 17 | 21 | 22 24 23	0 0.3333 1 0 0 0 1 1 1 2 2 1 0.3333
20 %	1 9 10 2 3 6 8 11 | 4 | 5 | 7 | 12 14 18 19 16 20 21 17 | 13 | 15 | 22 | 23 | 24	0.6250 0 2 2 0.1250 0 0 0 0 0
30 %	1 2 5 8 4 9 10 12 | 3 | 6 | 7 | 11 | 13 | 14 | 15 | 16 | 17 18 19 20 | 21 | 22 | 23 | 24	0.75 0 2 2 0 1 1 0 0 0.2500 1 0 0 0
40 %	1 | 2 | 3 | 4 | 5 | 6 | 8 9 10 12 14 15 22 7 | 11 | 13 | 16 | 17 | 18 | 19 | 20 | 21 | 23 | 24	0 0 0 1 1 1 1.1250 2 2 1 0 0 0 1 1 0 1

Table 7.c: HRP2^{+} : partial link disruption – complete node disruption

% of disruption	*Complete Disruption in Nodes and Partially disrupted Links:* **HRP2$^+$ control**	
	Clusters	**q**
5 %	1 2 6 8 3 9 11 5 | 4 7 | 10 12 14 15 16 18 19 20 | 13 | 17 | 21 | 22 24 23	0.3750 1.0000 0.8750 2.0000 2.0000 1.0000 0.3333
10 %	1 2 6 8 4 9 10 5 | 3 7 | 1 1 | 12 13 15 22 24 | 16 18 19 20 21 | 17 | 2 3	0.6250 1.0000 2.0000 0.2000 0 0.4000 2.0000 0
15 %	1 2 6 8 4 11 5 | 3 7 | 9 10 12 13 15 22 23 14 | 16 17 19 21 | 18 | 2 0 | 24	0.4286 1.0000 0.1250 0.2500 1.0000 0 0
20 %	1 | 2 | 3 | 4 7 | 5 | 6 8 9 10 12 14 15 16 | 11 | 13 | 17 19 18 21 | 2 0 | 22 24 23	1.0000 1.0000 1.0000 0.5000 1.0000 1.1250 2.0000 2.0000 0.2500 0 0.3333
30 %	1 2 6 8 9 10 7 3 | 4 | 5 | 1 1 | 12 13 15 22 23 | 1 4 | 16 17 19 21 | 18 20 | 2 4	0.6250 1.0000 2.0000 2.0000 0.2000 1.0000 0.2500 0.5000 0
40 %	1 2 6 8 4 11 9 10 | 3 | 5 | 7 | 12 13 15 22 | 1 4 | 16 17 19 21 | 1 8 | 2 0 | 2 3 | 2 4	0.7500 2.0000 2.0000 2.0000 0.2500 0 0.5000 2.0000 0 0 1.0000

Table 7.d: HRP2$^+$ control: partial link disruption – complete node disruption

- Applying the four methods while network is facing partial disruption in nodes and complete failure in links

% of disruption	*Complete Disruption in Links and Partially Disrupted Node:* **HRP1$^+$**	
	Clusters	**q**
5%	1,4,7,3,6,2, 5,8 | 9,10,12,13,14,16,18 ,20 | 11 | 15,22,24, 23 | 17 | 19,21	.1429,.5,1,.25,2,1
10 %	1,4,7,3,6,2 ,5 | 8 | 9,10,11 | 12,13,15,23,24, 22,14 | 16 | 17,19,20,21, 18	.1423,2,.6667,.2857,2,.2
15 %	1,2,5, 6,4 | 3 | 7 | 11 | 2 0 | 8,9,10,12,14,16 ,17,13 | 15,22,24, 23 | 18,19,21	.6,2,3,1,1,.5,.25,.3333
20 %	1 | 2,5,8,11,6,7 ,4 | 3 | 9 | 1 0 | 12,13,15,23,24, 22 | 14 | 16 | 17,18,20,21, 19	1,.5714,1,2,3,.1667,1,2,.2
30 %	1 | 2,5,6,3,8, 9,11 | 4 | 7 | 10,12,14,15,23, 24,22 | 1 3 | 1 6 | 17 | 18,19,21 | 20	2,.4286,2,2,.2857,0,3,2,1,1
40 %	1,3,6,2, 8,11,7 | 4 | 5 | 9 | 10 | 12 | 1 3 | 14,15, 22,23 | 16 | 17 | 18 | 19 | 20 | 21 | 24	.1429,0,1,1,2,3,1,.75,3, 2,2,2,1,1,1

Table 8.a: HRP1$^+$: partial node disruption – complete link disruption

% of disruption	*Complete Disruption in Links and Partially Disrupted Nodes:* **HRP1⁺control**												
	Clusters												**q**
5%	1 4 7 3 6 2 5 8	9 10 11	12 14 16 17 13 19 21 20	15	18	22 23 24							0.1250 0.6667 0.3750 2 1 0.3333
10 %	1 3 4 2	5	6 8 9 10 11 7	12 14 16 17 18 20 21 19	13 15 22 24 23								0.5 2 0.5 0.25 0.2
15 %	1 4 7 3 6 2 5 8	9	10 12 13 14 16 17 18 20	11	15 22 24 23	19	21						0.25 2 0.5 1 0.25 2 2
20 %	1 2 6 3 7	4	5	8 9	10 11 12 13 14 16 17 19	15 22 24 23	18	20	21				0.4 1 1 1.50 0.50 0.25 2 2 2
30 %	1	2 5 6 3 8 11 9	4 7	10 12 13 14 15 23 22	16 17 18	19	20	21	24				1 0.4286 0.50 0.2857 0.6667 2 2 2 1
40 %	1	2	3	4 7	5	6	8	9 10 11 12 14 16 13 15	17 19 18 21 20	22	23	24	1 2 1 0.50 2 1 3 0.3750 0 2 2 2

Table 8.b: HRP1⁺control: partial node disruption – complete link disruption

% of disruption	*Complete Disruption in Links and Partially Disrupted Nodes:* **HRP2⁺**										
	Clusters										**q**
5 %	1 2 6 8 4 9 10 5	3 7	11	12 13 15 22 23 24	14 16 18 19 20 21 17						0.6250 1.0000 2.0000 0.3333 0.1429
10 %	3 1 6 8 2 7 9 11	4	5	10 12 14 15 16 18 19 20	13	17	21	22 24 23			0.6250 2 2 0.8750 2 2 1 0.3333
15 %	2	3 1 8 6 11 10 12 13	4	5	7	9	14 15 16 18 19 20 23 21	1 7	22 24		1 0.8750 1 1 1 2 0.5 2 0.5
20 %	1 3	2 6 8 7 4 9 5	10 12 14 15 22 23 16 24	11	13	17 19 18 20	21				0.5000 0.1429 0.3750 1 2 0.2500 1
30 %	1 2 6 8 3 7 9 10	4	5	11	12 15 14 23 16 22 17 24	13	18 19	20 21			0.5 1 2 1 0.25 2 0.5 0.5
40 %	1	2 3 5 6 8 7 9 10	4	11	12	13	14 15 23 17	16 21 19	18 20	22 24	0 0.5 1 2 1 0 0.2857 0.5 0.5

Table 8.c: HRP2⁺ : partial node disruption – complete link disruption

% of disruption	*Complete Disruption in Links and Partially Disrupted Nodes:* **HRP2$^+$ control**									
	Clusters									**q**
5 %	1 3 6 8 4 11 5 2	7	9 10 12 14 15 16 18 19	13	17	20 21	22 24 23			0.3750 2 0.8750 2 2 0.5 0.3333
10 %	1 2 6 8 4 11 5	3 7	9 10 12 14 15 16 23 13	17 19 21 18 20	22 24					0.4286 1 0.3750 0.2 0.5
15 %	1 2 6 8 4 9 10 11	3 7	5	12 13 15 23 22 24	14 16 18 19 20 21 17					0.5 1 2 0.1667 0.1429
20 %	1	3	4	5	6 8 2 9 10 12 14 7	11	13 15 22 23 24	16 18 19 20 21 17		1 1 1 2 1.1250 2 0.2 0.1667
30 %	1 2 6 8 9 10 7 4	3	5	11	12 14 15 16 18 19 20 23	13	17	21	22 24	0.6250 2 1 2 0.75 2 2 1 0.5
40 %	1 2 6 8 4 9 10 12	3	5	7	1 1	13 15 22 23 24	14 16 17 19 21 20 18			1 2 1 2 1 0.2 0.1429

Table 8.d: HRP2$^+$ control: partial node disruption – complete link disruption

Appendix (II)

HRP1+

Matlab Code for HRP1+

```
for situ=1:9

    for N = start:step:fin
        connect= zeros(N,N);
        connect(:,1)=linspace(1,N,N);
        switch situ
            case 1
                % No disruption
                tic
                x=tx;
                wx=x;
                [q,result]= fhrplink(connect,x,W,Q,wx);
                fprintf('\n\n Case %6.0f \n',situ);
                result
                q
                toc
            case 2
                % Node no disruption - link complete disruption
                tic
                result = [];
                q = [];
                x = [];
                x = tx;
                [s_node,f_node]=find(x==1);
                rnd_dist1 = randi([1 length(s_node)],n_dist,1);
                x(s_node(rnd_dist1),f_node(rnd_dist1))=0;
                wx=x;
                [q,result]= fhrplink(connect,x,W,Q,wx);
                fprintf('\n\n Case %6.0f \n',situ);
                result
                q
                toc
            case 3
                % Node complete disruption - link no disruption
                tic
                result = [];
                q = [];
                x = [];
                wx = [];
                x = tx;
                [s_node,f_node]=find(x==1);
                rnd_dist2 = randi([1 fin],n_dist,1);
                x(rnd_dist2,:)=0;
                x(:,rnd_dist2)=0;
                wx=x;
                [q,result]= fhrplink(connect,x,W,Q,wx);
                fprintf('\n\n Case %6.0f \n',situ);
```

```
                result
                q
                toc
            case 4
                % Node complete disruption - link complete disruption
                tic
                result = [];
                q = [];
                x = [];
                wx = [];
                x = tx;
                x(s_node(rnd_dist1),f_node(rnd_dist1))=0;
                rnd_dist2 = unidrnd(fin,1,n_dist);
                x(rnd_dist2,:)=0;
                x(:,rnd_dist2)=0;
                wx=x;
                [q,result]= fhrplink(connect,x,W,Q,wx);
                fprintf('\n\n Case %6.0f \n',situ);
                result
                q
                toc
            case 5
                % Node no disruption - link partial disruption
                tic
                result = [];
                q = [];
                x = [];
                wx = [];
                x = tx;
                wx=x;

wx(s_node(rnd_dist1),f_node(rnd_dist1))=5.*wx(s_node(rnd_dist1),f_node(rnd_d
ist1));
                [q,result]= fhrplink(connect,x,W,Q,wx);
                fprintf('\n\n Case %6.0f \n',situ);
                result
                q
                toc
            case 6
                % Node Partial disruption - link no disruption
                tic
                result = [];
                q = [];
                x = [];
                wx = [];
                x = tx;
                wx=x;
                wx(rnd_dist2,:)=5.*wx(rnd_dist2,:);
                wx(:,rnd_dist2)=5.*wx(:,rnd_dist2);
                [q,result]= fhrplink(connect,x,W,Q,wx);
                fprintf('\n\n Case %6.0f \n',situ);
                result
                q
                toc
            case 7
                % Node partial disruption - link partial disruption ---->
                % HAS PROBLEM
```

```
                tic
                result = [];
                q = [];
                x = [];
                wx = [];
                x= tx;
                wx=x;

wx(s_node(rnd_dist1),f_node(rnd_dist1))=5.*wx(s_node(rnd_dist1),f_node(rnd_d
ist1));
                wx(rnd_dist2,:)=5.*wx(rnd_dist2,:);
                wx(:,rnd_dist2)=5.*wx(:,rnd_dist2);
                [q,result]= fhrplink(connect,x,W,Q,wx);
                fprintf('\n\n Case %6.0f \n',situ);
                result
                q
                toc
            case 8
                % Node complete disruption - link partial disruption
                tic
                result = [];
                q = [];
                x = [];
                wx = [];
                x = tx;
                wx=x;
                x(rnd_dist2,:)=0;
                x(:,rnd_dist2)=0;

wx(s_node(rnd_dist1),f_node(rnd_dist1))=5.*wx(s_node(rnd_dist1),f_node(rnd_d
ist1));
                [q,result]= fhrplink(connect,x,W,Q,wx);
                fprintf('\n\n Case %6.0f \n',situ);
                result
                q
                toc
            case 9
                % Node partial disruption - link complete disruption
                tic
                result = [];
                q = [];
                x = [];
                wx = [];
                x = tx;
                wx=x;
                x(s_node(rnd_dist2),f_node(rnd_dist2))=0;
                wx(rnd_dist2,:)=5.*wx(rnd_dist2,:);
                wx(:,rnd_dist2)=5.*wx(:,rnd_dist2);
                [q,result]= fhrplink(connect,x,W,Q,wx);
                fprintf('\n\n Case %6.0f \n',situ);
                result
                q
                toc
        end
    end
end
```

```
function [q,result]= fhrplink(connect,input,W,Q,wx)
while (max(card) <= W || max(q) >= Q)
    K = length(x); %
    for i=1:K
        x(i,i)=0;
        w(i,i)=0;
    end
    %%%q(s) calculation%%%
    q = zeros(1,K) ;
    alpha =zeros(1,K);
    c=[];
    q_temp=[];
    rn=[];
    for i=1:K
        rn=x(i,:)+x(:,i)';
        c(i)=length(find(rn~=0));
        q(i)=c(i)/card(i);
        q_temp(i)=q(i);
    end
    if (K == 2 || max(card)> W )
        break;
    end
    %%%%%%%q(s) calculation%%%%%%
    q_opt = 2000* ones(1,K);
    w_opt = 2000* ones(1,K);
    w_temp = 1* ones(1,K);
    q_opt_max =1000 ;
    neigh_opt_1 = 0 ;
    neigh_opt_2 = 0 ;
    neigh_opt_1_2 = 0;
    neigh_opt_2_2 = 0;
    %%%% FINDING THE q STAR%%%%
    for i = 1 : K-1
        for j = i+1:K
            if ((x(i,j)== 1 || x(j,i)== 1) && (card(i)+card(j))<= W &&  q(i)
> Q && q(j) > Q)
                [q_temp(i)]= merging(i,j,x,card);
                if (q_temp(i) <= q_opt(i) && w_temp(i) >= w(i,j))
                     q_opt(i) = q_temp(i) ;
                     w_temp(i) = w(i,j);
                     neigh_opt(i) = result(j,1);
                     neigh_opt2(i) = j;
                end
            end
        end
        if (q_opt(i)< q_opt_max)
           q_opt_max = q_opt(i)
           neigh_opt_1 = result(i,1);
           neigh_opt_1_2=i
           neigh_opt_2 =neigh_opt(i);
           neigh_opt_2_2=neigh_opt2(i)
        end

    end
```

```
    if (length(card)<=(length(input)/W)+1 || length(card) < neigh_opt_1_2 ||
length(card) < neigh_opt_2_2)
        break;
    end
    if (neigh_opt_1_2 ~= 0 || neigh_opt_2_2 ~= 0)
        if (card(neigh_opt_1_2)+card(neigh_opt_2_2))> W
            n=1;
        end
    end
    %%% FINDING THE q STAR %%
    neigh_opt_1;
    neigh_opt_2;
    if (neigh_opt_1_2== 0 || neigh_opt_2_2== 0)
        break;
    end

    %%%Merging%%
    x(neigh_opt_1_2,:)=x(neigh_opt_1_2,:)+x(neigh_opt_2_2,:);
    x(:,neigh_opt_1_2)=x(:,neigh_opt_1_2)+x(:,neigh_opt_2_2);
    x(neigh_opt_2_2,:)=[];
    x(:,neigh_opt_2_2)=[];
    x(x==2)=1;
    w(neigh_opt_1_2,:)=w(neigh_opt_1_2,:)+w(neigh_opt_2_2,:);
    w(:,neigh_opt_1_2)=w(:,neigh_opt_1_2)+w(:,neigh_opt_2_2);
    w(neigh_opt_2_2,:)=[];
    w(:,neigh_opt_2_2)=[];
    w(w==2)=1;
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Cardinality
Expression
    card_new=[];
    card_new=card;
    card_new(neigh_opt_1_2) = card_new(neigh_opt_1_2) +
card_new(neigh_opt_2_2);
    card_new(neigh_opt_2_2)=[];
    card = card_new ;
     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Cardinality Expression
Calculation
     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%Node Mapping %%%%%%%%%%%%%%%%%%%%%%%%
    neigh_opt_1=result(neigh_opt_1_2,1);
    neigh_opt_2=result(neigh_opt_2_2,1);
    temp1=find(result(result(:,1)==neigh_opt_2,:));
    temp2=result(result(:,1)==neigh_opt_2,temp1);
    temp3=find(result(result(:,1)==neigh_opt_1,:));

result(result==neigh_opt_1,length(temp3)+1:length(temp3)+length(temp2))=temp
2;
    result(result(:,1)==neigh_opt_2,:)=[]
    n=n+1



end
ec=length(x(x==1));
end
```

```
function [q,card,x]= merging(i,j,x,card)
neigh_opt_1=i;
```

```
neigh_opt_2=j;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%Merging%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
x(neigh_opt_1,:)=x(neigh_opt_1,:)+x(neigh_opt_2,:);
x(:,neigh_opt_1)=x(:,neigh_opt_1)+x(:,neigh_opt_2);
x(neigh_opt_2,:)=[];
x(:,neigh_opt_2)=[];
x(x==2)=1;
%%%%%%%%%%%%%%%%%%%%%%%%%%%% Cardinality Expression%%%%%%%%%%%%%%%%%%%%%%
card_new = card(neigh_opt_1) + card(neigh_opt_2);
card(neigh_opt_1)=card_new;
card(neigh_opt_2)=[];
%%%%%%%%%%%%%%%%%%%%%%%%%%q(s) calculation%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
rn=x(i,:)+x(:,i)';
c=length(find(rn~=0));
q=c/card_new;

end
```

Appendix (III)

HRP2^{+}

Matlab Code for HRP2^{+}

```
function [fq,result]= f2hrp2link(connect,input,W,Q,wx)

while (K > 0)

    %%%q(s) calculation%%%%%

    X=0;
    w_temp=1;
    cont=0;
    j1=-1;
    for i=1:K
        for j=1:S(i)
            r=ls(i,j);
            [w]= merging2(i,r,x,nb);
            if (w > X) && (nb(i)+nb(r) <= W) && (w > Q)  && wx(i,r)<=w_temp
                X=w
                i1=i
                j1=r
                cont=1;
            end
        end

        for j=1:P(i)
            r=lp(i,j);
            [w]= merging2(i,r,x,nb);
            if (w > X) && (nb(i)+nb(r) <= W) && (w > Q)  && wx(i,r)<=w_temp
                X=w
                i1=i
                j1=r
                cont=1;
            end
        end
    end

    if cont == 0
       break;
    end

    if j1 == -1
        break;
    end


  %%% q(s) calculation%%%%
```

```
    neigh_opt_1 = i1 ;
    neigh_opt_2 = j1 ;
    neigh_opt_1_2 = i1;
    neigh_opt_2_2 = j1;

    %%%%%% FINDING THE q STAR %%%%%%
    neigh_opt_1;
    neigh_opt_2;
    if (neigh_opt_1_2== 0 || neigh_opt_2_2== 0)
        break;
    end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%Merging%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    x(neigh_opt_1_2,:)=x(neigh_opt_1_2,:)+x(neigh_opt_2_2,:);
    x(:,neigh_opt_1_2)=x(:,neigh_opt_1_2)+x(:,neigh_opt_2_2);
    x(neigh_opt_2_2,:)=[];
    x(:,neigh_opt_2_2)=[];
    x(x==2)=1;
    K = length(x);
    for i=1:K
        x(i,i)=0;
    end
    %%%% Cardinality Expression
    S=[];
    ls=[];
    P=[];
    lp=[];
    q=[];
    nb(neigh_opt_1_2)=nb(neigh_opt_1_2)+nb(neigh_opt_2_2);
    nb(neigh_opt_2_2)=[];
    for i=1:K
        S(i)=length(find(x(i,:)==1));
        ls(i,1:length(find(x(i,:)==1)))=find(x(i,:)==1);
        P(i)=length(find(x(:,i)==1));
        lp(i,1:length(find(x(:,i)==1)))=find(x(:,i)==1);
        q(i)=S(i)+P(i);
        e(i)=1;
    end
     %%%%%%%%%%%%%%%%%%%%%%%%%%%% Cardinality Expression Calculation
     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%Node Mapping %%%%%%%%%%%%%%%%%%%%%%%%
    neigh_opt_1=result(neigh_opt_1_2,1);
    neigh_opt_2=result(neigh_opt_2_2,1);
    temp1=find(result(result(:,1)==neigh_opt_2,:));
    temp2=result(result(:,1)==neigh_opt_2,temp1);
    temp3=find(result(result(:,1)==neigh_opt_1,:));

result(result==neigh_opt_1,length(temp3)+1:length(temp3)+length(temp2))=temp
2;
    result(result(:,1)==neigh_opt_2,:)=[]
 n=n+1
end
fq=q./nb;
% ec=length(x(x==1))
end
```

```
function [w]= merging2(i,j,x,nb)
neigh_opt_1=i;
neigh_opt_2=j;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%Merging%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
x(neigh_opt_1,:)=x(neigh_opt_1,:)+x(neigh_opt_2,:);
x(:,neigh_opt_1)=x(:,neigh_opt_1)+x(:,neigh_opt_2);
x(neigh_opt_1,neigh_opt_2)=0;
x(neigh_opt_2,neigh_opt_1)=0;
K = length(x);
for i=1:K
    x(i,i)=0;
end
rn=x(neigh_opt_1,:)+x(:,neigh_opt_1)';
x(neigh_opt_2,:)=[];
x(:,neigh_opt_2)=[];
x(x==2)=1;

%%%%%%%%%%%%%%%%%%%%%%%%%%%% Cardinality Expression%%%%%%%%%%%%%%%%%%%%%

% q=length(find(rn~=0));
q=sum(rn);
nb(neigh_opt_1)=nb(neigh_opt_1)+nb(neigh_opt_2);


%%%%%%%%%%%%%%%%%%%%%%%%%q(s) calculation%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
w=q/nb(neigh_opt_1);
end
```

Appendix (IIII)

HRP1^{+}control and HRP2^{+}control

Control step code for HRP1^{+}control and HRP2^{+}control

```
function [cric,ls]= control(x,result)
K = length(x);
[M N] = size(result);
cric=[];
for i=1:K
    x(i,i)=0;
end
for i=1:K
    ls(i,1:length(find(x(i,:)==1)))=find(x(i,:)==1);
end
 for i=1:M
     for j=1:N
         if result(i,j)~=0
             temp_x=result(i,j);
             ch=ls(temp_x,:);
             for k=1:length(ch)
                 if ch(k) ~= 0
                     [pr pc]=find(result==ch(k));
                     if pr~=i
                         [qr qc]=find(cric==temp_x);
                         if isempty(qr)
                            cric=[cric;temp_x];
                         end
                     end
                 end
             end
             temp_x=[];
             ch=[];
         end
     end
 end
 end
```