A DECENTRALIZED APPLICATION OF DYNAMIC PROGRAMMING TO COMMUNICATION NETWORK RECONFIGURATION

A Thesis
Presented to
The Graduate Faculty of The University of Akron

In Partial Fulfillment
of the Requirements for the Degree
Master of Science

Anthony R. Batey
August, 2022

A DECENTRALIZED APPLICATION OF DYNAMIC PROGRAMMING TO COMMUNICATION NETWORK RECONFIGURATION

Anthony R. Batey

Thesis

Approved:	Accepted:
Advisor Dr. Robert Veillette	Department Chair Dr. Robert Veillette
Committee Member Dr. J. Alexis De Abreu Garcia	Dean of the College Dr. Craig Menzemer
Committee Member Dr. Nghi H. Tran	Interim Director of the Graduate School Dr. Marnie Saunders
	Date

ABSTRACT

A decentralized framework for network optimization is presented for wireless sensing nodes. The wireless sensing nodes use a dynamic programming algorithm to choose optimal routes for data transmission from any network node to a specialized 'gateway' node that provides access to the wider internet. The dynamic programming algorithm is a variation of the Bellman-Ford algorithm and allows for the wireless sensing nodes to make decisions based on locally available network information, resulting in a decentralized routing algorithm. Routing decisions depend on the cost it takes to communicate from a node to a gateway, either directly or indirectly, using neighboring nodes as relay points. Nodes constantly share information with neighbors and when something effects the cost of a path, such as a node failure or the discovery of a less costly route, all nodes upstream along the existing path are made aware and re-route accordingly. A sample network is used to illustrate and verify the functionality of the proposed algorithm. The network and node decisions are simulated to show the evolution of the network routing decisions, and the simulation consistently shows the network converging to an optimal configuration. The speed of convergence depends on the order in which the nodes are assumed to attempt to establish and optimize their connections.

DEDICATION

I would like to dedicate this thesis to all those who have supported me while taking on this endeavor.

To my wife Sasha, for loving me and all that I do without question or hesitation. In helping me manage stress, properly manage priorities, and reminding me to take a break from writing to take a breath and have fun.

To my parents Doug and Jill Batey who have always encouraged me to continue learning, pursue my interests, and pushed me to produce work that is nothing shy of my best.

To my advisor Dr. Robert Veillette, and professors Dr. Jose Alexis De Abreu-Garcia and Dr. Michael French, for their mentorship, friendship, and showing me how satisfying it is to teach others.

To my siblings and friends, for being genuinely interested in what I am working on, for asking questions and challenging me on what I'm doing to make sure that I know what I'm talking about. For being patient and understanding when I was unable to hangout.

Finally, to my dogs Bubbles, Buttons, and Tilly for sitting on my feet and keeping warm while writing and researching and my cats Mocha and Mochi, for silently judging me when I consider taking "the easy way out."

TABLE OF CONTENTS

Page

LIST OF FIGURES

CHAPTER I

INTRODUCTION

A sensing network is generally expected to collect data from distributed sources and convey the data to a central location for analysis and decision making. As an example, consider a wireless sensor network deployed onto high-voltage transmission-line towers in order to observe and monitor the condition of the towers and transmission lines. Each sensor might be powered from energy harvested from the transmission lines [1] and could be able to record events that happen near the towers such as gunshots [2], lightning strikes [3], or partial discharge of the transmission lines [4]. Information from these sensors could help utility and power companies prevent transmission line failures by pinpointing the location of incipient faults on the transmission lines, assuming the data could be viewed or processed in a central database. In such a situation, it would be crucial to implement a communication network to convey the data to the central database in an efficient and reliable manner.

To allow the autonomous recording and processing of collected sensor data the wireless sensor network should be connected in some way to the internet. One solution would be to design each sensor node to access the internet directly, by a cellular connection or the like. However, establishing individual cellular connections is an expensive option to implement on a large number of individual sensors. It would be more practical to have just a few of the sensing nodes doubling as gateways to the

internet database with the other nodes forming a dedicated communication network to relay data to the gateways.

For such a design, an issue that needs to be resolved is how the data should be routed through the network to efficiently reach a gateway. This thesis proposes a routing algorithm based on principles of dynamic programming to ensure that an optimal path from each sensor to a gateway is realized. According to the routing algorithm, each sensor node figures out where to route information, with help from adjacent nodes, such that it will reach a gateway with minimum cost incurred.

The notion of optimal paths through a network depends on the definition of a cost associated with communication between any two nodes. Cost of communication within the wireless sensor network is a user-defined metric that can allow for network optimization based on preferred network characteristics such as received signal strength, required power to transmit, or latency. For the purposes of this discussion, consider a network in which each node keeps track of the signal strength of transmissions received from neighboring nodes. For any given node, the assumed cost of communicating with a given neighbor could be defined by an inverse relation to the strength of signals received from that neighbor. The cost function in such a network would then be related to the strength of the wireless connections between nodes, where stronger communication links are given lower link costs. For example, depending on its exact relationship to the received signal strength via a link, the link cost might be proportional to the power required for a node to transmit reliably to its neighbor.

Nodes that have no direct link to a gateway must rely on neighboring nodes to deliver their data. The total cost to send data from the sensing node to the gateway when

the two are not directly linked depends on the neighboring nodes and their further connections in the network. For a gateway connection using multiple nodes as relays, the total cost of communication to a gateway is considered as the sum of the link costs along the full communication path. Regardless of the exact definition of the cost function, the objective of the optimization will be to minimize this sum for transmissions from any node to a gateway. In the case where the link cost is proportional to the power required for reliable transmission, the optimization would result in communications that consume the minimum possible energy in their transmission over the full path from a node to a gateway.

The efficiency of the network will rely on the overall routing algorithm determined by the routing decisions of each node and will be quantified by the calculated sums of link costs. Using principles of dynamic programming, nodes will work collectively to configure the overall network such that the least costly path from any sensing node to a gateway is utilized. To understand the idea of how the configuration would work, consider a new node joining the network. Such a node must determine which available link it should use to transmit messages. To do this, it will begin by broadcasting a query that will be received by all established nodes within range. An established node receiving a query from the new node will report back with the value of its overall cost to communicate to a gateway. The new node will then consider the sum of that reported cost value and the cost of communicating via the link with that specific established node as a possible total cost of communication with the gateway. As the new node receives acknowledgements from the various established neighboring nodes on the

network, it will compare all total costs calculated and choose, from then on, to communicate through the link on the lowest-cost path to the gateway.

Until the network is optimized, the nodes within the whole network continue to work towards the optimal solution. During this process, new information that must be considered may become available to existing nodes. For example, a node will be able to overhear information sent from one neighboring node to another. If the overheard cost of communication is lower than the cost of the path by which the node currently communicates to the gateway the node should broadcast a new query in an attempt to find a less costly communication path. This sort of reconfiguration also allows the network to reliably reconnect a sensing node to a gateway even in the event of a link drop. During a link drop, a node is able to identify the faulty section if at any time it does not receive an acknowledgement from the neighbor to which it regularly transmits. After determining its previous link is no longer present, it must once again act as a new node joining the network and broadcast a query to attempt to reconnect. The network reconfigures, the nodes create less expensive connections to the gateway nodes and, in doing so, begin to converge towards an optimized network.

To illustrate the configuration and operation of such a self-optimizing network, a simulation is presented for 25 stationary nodes arranged in a 5×5 rectangular grid to represent a decentralized system with certain nodes configured to act as gateways. The simulation program consists of an iterative optimization algorithm. In each iteration, the program sequentially considers all nodes in the network. When the program considers a node, it checks for the gateway connection status of all available neighbors. It applies the Bellman Equation to the set of information gathered from each neighbor to calculate the

optimal path from the node being examined to a gateway. The calculation minimizes the sum of the cost of communication from the neighbor to a gateway and the cost of the link from the node to that neighbor. After the simulation has gone through this process for every node in the network, completing a full iteration, it proceeds to another iteration. The process terminates when the total gateway cost for all nodes no longer changes, indicating that the Bellman equation is satisfied at each node and the optimal network configuration has been achieved.

In all simulations conducted, the network routing configuration converges to an optimal solution, but the number of iterations it takes to converge depends on the link costs, on the number of gateways present in the system, and on the order in which the nodes are considered for each iteration. In a real-time decentralized implementation of the algorithm, there would be no strict order in which new nodes joining the network broadcast queries and make routing calculations, as the nodes would act asynchronously; therefore, the practical rate of convergence would be expected to differ from that found in the simulations. The observed convergence of the simulation leads us to believe that a network with the proposed structure will consistently re-optimize and reliably deliver data to the end user. The ability to configure on startup, reconfigure in the event of a link or node drop, and reconfigure after observing a competitive cost to the gateway between neighbors, pushes the network towards its optimal structure.

When the proposed routing algorithm is applied to a network, nodes that are closer to a gateway will end up being relay points for nodes farther away. Any change in a node's total cost to communicate to a gateway will change the total cost for all upstream nodes using it as a relay point. When a link cost changes, it is first observed by

a node that uses that link directly as the first hop along an optimal path. When that node transmits via the affected link, the acknowledgement that it receives will indicate that a change has occurred, and the node will update its own cost or begin the query process if there is no longer a connection to the gateway. Now, when a node farther upstream transmits to a node that first observed the change in the network, it in turn observes that the total cost to the gateway has changed and updates its own cost to the gateway accordingly. With this process of observing and updating, information of the cost to the gateway is communicated to nodes upstream and farther away from a gateway. With this approach, the information propagates through the network the same way information propagates in optimal control problems using a Bellman equation, where optimization calculations are made backwards in time. [5]

The remainder of this thesis establishes a decentralized method to optimize a network. Desirable characteristics of existing network structures, network parameters used in routing calculations, and existing algorithms that optimize similar structures are discussed in Chapter 2. The Principle of Optimality is introduced in Chapter 3 along with a form of the Bellman equation suited to the routing optimization. Examples are presented to demonstrate how these ideas are crucial to the operation of the network optimization. Chapter 4 discusses the decentralized application of the developed routing algorithm and nodal decision-making process. Chapter 4 also documents a simulation of the decentralized routing algorithm resulting in a solution that matches the optimal network structures verified in Chapter 3.

CHAPTER II

LITERATURE REVIEW

The structure of a wireless sensor network of the kind considered in Chapter 1 resembles that of a wireless mesh network or a wireless ad-hoc network. This chapter compares the proposed network structure with those of the mesh and ad-hoc networks, and discusses some of the performance indices and designs that have been presented in the literature for their optimization.

A wireless mesh network is a communication network where wireless devices connect to one another using a mesh topology. A mesh topology describes an infrastructure in which network devices connect directly, and dynamically to as many other devices as possible in order to route data to and from clients. In wireless mesh networks, data is delivered from point A to point B by using dedicated clients, gateways, bridges, routers, and routing tables, to minimize the number of hops [6]. They provide secure connectivity between devices due to the stationary devices that create the infrastructure of the network. Redundancy can be introduced to the network by adding more gateways, bridges, and routers, thereby increasing routing efficiency and allowing the network to self-heal in the event of a device failure.

An ad-hoc network is a communication network that allows devices to communicate with one another without a pre-determined infrastructure [7]. Networks can make use of routers to manage communication between devices, but for an ad-hoc

network, devices use one another to relay data to the proper destination. This creates dynamic, constantly reconfiguring, self-organizing, and self-healing networks that allow devices to freely join and leave the network. Similar to mesh networks, ad-hoc networks become more efficient as redundancy is introduced.

The proposed wireless sensor network structure will resemble both network types in the sense that any node can connect to any other; however, instead of dedicated routers and bridges as found in mesh networks, each individual node will make its own routing decisions, choosing the link to the neighboring node that provides the least costly path to the gateway. The proposed network structure is also similar to that of an ad-hoc network, in that it is self-organizing and allows the sensor nodes to freely join and reconfigure creating a dynamic network. Where it differs from an ad-hoc network is the use of gateway nodes that are assumed to remain stationary. With a customized routing protocol focused on low cost of communication, the self-organization of the network will optimize the paths from nodes to gateways. Under the proposed structure, the network of data-collecting nodes may be dynamic and self-healing, like in an ad-hoc network.

The cost values used for the optimization of the proposed network can be derived from any quantifiable network parameters. Routing Information Protocol (RIP) optimizes a network by minimizing the number of hops to the gateway [8]. The routing metric used for Interior Gateway Routing Protocol (IGRP) uses a combination of bandwidth, delay, utilization load, and reliability of links [8]. The network optimization proposed in this thesis can apply any routing metric, assuming it is presented as a real, positive number, to establish connections between nodes.

When establishing connections in the dynamic network, nodes will make calculations independently from one another as seen in link-state protocols such as Intermediate System-Intermediate System (IS-IS) or Open Shortest Path First (OSPF). IS-IS is a routing protocol that relies on flooding routers in a network with link information of the entire network allowing routers to calculate the network optimization [9]. OSPF uses link-state databases to compute and distribute information regarding the changing network topology [9] and is applied by all routers on a network with parameters weighted by the user to achieve a more customized optimal network structure. While the IS-IS and OSPF achieve optimal solutions, routers computing the optimal network structures require a significant amount of memory and processing power to reach the proper conclusion. By comparison, the algorithm proposed here dramatically reduces the amount of memory needed per node, down to only a few bytes, as calculations depend on information shared among neighboring nodes rather than on global system information.

Regardless of the performance metric chosen, the proposed routing algorithm becomes a way to solve a shortest path problem. The shortest path problem involves finding a path between nodes where the sum of link costs or weights between the nodes along a communication path is minimized. Dijkstra's algorithm is a popular approach to solving these types of problems [10]. When Dijkstra's algorithm is implemented, a single node evaluates the link costs between two points and distributes that data to the rest of the established network [11]. OSPF and IS-IS use Dijkstra's algorithm on specific routers to calculate costs within the network [9]. The calculations and decisions are based on the vertices with the lowest costs to unvisited nodes and when a node's connection has been established, it is not revisited.

The Bellman-Ford algorithm is another approach to solving shortest path problems. The algorithm primarily looks at systems where there is a single source node and works outward from that source node to form the solution. A table is developed by one node and shared between neighbors; as nodes receive tables, they calculate the shortest route to the source node [11]. RIP and IGRP are just two examples of protocols that apply a variation of the Bellman-Ford algorithm to achieve optimality.

In the implementation of the Dijkstra and Bellman-Ford algorithms, all decisions are made by specialized agents such as routers, which then route data or communicate the routing structure to all nodes in the network. By contrast, the structure of the proposed algorithm is a more decentralized one, in which each individual node makes its own decisions to achieve the same network optimization. The decentralized calculations of the nodes produces the effect of solving the optimization problem, starting from the gateway node and proceeding outward, away from the gateway. It does not utilize tables thereby reducing the amount of information that needs to be retained and transmitted by each node. Nodes continue to update their cost estimates and routing decisions until the Bellman equation is satisfied at all points within the network, at which time the system has converged to the optimal solution.

The optimal routing structure of a given network describes a set of data paths that result in the least costly path from any node to a gateway. For simple networks within this thesis, an optimal solution will be reached by inspection and verified by applying the Bellman equation. By checking the Bellman condition at each node, the optimality of any given routing scheme can be easily verified, no matter how the design was produced.

The resulting network is the solution where all nodes have made the decision to take the least costly path to a gateway.

In Chapter 3, two sample networks are created to demonstrate the propagation of cost information throughout a network, how the Bellman Equation optimizes network routing configurations, and how routing decisions are made at each node.

CHAPTER III

OPTIMAL SOLUTION

The process of solving a dynamic programming problem is essentially the repeated application of the Bellman Equation. In this chapter, the condition for optimal routing is first discussed conceptually, and a corresponding form of the Bellman Equation is introduced. A sample network is defined, and the routing configurations are optimized by inspection and verified using that form of the Bellman Equation. Faults are introduced to the network to highlight certain properties of the optimization process and to discuss the behavior of the resulting routing configuration.

The Bellman Equation

Bellman's principle of optimality states that "An optimal policy has the property that no matter what the previous decisions (i.e., controls) have been, the remaining decisions must constitute an optimal policy with regard to the state resulting from those previous decisions" [12]. This limits the number of possible solutions that need to be investigated and implies that optimal solutions are determined by working backwards from system endpoints [12]. Algorithms to solve dynamic programing problems apply the Bellman equation, which is based on the principle of optimality, to compute optimal paths at each point in the system. The calculation of the solution begins at the endpoint of an optimal path, as implied by the principle of optimality, and works backwards

towards all possible initial points thereby eliminating non-optimal paths. The optimal path beginning at any point minimizes the total cost from that point to a desired final point. The Bellman Equation will be used to optimize a routing problem for computer networks where paths are represented as successive communication links and a point is a network node where data originates.

When solving an optimization problem, it is necessary to identify the boundary conditions or the gateway nodes to which all nodes are attempting to connect. For simplicity, let us consider an example as illustrated in Figure 1 where nodes happen to be arranged in layers. A node in a given layer can receive messages only from nodes in the previous layer and transmit messages only to nodes in the next layer. Nodes are assumed not to communicate with other nodes in the same layer. The nodes in Layer Z are to be the gateway nodes to which all other nodes must communicate. The layered structure of this example network is chosen to simplify the illustration.

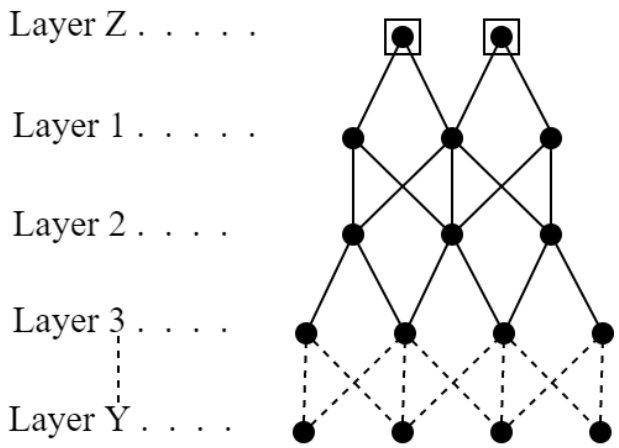


Figure 1. Layered Network Structure.

The starting point for this analysis is either of the gateway nodes in Layer Z to which all nodes are trying to connect. The first set of connections will occur between the

nodes in Layer 1 and a node in Layer Z. Assuming Layer 1 nodes do not connect to another point within the same layer, a Layer 1 node will make a direct connection to a gateway. The leftmost and rightmost nodes in Layer 1 each has only one available link to Layer Z, making that link the optimal connection for those Layer 1 nodes. It can be observed that the center node in Layer 1 can choose either the left or right node in Layer Z; for optimality, that node should choose the less costly of the two available links to a gateway.

Given that the nodes in Layer 1 have determined the optimal transmission paths to the gateway layer, the nodes in Layer 2 can now determine their optimal paths via relay nodes in Layer 1. All Layer 2 nodes must connect to a gateway in Layer Z in the most efficient way meaning they need to minimize the summed cost of the available links from Layer 2 to Layer 1 and of the subsequent optimal paths already chosen from Layer 1 to Layer Z. Nodes in each subsequent layer continue to choose links that minimize the cost from their position on any Layer Y to a Layer Z node.

The decision-making process that produces an optimized network can be expressed mathematically. Let a given node be denoted as node m, where m is an integer index. Let the cost of communicating from node m to a gateway node Z by some path be denoted as J_m. The value of J_m depends on the communication path chosen. If the lowest-cost, or optimal, path from node m to gateway Z is chosen, the corresponding optimal cost of communication is denoted as J_m^0. Suppose that node n is a neighbor of node m, and that the cost of transmission from m to n is denoted as $L[n, m]$. Then the cost of communication along any path from m, through n, to the gateway Z can be calculated as $J_m = J_n + L[n, m]$. If node n communicates along an optimal path to Z,

then $J_m = J_n^0 + L[n,m]$. Further, if each of the neighbors of node m communicates along an optimal path to Z then the optimal cost of communication from m to the gateway can be calculated as

$$J_m^0 = \min_n \{ J_n^0 + L[n,m] \} \quad (1)$$

This is the Bellman equation, which embodies the principle of optimality and is a condition that is satisfied at each node in an optimized network. A boundary condition for the Bellman equation is imposed at each gateway node Z, as $J_Z = 0$. Starting with Layer Z nodes and moving to layers successively farther away from Layer Z, Equation 1 is applied to all nodes in a network minimizing the cost to a gateway, resulting in an optimal system.

If any node were to disappear from the optimized network, that change would affect the routing decisions of all nodes upstream along an original optimal path. If we assume a single Layer 1 node has disappeared, all Layer 2 nodes connected to that node realize the data is not being received and broadcast a query to find a new connection to a gateway. In turn all Layer 3 nodes connected to the affected Layer 2 nodes learn that the endpoint connection has disappeared. Thus, this information travels upstream through the affected portions of the network until it reaches the points in Layer Y. Figure 2 illustrates a possible response of the network in the event of a node outage in the middle of Layer 1. The first part of the figure illustrates an optimized network before any node failure. The second part shows the effects of the node outage. The upstream nodes affected by this outage are circled in red. These nodes must then establish new connections to nodes that connect to a gateway. The third part of Figure 2 illustrates how the resulting reconfigured network might turn out.

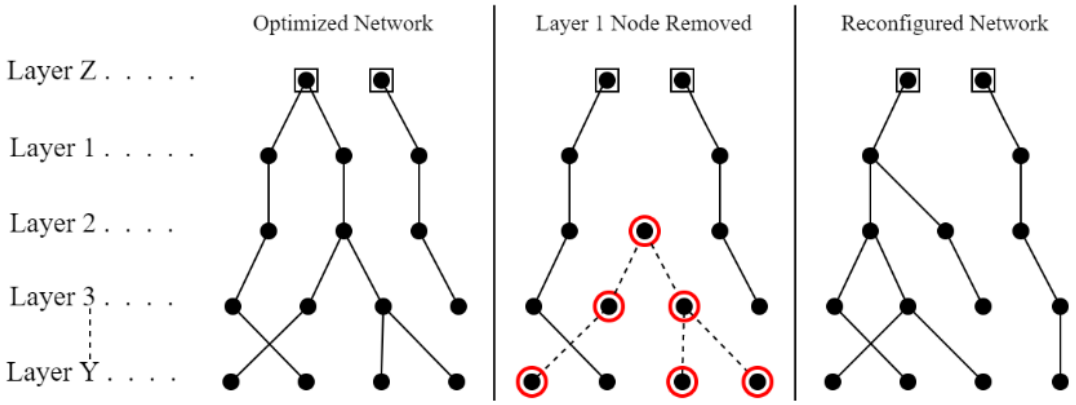


Figure 2. Illustration of Link Drop Information Propagation.

This layered network structure is helpful in showing how information propagates upstream in a network along with the overall network response when a node disconnects. For any given network, regardless of its structure, the upstream propagation of information is a crucial interaction that must occur between nodes for the network to reach an optimal routing configuration.

Network Structure and System Behavior

For the rest of this chapter, a 5×5 rectangular grid will be used to represent a wireless sensor network, such as Figure 3, with each dot representing a sensing node and each line representing a link between two nodes. Nodes are numbered 1-25, link costs are represented by the italicized numerals, and gateways to the database are indicated by the boxed dots at nodes 1 and 25. For the sake of simplicity, it is assumed that the cost of communication between every node pair is constant and independent of the direction of transmission. This network is the example problem used to illustrate the nature of the optimal solutions in this chapter and for simulating the network optimization via dynamic programming in Chapter 4.

Note that the grid shown in Figure 3 does not fit the layered network structure illustrated in Figures 1 and 2. It is a more general structure that incorporates significantly more redundancy of communication paths and does not restrict the communication between a given node and a gateway to use a specific fixed number of links. Link costs $L[n, m]$ between adjacent nodes are shown as italicized values below or beside a link. The Bellman equation is a valid representation of the principle of optimality for this more general network and will be applied to arrive at or verify an optimal design. In the previously discussed layered network configuration, indices m in the Bellman equation only needed to consider neighbors n that could be reached in the next layer. For the generally mesh topology, the indices n have to include all nodes within the transmission range of node m.

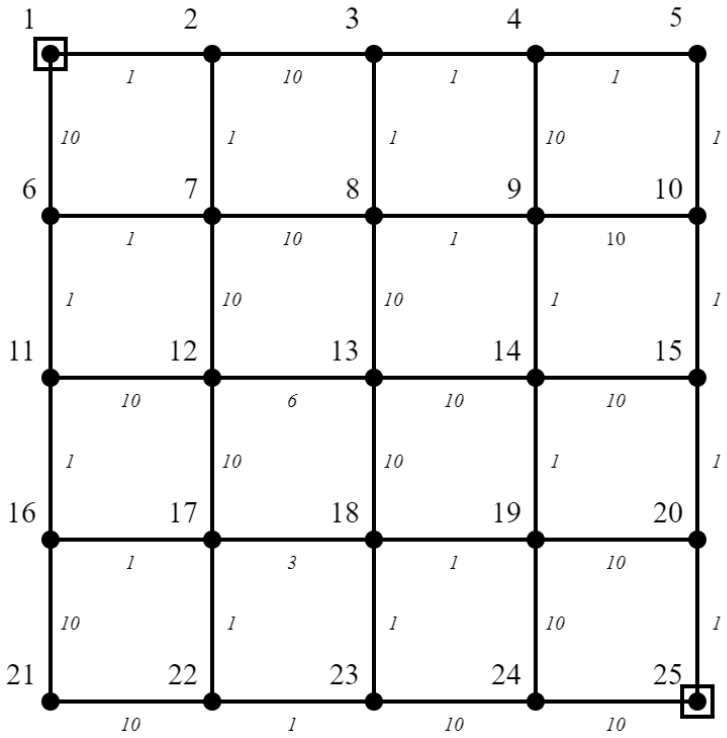


Figure 3. Grid Network Simulation Representation.

The cost of transmitting a message from each node to a gateway must be minimized for the network routing structure to be considered optimized. The optimization of the grid network is initially completed by inspection producing Figure 4. Next to each link in an optimal path, an ordered pair is displayed in italics. The first value in the pair indicates the cost of using that one link and the second value indicates the sum of the link costs along the optimal path from the transmitting node to the gateway.

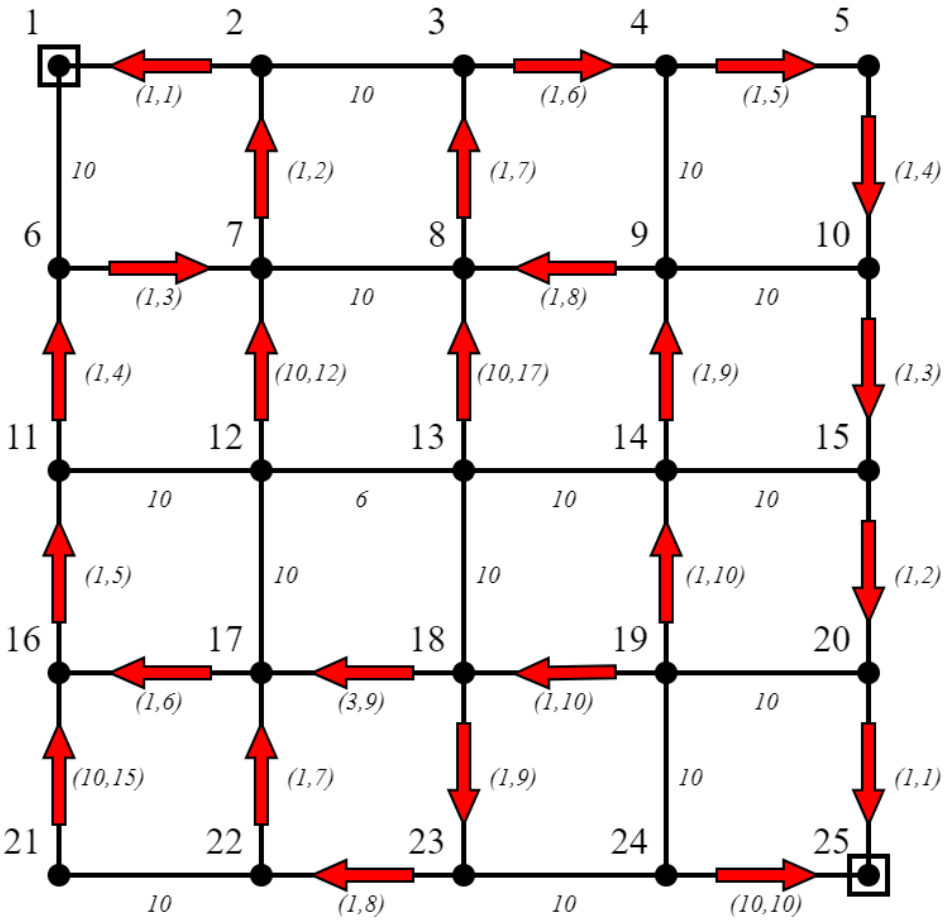


Figure 4. Optimal Paths for the Nominal Network.

For the network described in Figure 3, Equation 1 may be used to verify the optimal solution obtained via inspection in Figure 4. As an example, we now apply the Bellman equation to verify the optimal choice of the first relay point from node 14. Applying Equation 1 at node 14 and choosing n from a set of indices corresponding to nodes neighboring node 14 gives the equation

$$J_{14}^{0} = \min_{n \in \{9,13,15,19\}} \{ J_{n}^{0} + L[n,14] \} \quad (2)$$

It is assumed that the value of J_{n}^{0} has been previously determined at each of the neighboring nodes n. Expanding the set of indices n in Equation 2:

$$J_{14}^{0} = min\{ J_{9}^{0} + L[9,14], J_{13}^{0} + L[13,14] , J_{15}^{0} + L[15,14] , J_{19}^{0} + L[19,14] \} \quad (3)$$

$$J_{14}^{0} = min \{1 + 8, 10 + 17, 10 + 2, 1 + 11 \} \quad (4)$$

$$J_{14}^{0} = min \{9, 27, 12, 12 \} = 9 \quad (5)$$

The cost to transmit from node 14 to a gateway through each neighbor can be seen in Equation 5 before minimizing the set of data. It also shows that the optimal path for node 14 has a total cost of 9 which results from using node 9 as the first relay point to a gateway.

It is important to observe that a network node may have more than one optimal communication path available. Figure 4 shows, for example, that node 18 has the flexibility to choose between two optimal paths. The optimal paths and associated costs are observed when Equation 1 is applied at node 18:

$$J_{18}^{0} = \min_{n \in \{13,17,19,23\}} \{ J_{n}^{0} + L[n,18] \} \quad (6)$$

Expanding the set of indices n in Equation 6:

$$J_{18}^{0} = min\{ J_{13}^{0} + L[13,18], J_{17}^{0} + L[17,18] , J_{19}^{0} + L[19,18] , J_{23}^{0} + L[23,18] \ \} \quad 7$$

$$J_{18}^{0} = min\{10 + 17, 3 + 6, 10 + 1, 8 + 1 \ \} \quad 8$$

$$J_{18}^{0} = min\{27, 9, 11,9 \} = 9 \quad 9$$

The result shows that node 18 can establish a gateway connection with the minimum cost of 9 by transmitting to either node 17 or node 23. A similar scenario can be observed at node 19 but with each initial link having an identical cost. Regardless of the decision, the Bellman Equation is satisfied at all points within the network ensuring optimality.

Nodes will not necessarily choose the lowest-cost initial link but rather the one satisfying the Bellman equation, which will lead to a globally optimal solution. This can be seen at Node 12 in Figure 4, where the optimal path starts with a link of cost 10 even though a local link with a cost of 6 is available.

In the event a node fails to transmit or receive data, the network must reconfigure its routing structure. The optimal reconfiguration may again be determined by inspection of the faulty network. Assuming, for example, that nodes 8 and 16 become unable to transmit messages, the network routing structure should reoptimize into what is shown in Figure 5. The links around nodes 8 and 16 are all dropped and alternative links are chosen to provide optimal routing of transmissions to the available gateways. In the reoptimized faulty network, each node has a cost of communication to a gateway that is equal to or greater than it was in the fully functioning network. It can be observed that the optimal transmission paths can switch directions as links are dropped. For example, a

comparison of Figure 4 and Figure 5 shows that the direction of data transmission between nodes 9 and 14 is reversed when the network re-optimizes after the given fault, and similarly for nodes 22 and 23.

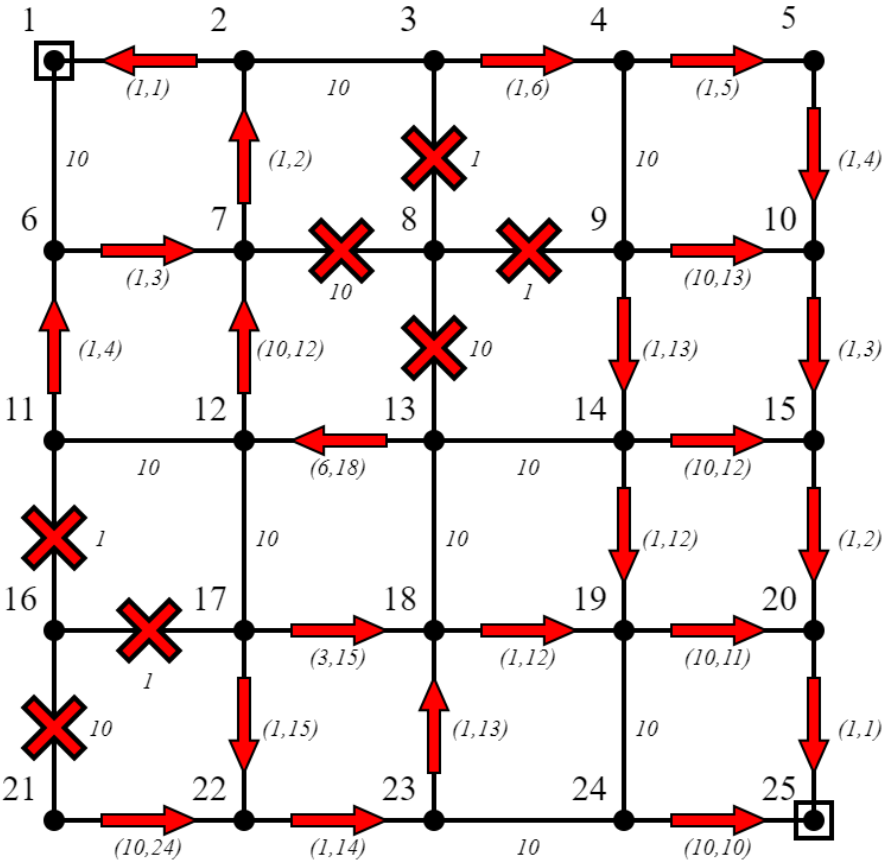


Figure 5. Reconfiguration Optimization.

In this chapter, the Bellman Equation to has been used to optimize the routing configuration of a wireless sensor network. A sample network was created optimized by inspection to illustrate how wireless sensing nodes could connect to one another. The result of the optimization via inspection was then verified mathematically. The optimized network structure was achieved using a centralized approach meaning one device, in this case a person solving via inspection, made all the calculations and routing

decisions. In a real-world application of the proposed design, each wireless sensing node would make its own routing decisions, creating a decentralized solution to the routing problem. Chapter 4 goes on to discuss how the nodes would operate in a real network and the process by which they would connect to one another. A simulation is conducted to verify that the decentralized application of the Bellman equation will result in the expected optimal routing solution.

CHAPTER IV

DECENTRALIZED IMPLEMENTATION AND SIMULATION

The establishment of optimal communication paths within the network is the result of the repeated application of the Bellman Equation by the network nodes. When a node detects a deterioration in conditions downstream along its established communication path or receives data indicating a lower cost alternative path may be available, the node queries for information from neighbors and re-applies the Bellman Equation. The node will either update its already established path cost or establish a new connection to a different neighbor, thereby creating an optimal path. All network nodes use the same decision-making algorithm when establishing connections to neighbors.

Decentralized Application

In an unoptimized network, there will always be at least one route somewhere within the system that can be improved through rerouting. For example, early in the network configuration process, certain nodes may initially take on traffic enabling neighbors to transmit data to a gateway, but the first paths established may not be the final paths used to achieve global optimization. As more information becomes available and more nodes establish gateway connections, nodes may find it advantageous to transmit data to different neighbors after the Bellman Equation is applied to the set of updated information.

The convergence of the decentralized routing algorithm to an optimal routing solution relies on network nodes, m, to apply the Bellman Equation to communication-cost data, $J_n + L[n, m]$, gathered from neighboring nodes, n, to determine the link along the optimal communication path, and total cost to a gateway, J_m. In the proposed design, the optimization of the network depends on the repeated upstream propagation of the necessary re-calculations that occur under the conditions described in the following paragraphs.

The independent nodes that constitute the decentralized network obtain cost data from each other by a process of query and response. The query process allows for nodes to establish, re-establish, or improve upon network connections to achieve an optimal network structure. The process begins when a network node m needs to connect to a gateway and requests information from neighbors within its transmission range. Each neighbor that receives the query responds with its address n, its link cost $L[n, m]$ to the querying node, its gateway connection status, and its communication cost J_n to a gateway. When real wireless sensing nodes are deployed, each will have its own unique wireless address adhering to the chosen form of communication, such as Bluetooth, Wi-Fi, or LoRaWan. For the simulation of the decentralized application, the address of a node is represented simply by its index number.

When a neighbor responds to a query with a gateway connection status of TRUE, meaning that is has an established communication path to a gateway, the querying node saves the address n of that neighbor and the corresponding value of J_n as one term to be used in the Bellman equation. After a configurable amount of time or a broadcast limit has been reached, the node stops querying for data from neighbors and applies the

Bellman Equation to the set of collected addresses and communication costs, to determine its lowest available cost of communication $J_m = J_n + L[n, m]$. The corresponding address n to which node m should transmit to achieve that lowest cost is denoted as u_m. These operations of a node broadcasting a query are shown in Figure 6.

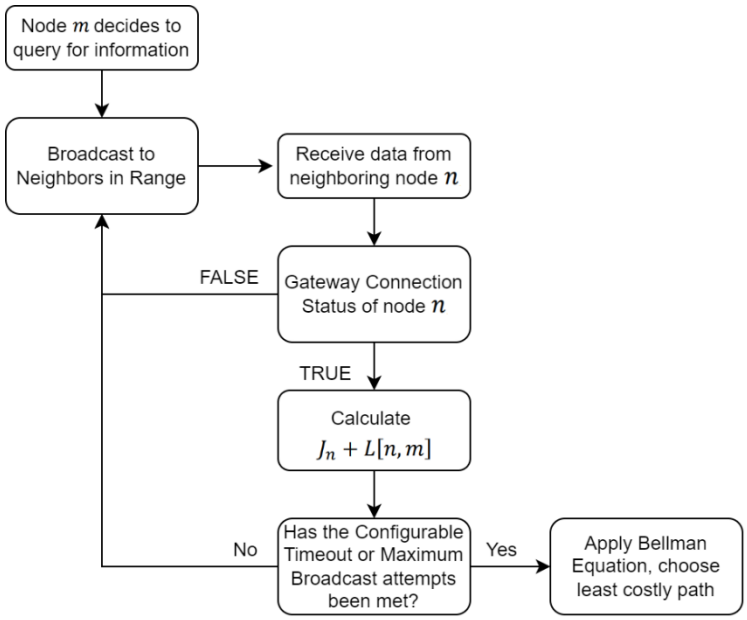


Figure 6. Querying Node Decision Chart.

There are three conditions, shown in Figure 7, that cause a node to query for information from neighbors: (1) if the node is joining or re-joining the network, (2) if the node encounters an error when attempting to use an established link, or (3) if the node overhears data indicating the availability of an alternative path with a competitive cost to a gateway. The first of these three conditions is rather obvious, as a node initially joining or re-joining the network must request information to determine the least costly transmission path to a gateway. Having done this, a node should always transmit via this chosen path until some change in the downstream conditions requires a new optimal path

to be sought. A node becomes aware of such a change in the downstream conditions when it encounters the second or the third condition.

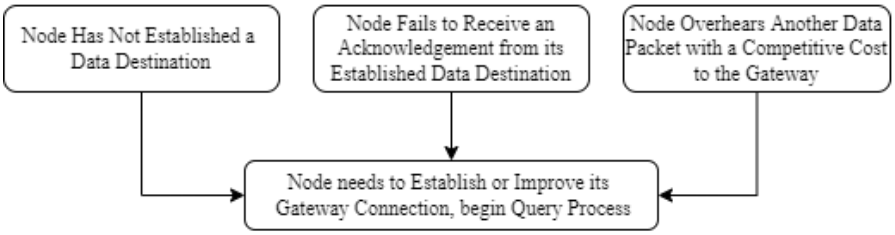


Figure 7. Node Query Conditions.

The second condition under which a node begins to query for link information occurs when a fault is observed. When a node does not receive an acknowledgement from an established neighbor to which it has been transmitting, it can infer that the neighbor has been disconnected from the network or has experienced a failure and is no longer able to communicate. At this time, the node will broadcast a query to establish a connection to a different neighbor. Until it establishes a new connection, the querying node sets its own gateway connection status to FALSE. If at this point an upstream neighbor attempts to transmit data through the querying node before a new connection is established, the neighbor receives an acknowledgement indicating there is no gateway connection and will begin the query process as well.

The final condition occurs when a network node overhears, from the header of a data package or acknowledgement addressed to a neighbor, a competitive cost to communicate to a gateway. This helps to ensure that the optimal routes are maintained and updated to take advantage of redundant paths that may have lower costs, and also

eliminates any routing loops that may have formed in the network. If a network node overhears a total cost to communicate to a gateway, $J_{Overheard}$, that is at or below a query activation threshold, J_{th}, the node begins the query process in an attempt to establish a more efficient link. The query activation threshold could be given by the expression $J_{th} = J_{Node} \times (110\%)$ where J_{Node} is a node's total cost to communicate to a gateway. With this established threshold, a node would begin to query if the overheard cost satisfies $J_{Overheard} \leq J_{th}$.

There could be a situation where a node continuously overhears a desirable cost from a neighbor, triggering the query process, only to find the original path is still the optimal one. For this to happen the cost from the neighbor to the gateway would be below the node's J_{th}, only to find the addition of the transmission cost to the neighbor creates a total gateway cost that is greater than what was originally established. In this case, a node could remember the neighbor that transmits the competitive, but not optimal, cost and reduce J_{th} specifically for that neighbor.

There could be such an event that a node observes a link drop, re-connects to a new node, and updates it total cost information, presumably to a higher value than before, before any neighbors take notice. In this case, the neighbors will become aware of updated cost information through the reconnected node's acknowledgements. Now having an updated cost to the gateway, neighbors are more likely to query and optimize their path as they overhear competitive cost information. When the neighbors indicate a change in the cost to a gateway, all other nodes upstream from that point are made aware of the new cost and may repeat this process. In this way, the observed status and total

cost changes are able to propagate through the network nodes as illustrated by Figure 2 in Chapter 3.

Simulation of Decentralized Operation

The intent of implementing the algorithm described above on each network node is to achieve an optimized routing configuration in the wireless network. To verify the effectiveness of the algorithm, a program is used to simulate its application on nodes operating within a decentralized network. The simulation demonstrates that nodes will drop or change links when lower cost links are observed, thereby improving network efficiency as more knowledge about the surrounding neighbors is acquired.

Five-node Network Example

To show how the process of the decentralized simulation operates, a simple five-node network, illustrated in Figure 8, is used. The simulation uses a set of vectors and variables to properly emulate events that would occur in an actual application thereby simulating a decentralized system. An adjacency matrix may be constructed to illustrate how all nodes are linked to one another.

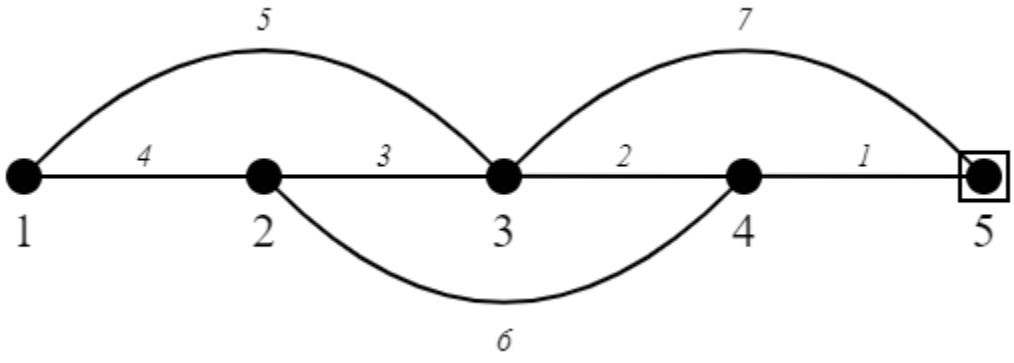


Figure 8. Five-node Network.

The adjacency matrix A, is described as an $M \times M$ matrix where M indicates the total number of nodes in the network. With m representing a node and n an adjacent node, the individual entries $a[n,m]$ for A are given by [13]

$$a[n,m] = \begin{cases} 1, & \text{if nodes } m \text{ and } n \text{ are adjacent;} \\ 0, & \text{otherwise.} \end{cases}$$

In our case, a weighted adjacency matrix will be used, where the matrix entries indicate instead the link costs in the network, as [13]

$$a[n,m] = \begin{cases} L\,[n,m], & \text{if nodes } n \text{ and } m \text{ are adjacent;} \\ Inf, & \text{otherwise.} \end{cases}$$

Using Figure 8 as an example, the corresponding weighted adjacency matrix is

$$A = \begin{bmatrix} 0 & 4 & 5 & Inf & Inf \\ 4 & 0 & 3 & 6 & Inf \\ 5 & 3 & 0 & 2 & 7 \\ Inf & 6 & 2 & 0 & 1 \\ Inf & Inf & 7 & 1 & 0 \end{bmatrix}$$

The weighted adjacency matrix describes the cost of communication between each pair of nodes within the network. The matrix rows correspond to nodes transmitting data and the columns correspond to the neighboring nodes receiving the data. Looking at $a[1,2]$ the value entered is 4, meaning that the cost of the link from node 1 to node 2 is 4. The link between any two given nodes is assumed to have the same cost, regardless of the direction of the transmission; as a result, the matrix is symmetric. The matrix entry corresponding to any pair of non-neighboring nodes takes a value of infinity and the diagonal elements are set to zero to indicate there is no cost associated with any node transmitting to itself. Stronger connections between nodes are indicated by smaller positive numbers.

During the simulation, a vector V_{GCS}, the gateway connection status vector, is used to keep track of the gateway connection status of all simulated nodes and another vector V_{QE}, the query enable vector, tracks which nodes need to query to establish or update a connection. V_{GCS} and V_{QE} are composed of Boolean logic bits that represent the gateway connection status and query enable of each simulated node. The first step in the simulation is to confirm that the appropriate V_{QE} element for the simulated node is TRUE. When it is, the simulation emulates the operation of the actual decentralized behavior of gathering cost information from all neighbors by referencing the adjacency matrix and the cost vector. If the simulated node gathers data from a neighbor and the corresponding V_{GCS} element is FALSE, the cost data and address of the neighbor is not considered when the routing optimization calculation occurs. After the simulated node establishes a connection to the gateway, the V_{GCS} element for the node becomes TRUE and the V_{QE} element changes to FALSE, emulating the end of the querying process. In the simulation, the query enable status of a node is set to TRUE if any simulated neighbors made an improvement to their total gateway cost. This is done to simulate the query condition where a node overhears a competitive cost to the gateway.

Considering the sample network in Figure 9 the initial V_{GCS} and V_{QE} vectors would be represented as

$$V_{GCS} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \qquad V_{QE} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

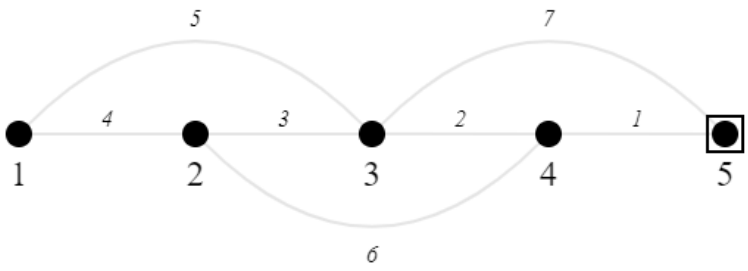


Figure 9. Five-node Network Solution, Initial Setup.

At the beginning of the simulation V_{GCS} and V_{QE} are initialized to show that the simulated network nodes are not connected to a gateway and will emulate the decentralized query process when prompted. Since gateway nodes are the ends of the network transmission paths, they will never simulate a query, nor will they change their gateway connection status.

When the program simulates a query by node m, corresponding to row m of the adjacency matrix, each other node in the network, represented by a column n of the adjacency matrix, with $m \neq n$, is checked to see if there is a connection. To determine a connection, the program looks for any finite values greater than zero for all elements n in row m. The minimum cost to a gateway from node m, J_m, is calculated using Equation 1. The neighbor that provides an optimal path for the simulated node will be recorded as u_m. A Performance Index Vector J is created to contain the total cost values from any given node in a network to a gateway as

$$J = \begin{bmatrix} J_1 \\ J_2 \\ \vdots \\ J_M \end{bmatrix}$$

A Control Vector u is used to indicate the corresponding addresses u_m to which each node should transmit to achieve the total costs listed in J, as

$$u = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_M \end{bmatrix}$$

Considering the network in Figure 9 at the beginning of the simulation, J and u take the initial values

$$J = \begin{bmatrix} \text{Inf} \\ \text{Inf} \\ \text{Inf} \\ \text{Inf} \\ 0 \end{bmatrix} \qquad u = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

To demonstrate how the simulation will solve a network, let us work through the optimization of the five-node network proposed in Figure 8. Initially, nodes have not begun establishing connections using the available links. In Figure 9 available links are shown as grey lines between adjacent nodes. The program considers the nodes in order, as if each node is querying for a connection to a gateway beginning with node 1 and ending with node 4 where node 5 acts as the gateway. The first iteration of the simulation results in Figure 10. The values inside the parentheses use the same notation as the examples considered in Chapter 3 where the first value is the cost of the link and the second value is the minimum total cost found from the transmitting node to the gateway.

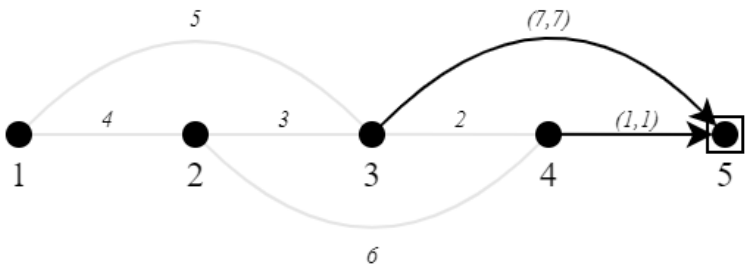


Figure 10. Five-node Network Solution, Step 1.

At the end of the first iteration, two links are assigned as connections from nodes 3 and 4 to the gateway. The vectors V_{GCS}, V_{QE}, J, and u are updated as

$$V_{GCS} = \begin{bmatrix}0\\0\\0\\0\\0\end{bmatrix} \rightarrow \begin{bmatrix}0\\0\\1\\1\\1\end{bmatrix} \qquad V_{QE} = \begin{bmatrix}1\\1\\1\\1\\0\end{bmatrix} \rightarrow \begin{bmatrix}1\\1\\1\\1\\0\end{bmatrix} \qquad J = \begin{bmatrix}\text{Inf}\\\text{Inf}\\\text{Inf}\\\text{Inf}\\0\end{bmatrix} \rightarrow \begin{bmatrix}\text{Inf}\\\text{Inf}\\7\\1\\0\end{bmatrix} \qquad u = \begin{bmatrix}0\\0\\0\\0\\0\end{bmatrix} \rightarrow \begin{bmatrix}0\\0\\5\\5\\0\end{bmatrix}$$

Nodes 1 and 2 are not connected in this step because, in this iteration, they were considered first, before nodes 3 and 4 had established a connection.

The result of the second iteration is shown in Figure 11 with the vectors V_{GCS}, V_{QE}, J, and u updated as

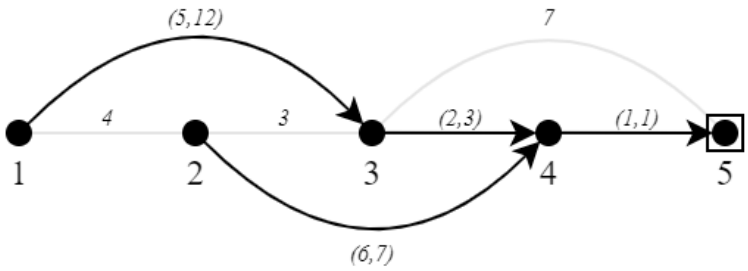


Figure 11. Five-node Network Solution, Step 2.

$$V_{GCS} = \begin{bmatrix}0\\0\\1\\1\\1\end{bmatrix} \rightarrow \begin{bmatrix}1\\1\\1\\1\\1\end{bmatrix} \qquad V_{QE} = \begin{bmatrix}1\\1\\1\\1\\0\end{bmatrix} \rightarrow \begin{bmatrix}1\\1\\1\\0\\0\end{bmatrix} \qquad J = \begin{bmatrix}\text{Inf}\\\text{Inf}\\7\\1\\0\end{bmatrix} \rightarrow \begin{bmatrix}12\\7\\3\\1\\0\end{bmatrix} \qquad u = \begin{bmatrix}0\\0\\5\\5\\0\end{bmatrix} \rightarrow \begin{bmatrix}3\\4\\4\\5\\0\end{bmatrix}$$

V_{QE} indicates which nodes must query potentially update the J, u, and V_{GCS} vectors. It might appear by inspection of this result that there is an error with node 1's total gateway cost being 12 when it should be 8; however, this is not actually an error at this stage. Because the simulation makes one node calculation at a time, when the node 1 calculation was made, node 3 was still connected to node 5 using the more costly single-link path. After the calculation at node 1, the simulation toggles the node 3 V_{QE} element to TRUE. When the node 3 query process is simulated, a more efficient link is calculated triggering reconfiguration to use the optimal link. The final iteration shown in Figure 12 shows the simulation updating the cost at node 1 and finding a less costly path from node 2 to the gateway.

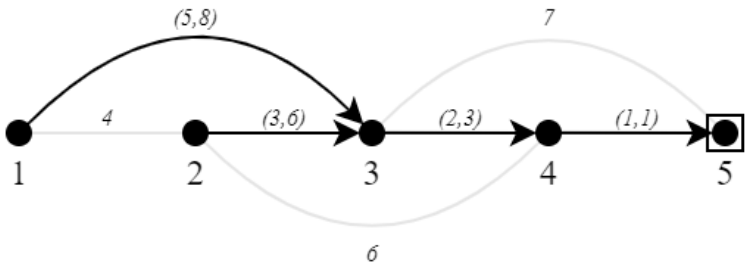


Figure 12. Five-node Network Solution, Step 3.

The vectors V_{GCS}, V_{QE}, J and u are updated as

$$V_{GCS} = \begin{bmatrix}1\\1\\1\\1\\1\end{bmatrix} \rightarrow \begin{bmatrix}1\\1\\1\\1\\1\end{bmatrix} \qquad V_{QE} = \begin{bmatrix}1\\1\\1\\0\\0\end{bmatrix} \rightarrow \begin{bmatrix}1\\1\\0\\0\\0\end{bmatrix} \qquad J = \begin{bmatrix}12\\7\\3\\1\\0\end{bmatrix} \rightarrow \begin{bmatrix}8\\6\\3\\1\\0\end{bmatrix} \qquad u = \begin{bmatrix}3\\4\\4\\5\\0\end{bmatrix} \rightarrow \begin{bmatrix}3\\3\\4\\5\\0\end{bmatrix}$$

To determine that the system has converged, the simulation does one more iteration. The network configuration for step 4 is the same as Figure 12, however V_{QE} changes to all zeros, showing that no improvements to the system have been observed and all nodes are done querying. This example demonstrates how the simulation optimizes a given network with the simulation operating as outlined in Figure 13.

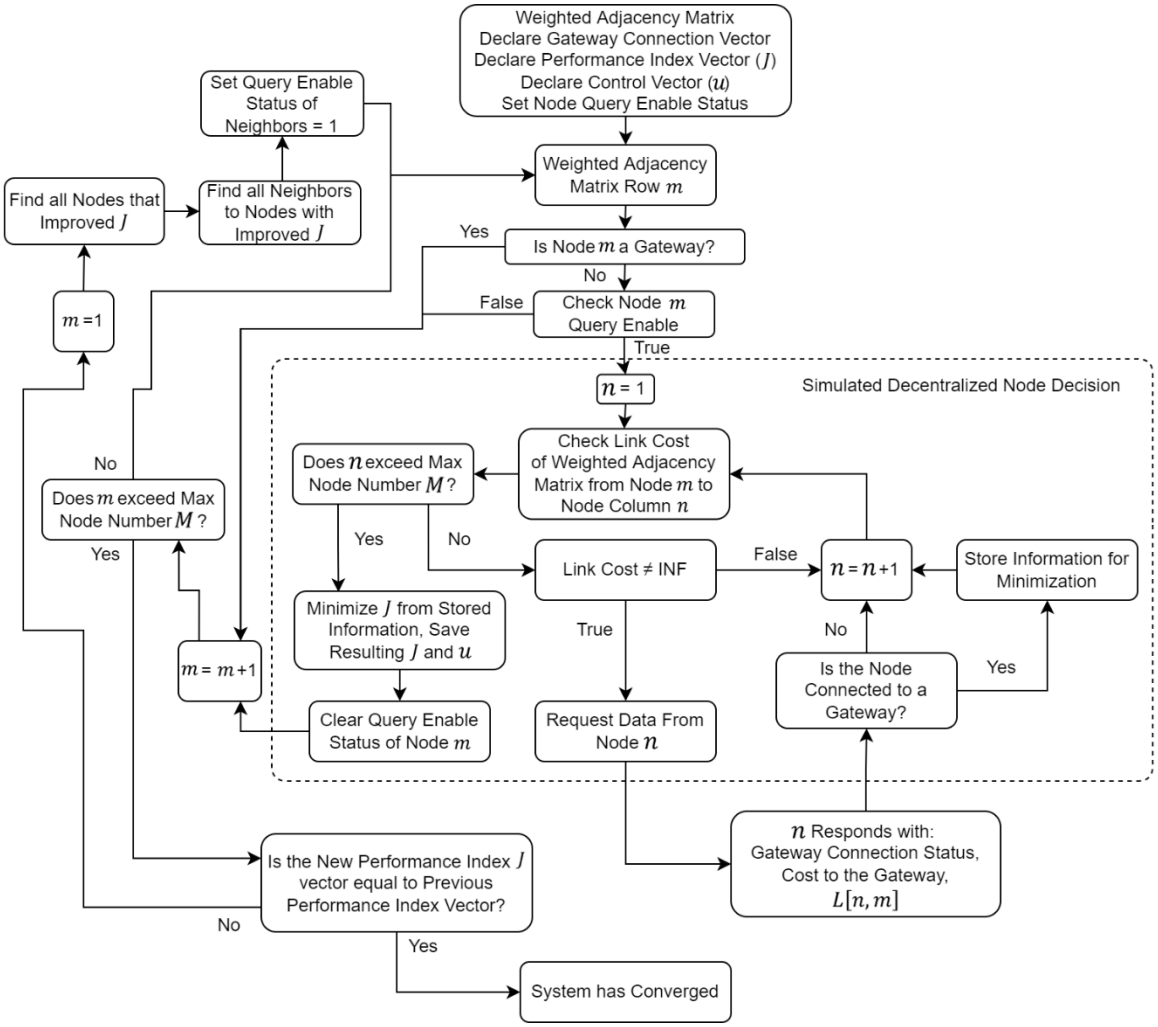


Figure 13. Simulation Decision Flowchart.

If node m is not a gateway and its querying status has been enabled, the simulation begins by checking each entry of row m of the adjacency matrix for any finite values to indicate links available to node m. The position, n, of any finite values within

the row are temporarily saved, along with the corresponding link costs found at those positions. These saved values are referred to as link cost pairs $(n, L[m, n])$. When all finite values within a row have been recorded, the system applies Equation 1 for the saved link cost pairs to determine the optimal communication path for the transmitting node. The query enable is cleared for the transmitting node after the optimal path has been chosen. An optimal link cost pair indicates that the neighbor corresponding to column n will receive all future data transmission from node m. To simulate the idea of nodes re-querying after overhearing competitive cost data, all neighbors to nodes that have experienced a change in J will have querying enabled. The simulation continues to run until the vectors J and u remain constant for two loop iterations. The Matlab code is shown in Appendix A at the end of the thesis.

Originally Proposed 25-Node Network

The algorithm as described for the five-node network can optimize larger networks of any node size. It is now applied to the 25-node 5×5 that rectangular grid network in Figure 3. For each iteration, the nodes are considered in descending order starting at twenty-five and ending at one. After all nodes have been checked, the current and previous performance index vectors are compared. If there is no change between the current and previous J the simulation ends, otherwise the simulation continues to run looking for nodes that have querying enabled.

Upon completion of the simulation, a directed graph, is constructed showing the nodes, gateways, total cost of communication to the gateway, and link direction. The rate of convergence of the simulated network is dependent on gateway locations, the number

of gateways, and the order in which nodes establish and re-establish connections. The simulation utilizes two gateways at nodes 1 and 25, calculates connections sequentially starting at node 25 for all nodes with querying enabled, and repeats until the performance index remains the same between two iterations. The output is an optimized network structure shown in Figure 14. To compare the following figure to the five-node network diagrams, the arrow shows the direction of communication using the link from a transmitting node m to a neighboring node n. J_m is the second value in the parentheses, and u_m is the node at the point of an arrow. For example, if we were to consider node 13 in Figure 14 we would get $m = 13,\ J_{13} = 17,$ and $u_{13} = 8.$

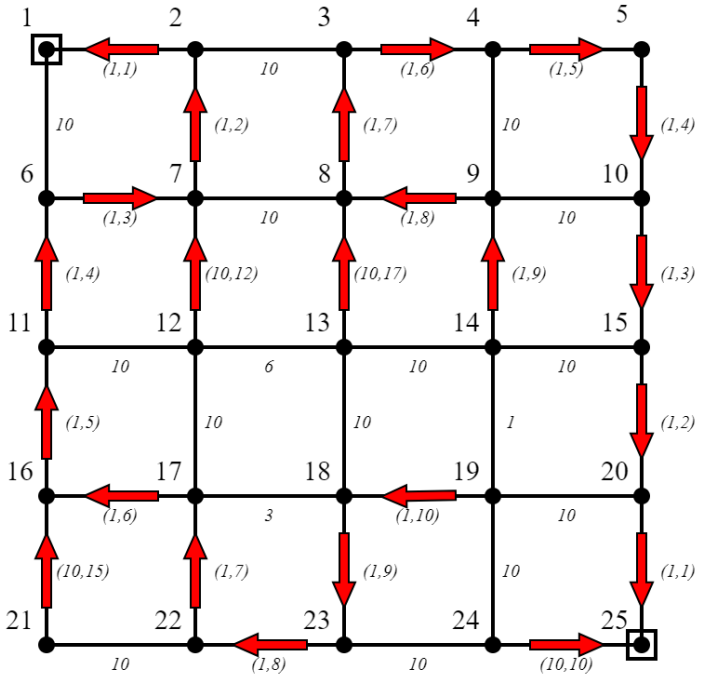


Figure 14. Simulation Output with two Gateways, nodes considered in order of descending indicies.

Figure 14 shows each node is connected to only one destination, whereas the optimal solution in Figure 4 shows some nodes with two distinct optimal paths to choose

from. In Figure 4, node 18 and node 19 are two such nodes. Figure 14 shows each of these two nodes choosing the link to the node with the higher index. This is due to the order in which the simulation considers nodes in row m. When the simulation considers nodes in descending order, the highest value node will be the chosen path as seen in Figure 14 whereas if nodes are considered in an ascending order, the lowest value node is chosen. This behavior shows that no matter the structure or order of nodes, the simulation will converge to an optimal solution. With that said, the order in which nodes are considered will change the convergence rate of the optimal solution. To show the differences in convergence rates, the simulation results are shown twice more with only one gateway at node 25. The first single-gateway simulation considers nodes in order of decreasing index where the second simulation considers node index in ascending order.

The single-gateway descending node index simulation begins at the gateway node 25 and ends at node 1. These network figures are like those used in Chapter 3 with the addition of arrow colors to distinguish how nodes connect and update as the simulation works toward an optimal solution. Red arrows indicate that a node is transmitting to the same neighbor as at the end of the previous iteration, and there is no change in the total cost to communicate to the gateway from one simulation iteration to the next. Yellow arrows indicate that the neighbor to which a node transmits remains the same, but the total cost to the gateway has been updated. Green arrows indicate a new connection between two neighboring nodes. The series of Figure 15 through Figure 20 shows how the nodes in the sample network connect to one another and reconfigure when a more optimal path is observed.

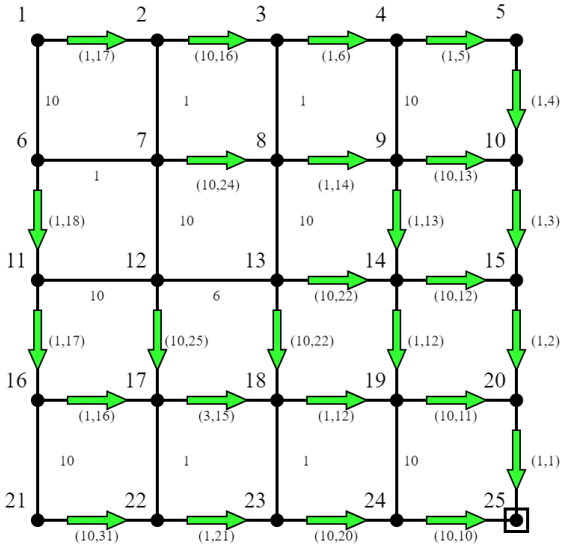


Figure 15. Intermediate result after 1st iteration, nodes considered in order of descending indices.

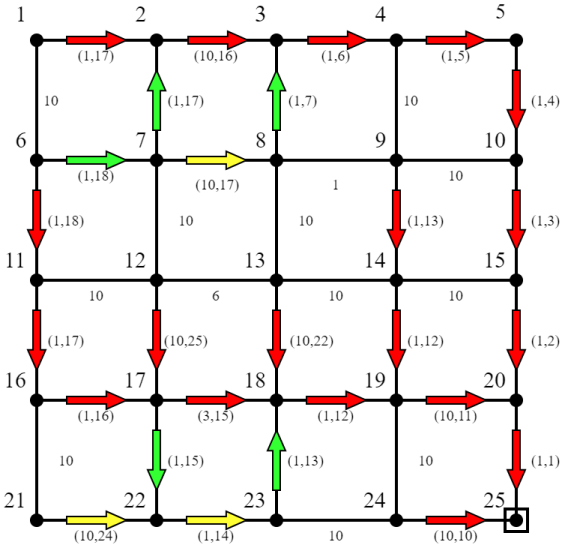


Figure 16. Intermediate result after 2nd iteration, nodes considered in order of descending indices.

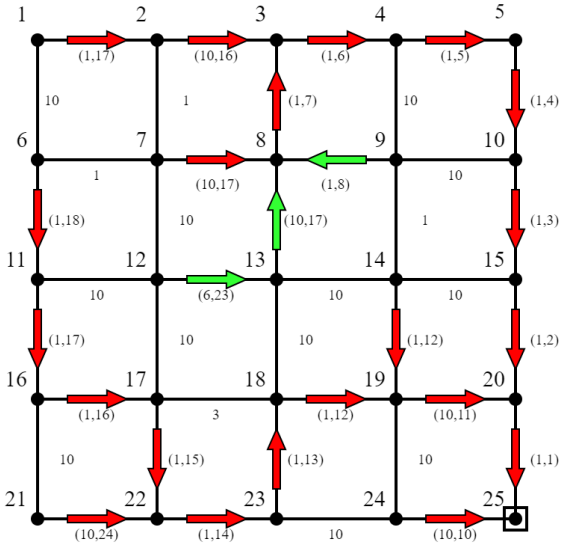


Figure 17. Intermediate result after 3rd iteration, nodes considered in order of descending indices.

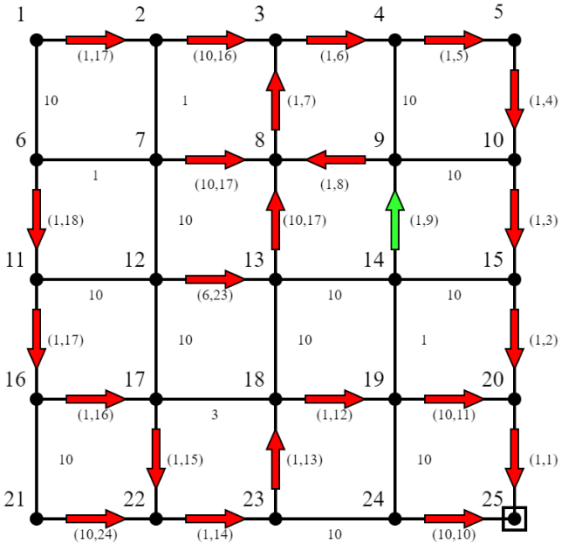


Figure 18. Intermediate result after 4th iteration, nodes considered in order of descending indices.

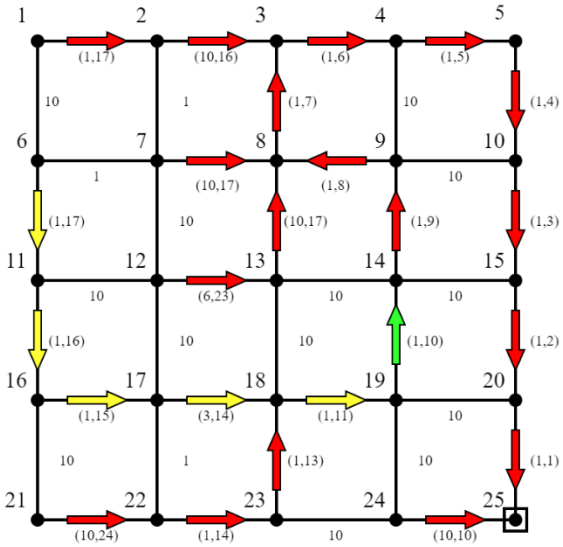


Figure 19. Intermediate result after 5th iteration, nodes considered in order of descending indices.

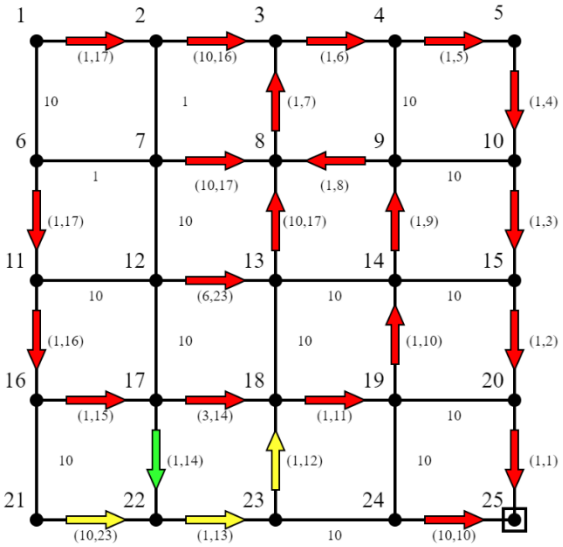


Figure 20. Intermediate result after 6th iteration, nodes considered in order of descending indices.

Figure 20 is the final step in which any nodes in the network have made any connection or cost updates in the optimization process. The simulation needs go through only one more iteration to verify the Bellman Equation has been satisfied at all points, the result of which is Figure 21. For each node, the performance index and the choice of links (Figure 21) is seen to be the same as in the previous iteration (Figure 20); therefore, the algorithm ends and the network has reached an optimized solution where all nodes satisfy the Bellman Equation.

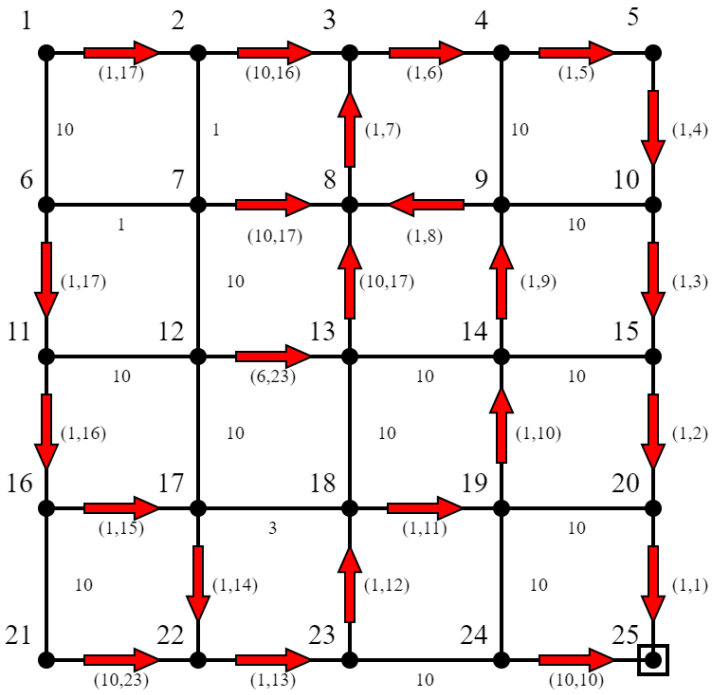


Figure 21. Convergence Confirmation, nodes considered in order of descending indices.

Since the simulation considers nodes closest to the gateway first, we can see in Figure 15 to Figure 21, that information propagates relatively quickly through paths to nodes farther away from the gateway. If the query order of the simulated nodes is reversed, it can be observed that information propagates more slowly between nodes. Instead of simulating queries from node 25 to node 1, Figure 22 through Figure 25 show the results of the simulating queries starting at node 1 and ending at node 25 with node 25 acting as a gateway.

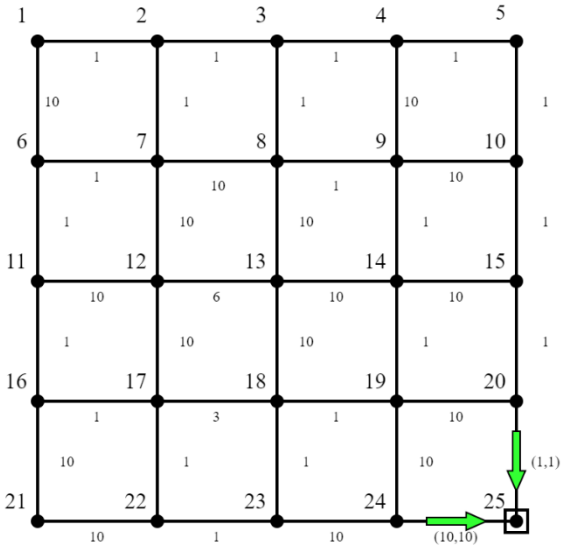


Figure 22. Intermediate result after 1st iteration, nodes considered in order of ascending indices.

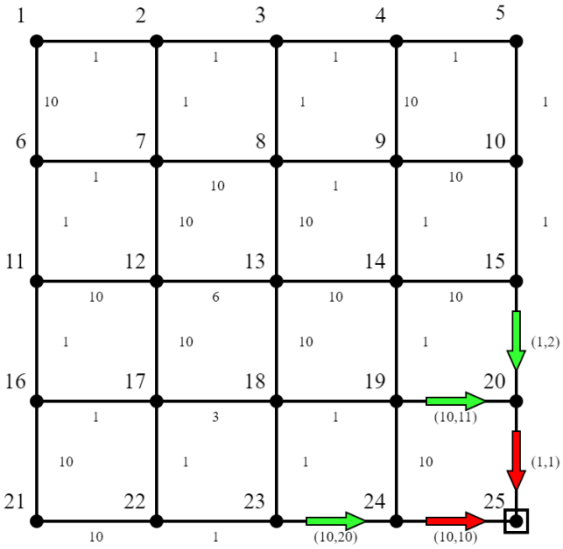


Figure 23. Intermediate result after 2nd iteration, nodes considered in order of ascending indices.

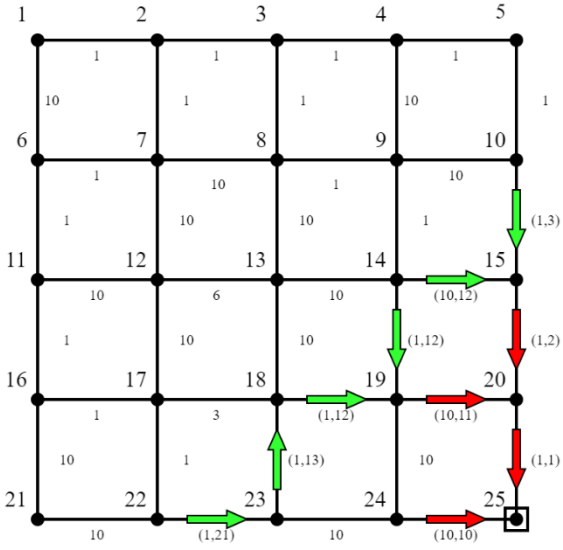


Figure 24. Intermediate result after 3rd iteration, nodes considered in order of ascending indices.

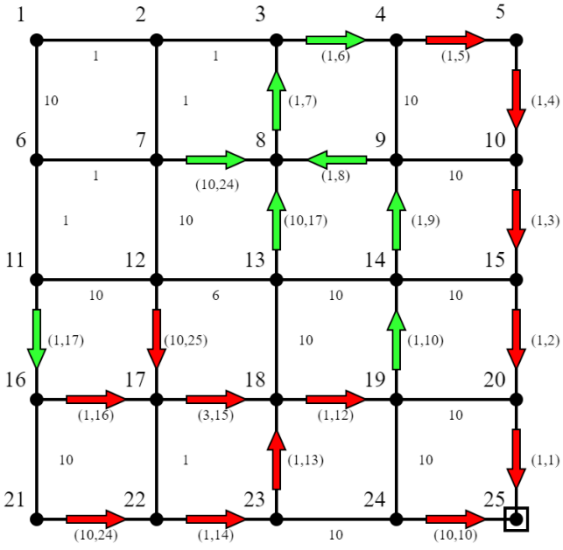


Figure 25. Intermediate result after 6th iteration, nodes considered in order of ascending indices.

It can be seen that the formation of the network slows down with the new query order. Figure 20 shows that with a descending query order, the network has converged to a solution on its 6th iteration and needs just one more iteration to be performed to verify the solution. By comparison, Figure 25 shows that after the 6th iteration with an ascending query order, the network does not even have all nodes connected to a gateway yet. While the simulation shows that it takes longer for the network to converge, the optimization is effective and the resulting costs to the gateway are the same regardless of the node query order and can be seen by comparing Figure 21 and Figure 26.

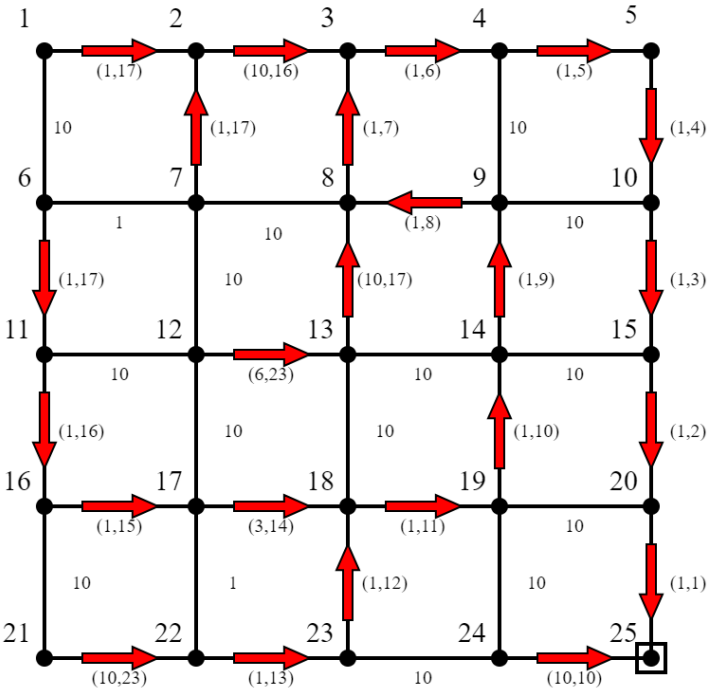


Figure 26. Convergence Confirmation, nodes considered in order of ascending indices.

To further verify the simulation accurately models the proposed routing algorithm, a new node is introduced to an already optimized network to simulate re-optimization. Figure 27 shows an optimal solution before node 19 is introduced to the

network. After node 19 queries and connects to the network, node 18 overhears a competitive cost to the gateway, queries to see if it can establish a more efficient gateway connection, finding that connecting to node 19 results in an optimal path. In turn, nodes 17, 16, 11, and 6 discover that their optimal paths branch off from node 18 and thereby node 19 and can be seen reconfiguring in Figure 28. Another iteration of the simulation shows that node 22 and 23 can connect to node 18 to reduce their cost to the gateway as shown in Figure 29. The final iteration results in Figure 30 showing that the simulation has converged and produced a solution where nodes are considered in order of ascending indices. When compared to the optimal network solution in Figure 26, Figure 30 is identical, verifying that when a new node is introduced to the network, an optimal solution will be reached.

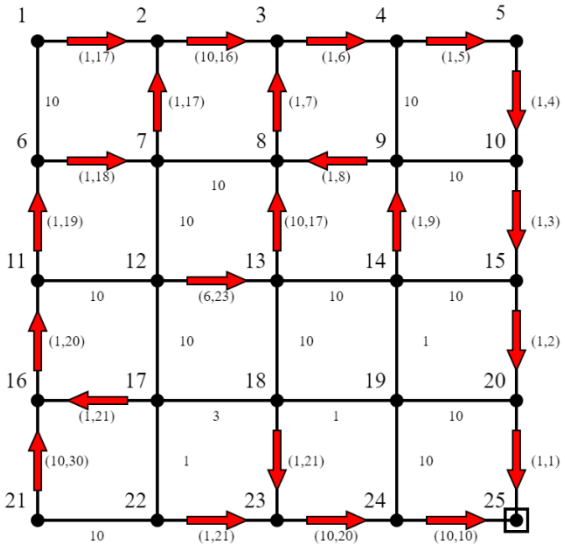


Figure 27. Optimized Solution, Node 19 Disconnected.

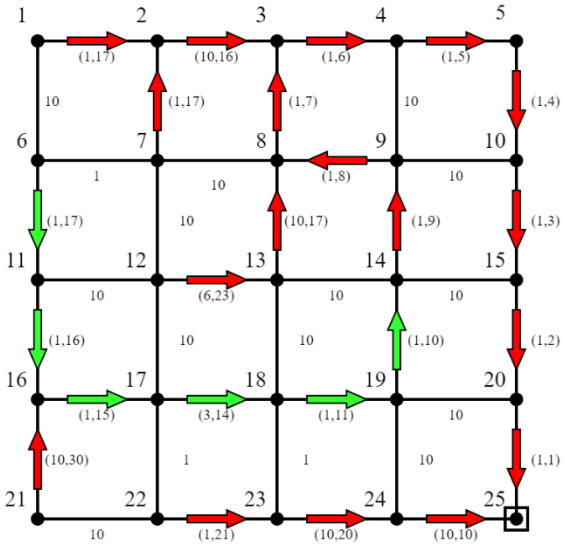


Figure 28. Node 19 Connected, 1st Iteration.

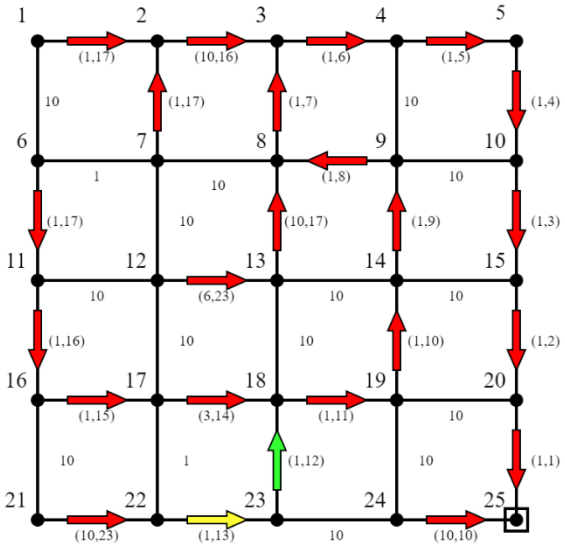


Figure 29. Node 19 Connected 2nd Iteration.

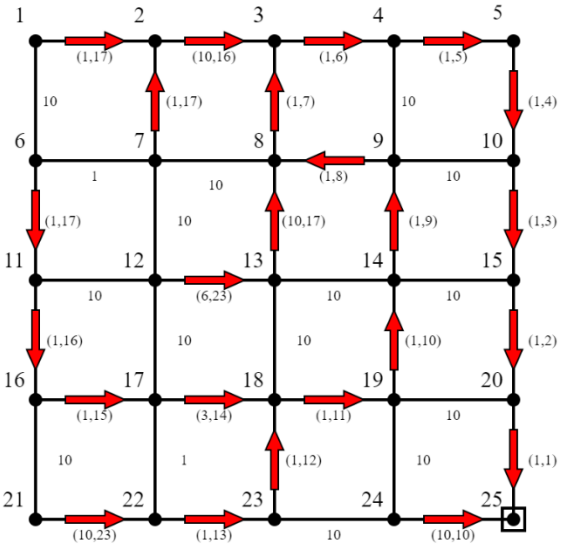


Figure 30. Node 19 Connected, Optimal Solution.

CHAPTER V

CONCLUSION

A decentralized framework for network optimization for wireless sensing nodes is presented. The wireless sensor network is modeled and optimized using an application of the Bellman Equation and Bellman's Principle of Optimality in the form of a dynamic programming algorithm. The dynamic programming algorithm allows for the wireless sensing nodes within the network to make decisions based on locally available information, creating a decentralized routing algorithm. Sample networks are solved both by hand and by the developed algorithm, to prove the validity of the proposed decentralized dynamic programming algorithm. Our observations indicate simulated networks will converge to an optimal configuration. The rate of convergence of the proposed network structure is dependent on the order in which the nodes join the network and the number of gateways nodes.

Future work on the decentralized dynamic routing protocol could include the simulation of networks in which link costs between node pairs are not the same in both directions. The introduction of variable communication costs could better reflect how the network would respond to real world interferences such as weather or aging circuitry. Randomization could be introduced to acquire a more accurate model of how a network would optimize. In a real application, nodes will not query for a connection in any order.

The simulation could also be refined to create a much more accurate model of each node in the network by adding a model for the physical parameters of the sensing node, such as the antenna and received signal strength indication.

BIBLIOGRAPHY

[1] A. A. Ali, S. A. Najafi, O. Boler, Y. Sozer and J. A. DeAbreu-Garcia, "Magnetic Field Energy Harvester and Management Algorithm for Power Tower Sensors," *IEEE Energy ConversionCongress and Exposition,*, pp. 3653-3657, Portland, OR, USA 23-27 Sep. 2018.

[2] S. R. Samireddy, J. Carletta and K. Lee, "An Embeddable Algorithm for Gunshot Detection," *2017 IEEE 60th Internation Midwest Symposium on Circuits and Systems (MWSCAS)*, pp. 68-71, Boston, MA, USA, 6-9 Aug. 2017.

[3] T. Latif, N. Ida, K. Lee and J. Carletta, "A Current and Vibration Based Detection System for Lightning Strikes on Transmission Towers," *2017 IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS)*, pp. 1272-1275, Boston, MA, USA, 6-9 Aug. 2017.

[4] K. Nam, M. A. Rahman, J. A. DeAbreu-Garcia, R. Veillette, M. French, Y. Sozer and J. Lauletta, "Identifying Deteriorated or Contaminated Power System Components from RF Emissions," *2019 IEEE Applied Power Electronics Conference and Exposition (APEC)*, pp. 2015-2020, Anaheim, CA, USA, 17-21 March 2019.

[5] A. E. Bryson Jr., Dynamic Optimization, Menlo Park, California: Addison Wesley Longman, Inc., 1999.

[6] M. Eslami, O. Karimi and T. Khodadadi, "A Survey on Wireless Mesh Networks: Architecture, Specifications and Challenges," in *IEEE 5th Control and System Graduate Research Colloquium*, Shah Alam, Malaysia, 2014.

[7] S. Sharmila and T. Shanthi, "A Survey on Wireless Ad-Hoc Network: Issues and Implementation," in *2016 International Conference on Emerging Trends in Engineering, Technology and Science (ICETETS)*, Pudukkottai, India, Oct. 2016.

[8] K. Atefi, A. H. Shahin, S. Yahya and A. Erfanian, "Performance Evaluation of RIP and EIGRP Routing Protocols in IEEE 802.3u Standard," in *International Conference On Computer And Information Sciences*, 2016.

[9] M. P. Clark, Data Networks, IIPand the Internet: Protocols, Design and Operation, Chichester, England: John Wiley & Sons, 2003.

[10] M. Sniedovich, "Dijkstra's algorithm revisited: the dynamic programming connexion," *Control and Cybernetics*, Vols. Vol. 35, no 3, pp. 599-620, 2006.

[11] Z. Lingling and Z. Juan, "Comparison Study of Three Shortest Path Algorithm," in *International Conference on Computer Technology, Electronics and Communication (ICCTEC)*, Dalian, China, 2017.

[12] F. Lewis, D. Vrabie and V. Syrmos, Optimal Control Third Edition, Hoboken, New Jersey: John Wiley & Sons, Inc, 2012.

[13] N. Biggs, Algebraic Graph Theory Second Edition, Cambridge: Press Syndicate of the University of Cambridge, 1993.

APPENDIX A

MATLAB SCRIPT

```
       %%Decentralized System Solution
clc;clear all; close all;
I = Inf;
global Adj_Matrix

%%~~~~~~~~~~~~~~~~~~Inital Setup of Adjacency Matrix~~~~~~~~~~~~~~~~~~~~~~

Adj_Matrix=...
  [ 0 1 0  0 0 10 0 0  0  0  0 0  0  0  0  0 0  0  0  0  0  0 0  0;
    0 0 10 0 0 0  1 0  0  0  0 0  0  0  0  0 0  0  0  0  0  0 0  0;
    0 0 0  1 0 0  0 1  0  0  0 0  0  0  0  0 0  0  0  0  0  0 0  0;
    0 0 0  0 1 0  0 0  10 0  0 0  0  0  0  0 0  0  0  0  0  0 0  0;
    0 0 0  0 0 0  0 0  0  1  0 0  0  0  0  0 0  0  0  0  0  0 0  0;
    0 0 0  0 0 0  1 0  0  0  1 0  0  0  0  0 0  0  0  0  0  0 0  0;
    0 0 0  0 0 0  0 10 0  0  0 10 0  0  0  0 0  0  0  0  0  0 0  0;
    0 0 0  0 0 0  0 0  1  0  0 0  10 0  0  0 0  0  0  0  0  0 0  0;
    0 0 0  0 0 0  0 0  0  10 0 0  0  1  0  0 0  0  0  0  0  0 0  0;
    0 0 0  0 0 0  0 0  0  0  0 0  0  0  1  0 0  0  0  0  0  0 0  0;
    0 0 0  0 0 0  0 0  0  0  0 10 0  0  0  1 0  0  0  0  0  0 0  0;
    0 0 0  0 0 0  0 0  0  0  0 0  6  0  0  0 10 0  0  0  0  0 0  0;
    0 0 0  0 0 0  0 0  0  0  0 0  0  10 0  0 0  10 0  0  0  0 0  0;
    0 0 0  0 0 0  0 0  0  0  0 0  0  0  10 0 0  0  1  0  0  0 0  0;
    0 0 0  0 0 0  0 0  0  0  0 0  0  0  0  0 0  0  0  1  0  0 0  0;
    0 0 0  0 0 0  0 0  0  0  0 0  0  0  0  1 0  0  0  0  10 0 0  0;
    0 0 0  0 0 0  0 0  0  0  0 0  0  0  0  0 3  0  0  0  1  0 0  0;
    0 0 0  0 0 0  0 0  0  0  0 0  0  0  0  0 0  1  0  0  0  1 0  0;
    0 0 0  0 0 0  0 0  0  0  0 0  0  0  0  0 0  0  10 0  0  0 10 0;
    0 0 0  0 0 0  0 0  0  0  0 0  0  0  0  0 0  0  0  0  0  0 0  1;
    0 0 0  0 0 0  0 0  0  0  0 0  0  0  0  0 0  0  0  0  10 0 0  0;
    0 0 0  0 0 0  0 0  0  0  0 0  0  0  0  0 0  0  0  0  0  1 0  0;
    0 0 0  0 0 0  0 0  0  0  0 0  0  0  0  0 0  0  0  0  0  0 10 0;
    0 0 0  0 0 0  0 0  0  0  0 0  0  0  0  0 0  0  0  0  0  0 0  10;
    0 0 0  0 0 0  0 0  0  0  0 0  0  0  0  0 0  0  0  0  0  0 0  0;];
global Max;
Max = size(Adj_Matrix,2);
for X = 1:Max
    for Y = X:Max
        if(Adj_Matrix(X,Y)== 0)
```

```
            Adj_Matrix(X,Y) = I;
        end
%         if(X==Y)
%             Adj_Matrix(X,Y) = 0;
%         end
    end
end


%Create the actual Adjacency Matrix
Adj_Matrix = Adj_Matrix+Adj_Matrix'

%%~~~~~~~~~~~~~~~~~~~Variable Initilizations~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
global Gateway_Node;

Gateway_Node = [1,25];% access poinsts are 25 and/or 1 depending on example
J_0 = ones(1,Max)*I;
u_0 = ones(1,Max)*I;
GW_Connect_Status = zeros(1,Max);
Beacon = ones(1,Max);   % Set to 1, new nodes will always beacon to join
                        % Will change to 0 when first connected to GW
                        % Nodes with Beacon will change to 1 when
                        % compeitive data from neighobrs is visible
for n = 1:size(Gateway_Node,2)
    GW_Connect_Status(Gateway_Node(n)) = 1;
    J_0(Gateway_Node(n)) = 0;
    u_0(Gateway_Node(n)) = 0;
    Beacon(Gateway_Node(n)) = 0;
end

global NodeData;
%
%  NodeData = [#of nodes in the network; Gateway Connection Status;
%               Performance Index; Control Vector; Beacon Indication]
NodeData = [1:Max;GW_Connect_Status;J_0;u_0;Beacon];
global N;
N=1;
Stable = 0;
J = zeros(1,Max);

%%~~~~~~~~~~~~~~~~~~~Optimization Algorithm~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

while(Stable ~= 1)

    for x = 1:Max       %1 to 25
%    for x = Max:-1:1    %25 to 1
        skip = 0;
        for s = 1:size(Gateway_Node,2)
            if(x==Gateway_Node(s))
                skip = 1;
            end
        end
```

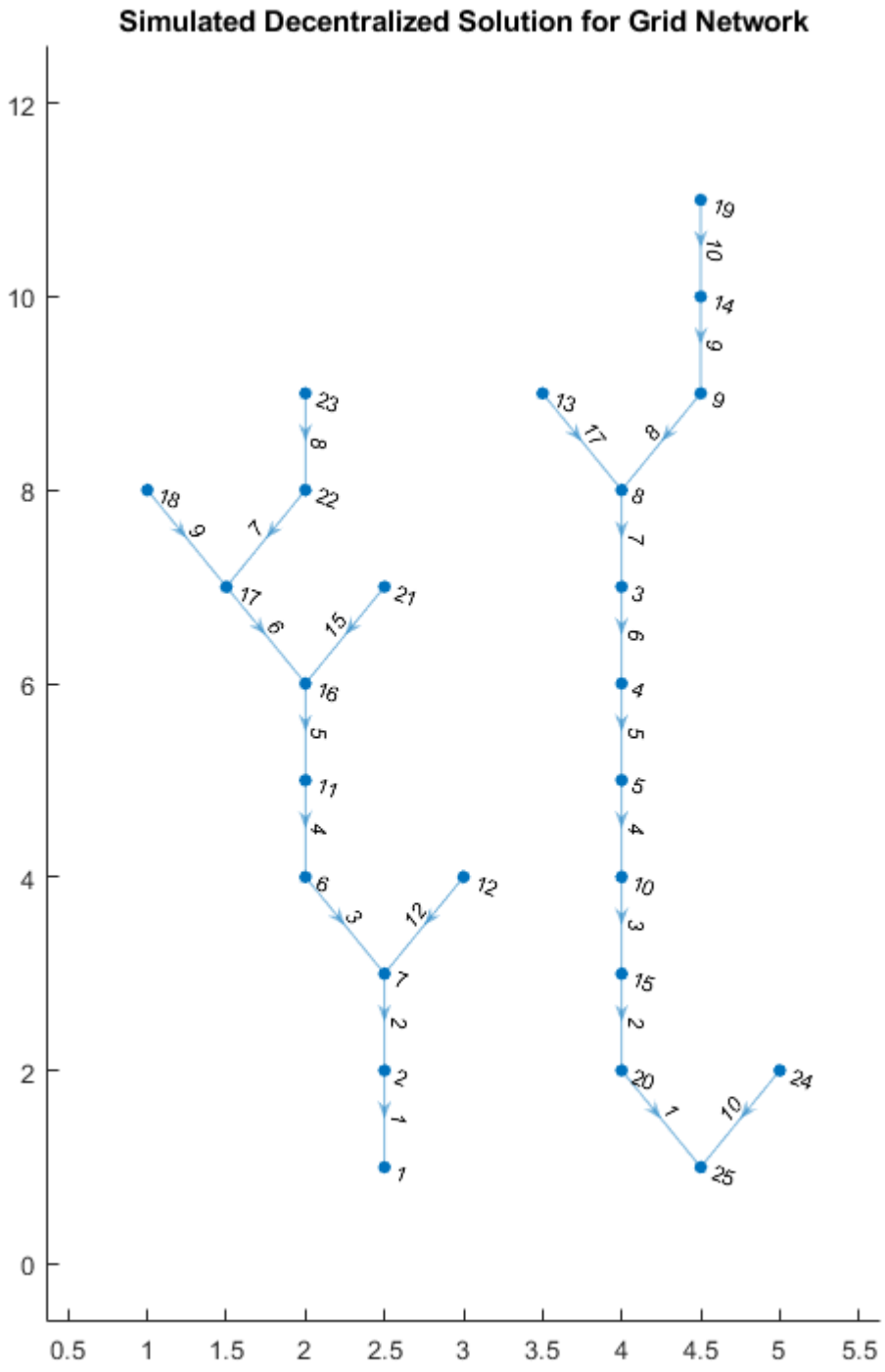
```
        if((skip~=1)&&(NodeData(5,x)==1))
            SensorNode(x);
        end
    end
    N=N+1;
    J(N-1,:) = NodeData(3,:);
    if(N>2)
        Nearby_Observers(J,N);
        Stable = isequal(J(N-2,:),NodeData(3,:));
    end
end
print_figure();


%%~~~~~~~~~~~~~~~~~~~~~~~~~Plotting~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
function print_figure()
    global Max;
    global NodeData;
    Mat = zeros(Max,Max);
    for n = 1:Max
        if((NodeData(4,n)~=0) & (NodeData(4,n)<Inf))
            Mat(n,NodeData(4,n)) = NodeData(3,n);
        end
    end

    %setting figure window size

    figure()
    set(gcf,'position',[10,10,550,800]);
    %title(['Decentralized Solution  Iterations =',num2str(N-2)]),hold on
    title(['Simulated Decentralized Solution for Grid Network']),hold on
    A = digraph(Mat);
    AA = plot(A);
    labeledge(AA,1:numedges(A),A.Edges.Weight)
end
%%~~~~~~~Function Simulating Decentralized Sensor Node Decision Making~~~~~~~
%%~~Looks at connected nodes and asks for data from the

function SensorNode(node,N)
    global Adj_Matrix;
    global NodeData;
    global Max;
    global N;
    i = 1;
    for n = 1:Max                       %Format of info = [
        if(Adj_Matrix(node,n)~= Inf)    %Requestor;Destination;
            if(node ~= n)               %Gate Connect; Perf Ind. J;link cost]
                info(:,i) = ...
                    [node;n;DataReq(n,node)];
                i = i+1;
            end
        end
```

```
    end
    Establish_GW_Connection = 0;

    for c = size(info,2)
        if (info(3,c)~=0)
            Establish_GW_Connection = 1;
        end
    end
    if(Establish_GW_Connection ~= 0)
        [J,u] = min(info(4,:)+info(5,:));   %Minimization of information
        NodeData(2,node)= 1;                %Gateway Connection Status
        NodeData(3,node)= J;                %Total Cost to Gateway J
        NodeData(4,node)= info(2,u);        %Destination u
        NodeData(5,node)= 0;                %Node has finished Beaconing
    end
end

%%~~~~~~~Function Simulating A Node Receiving an Information Request~~~~~~~
%%~~Sends the requesting node:
%%~~1) Access Point Connection Status
%%~~2) Cost to Access Point
%%~~3) Cost between Destination and Requestor

function DR = DataReq(Destination,Requestor)
    global NodeData;
    global Adj_Matrix;
    DR = [NodeData(2,Destination);NodeData(3,Destination);...
        Adj_Matrix(Requestor,Destination)];
end


% Function that indicates which nodes will beacon during the next iteration
% Serves to satisfy the "Competitive Cost" condition for nodes to begin
% beaconing
function Nearby_Observers(J,N)
    global Adj_Matrix;
    global NodeData;
    global Max;
    global Gateway_Node;

    J_Change = J(N-2,:)-J(N-1,:);
    for x = 1:Max
        skip = 0;
        for s = 1:size(Gateway_Node,2)
            if(x==Gateway_Node(s))
                skip = 1;
            end
        end
        if((skip~=1)&&((J_Change(x)~=0)))
            for observer = 1:Max
                if(Adj_Matrix(observer,x)~= Inf)    %Requestor;Destination;
                    if(observer ~= x)               %Gate Connect; Perf Ind. J;link cost]
```

```
                        NodeData(5,observer)=1;
                        J_Change(x);
                    end
                end
            end
        end
    end
end
    % Neighbor observes datatransmission and decides it needs to beacon
    % next iteration
```

```
Adj_Matrix =

  Columns 1 through 13

   Inf     1   Inf   Inf   Inf    10   Inf   Inf   Inf   Inf   Inf   Inf   Inf
     1   Inf    10   Inf   Inf   Inf     1   Inf   Inf   Inf   Inf   Inf   Inf
   Inf    10   Inf     1   Inf   Inf   Inf     1   Inf   Inf   Inf   Inf   Inf
   Inf   Inf     1   Inf     1   Inf   Inf   Inf    10   Inf   Inf   Inf   Inf
   Inf   Inf   Inf     1   Inf   Inf   Inf   Inf   Inf     1   Inf   Inf   Inf
    10   Inf   Inf   Inf   Inf   Inf     1   Inf   Inf   Inf     1   Inf   Inf
   Inf     1   Inf   Inf   Inf     1   Inf    10   Inf   Inf   Inf    10   Inf
   Inf   Inf     1   Inf   Inf   Inf    10   Inf     1   Inf   Inf   Inf    10
   Inf   Inf   Inf    10   Inf   Inf   Inf     1   Inf    10   Inf   Inf   Inf
   Inf   Inf   Inf   Inf     1   Inf   Inf   Inf    10   Inf   Inf   Inf   Inf
   Inf   Inf   Inf   Inf   Inf     1   Inf   Inf   Inf   Inf   Inf    10   Inf
   Inf   Inf   Inf   Inf   Inf   Inf    10   Inf   Inf   Inf    10   Inf     6
   Inf   Inf   Inf   Inf   Inf   Inf   Inf    10   Inf   Inf   Inf     6   Inf
   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf     1   Inf   Inf   Inf    10
   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf     1   Inf   Inf   Inf
   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf     1   Inf   Inf
   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf    10   Inf
   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf    10
   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf
   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf
   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf
   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf
   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf
   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf
   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf

  Columns 14 through 25

   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf
   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf
   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf
   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf
   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf
   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf
   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf
   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf
```

```
  1   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf
Inf     1   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf
Inf   Inf     1   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf
Inf   Inf   Inf    10   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf
 10   Inf   Inf   Inf    10   Inf   Inf   Inf   Inf   Inf   Inf   Inf
Inf    10   Inf   Inf   Inf     1   Inf   Inf   Inf   Inf   Inf   Inf
 10   Inf   Inf   Inf   Inf   Inf     1   Inf   Inf   Inf   Inf   Inf
Inf   Inf   Inf     1   Inf   Inf   Inf    10   Inf   Inf   Inf   Inf
Inf   Inf     1   Inf     3   Inf   Inf   Inf     1   Inf   Inf   Inf
Inf   Inf   Inf     3   Inf     1   Inf   Inf   Inf     1   Inf   Inf
  1   Inf   Inf   Inf     1   Inf    10   Inf   Inf   Inf    10   Inf
Inf     1   Inf   Inf   Inf    10   Inf   Inf   Inf   Inf   Inf     1
Inf   Inf    10   Inf   Inf   Inf   Inf   Inf    10   Inf   Inf   Inf
Inf   Inf   Inf     1   Inf   Inf   Inf    10   Inf     1   Inf   Inf
Inf   Inf   Inf   Inf     1   Inf   Inf   Inf     1   Inf    10   Inf
Inf   Inf   Inf   Inf   Inf    10   Inf   Inf   Inf    10   Inf    10
Inf   Inf   Inf   Inf   Inf   Inf     1   Inf   Inf   Inf    10   Inf
```



Published with MATLAB® R2019a