Generalized Network Routing Metrics and Algorithms

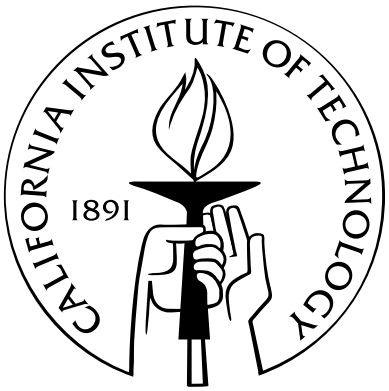
Thesis by

Edwin Soedarmadji

In Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy



California Institute of Technology

Pasadena, California

2008

(Defended May 7, 2008)

To my parents and family

Acknowledgements

I am forever grateful to my advisor Prof. Robert J. McEliece, whose personal involvement and generosity in time and attention have made this thesis possible. I have truly been privileged to have the honor to be one of his students, to learn from and work with one of the best in the field, and in the process, to have the opportunity to share the knowledge I learned from him with others. Prof. McEliece was the reason I became interested in the field of communication. His style of teaching is something I will forever attempt to emulate. His humility and sense of humor can turn any curious student into a passionate convert. Despite all the challenges I have faced during the course of my graduate study, he has been the most sympathetic and understanding.

In addition, my time at Caltech would have not been the same without the camaraderie and friendship that I can always count on from the members of my research group, and the inhabitants of the Posner Laboratory at Caltech. I will cherish the time I spent with my fellow group members Dr. Mostafa El-Khamy and Sarah Sweatlock. I always look forward to the fun weekly group meetings. Finally, I would like to thank Ms. Shirley Slattery for having been extremely helpful throughout my graduate study.

Finally, I would like to recognize and thank my wife and children, whose sacrifice, patience, understanding, and uplifting spirits have been the source of joy and energy that keep propelling me forward in my long-winded academic journey.

Abstract

In this thesis, we introduce generalized network routing metrics that represent probability density parameters of the most popular communication channel models such as (a) the q-ary Symmetric Channel (q-SC) (b) the q-ary Erasure Channel (q-EC); (c) the Gilbert-Elliot Channel (GEC); and (d) the constrained Additive White Gaussian Noise (AWGN). The GEC is a very important for modelling correlated errors in channels such as the ubiquitous TCP/IP links and the wireless fading channels. In this thesis, we prove that channel models (a)–(d) can be used as inputs to the Generalized Dijkstra's Algorithm without resulting in any routing loop.

We also define our own generalized Dijkstra's algorithm that can solve a modified standard shortest path problem that features: (1) a subset of network nodes that are capable of reducing the accumulated path cost down to zero, and (2) a constraint that the cumulative cost of any feasible path must never exceed a prespecified maximum value. We call this modified problem the Gas Station Problem, and its solution the Gas Station Algorithm. The algorithm can be applied in many different areas such as: vehicle routing, project management, and most importantly, network communication.

We investigate various auxilliary synchronization algorithms used in popular routing protocols. Synchronization is used by routers to ensure that all routers operate on an identical routing table — not a trivial task, considering network unreliabilities and possible malicious attacks. Our analysis produces a list of assumptions that guarantees synchronization. We also obtain the upper bounds to quantities such as transmission

period, memory requirement, etc. In turn, these bounds can be used to rate network performance.

Finally, in a related contribution, we analyze message synchronization where a message is retransmitted only if the number of identical messages received exceeds a certain threshold. We define the Chinese Generals Problem as the problem of identifying the set of assumptions under which synchronization is guaranteed. This threshold-base message passing algorithm has the benefits of a tunable gain and a higher noise resistance.

Contents

List of Figures

1

Generalized Network Routing

Metrics and Algorithms

THE history of telecommunication traces back to the days when ancient Persian, Indian, and Chinese postal systems utilized sophisticated messenger systems; to the day when the the first adhesive stamp was first introduced; and centuries later, to the present day in the Internet age where a vast amount of information can be transported around the globe in a fraction of a second.

History has also witnessed an increasingly rapid evolution of the types of information transmitted over telecommunication networks. Not too long ago, people communicate remotely with one another using hand-written and typed letters. At present, digital data packets are used to represent voice, image, music, three-dimensional drawing, etc. Similarly, telecommunication technology has followed a very rapid rate of evolution. Not too long ago, courier networks were replaced by telegraphs, which in turn were replaced by radio and telephone systems. In many applications, these systems are now progressively being replaced by a combination of wired and wireless packet-based networks which are often collectively referred to as the "Internet".

The Internet is an ensemble of networks that are linked together by specialized computer equipments called "routers." These routers are often organized hierarchically in

the same way post offices are organized hierarchically based on their geographical locations. Computers within a group of networks managed under a common administrative authority and control are regarded as an entity known as the Autonomous System (AS). An *interior network* refers to the network contained within a particular AS. In an interior network, communication between computers goes through the so-called *interior routers*. These interior routers implement the Interior Gateway Protocols (IGPs). There are also routers that provide connection between different Autonomous Systems. These routers are called *exterior routers*. They use a different class of algorithms called the Exterior Gateway Protocol (EGP) [7].

Routers play a central role in the Internet by providing each data packet with the instruction on how to proceed from the source to the destination, one router at a time. These instructions are produced by routing algorithms that consider network topology, traffic conditions, and other constraints.

These routing algorithms work by comparing different possible paths from the source to the destination according to their *routing metrics*. A path's routing metric determines whether it is considered by the algorithm as a better (or the best) path compared to other network paths. Routing metrics are assigned not only to paths, but also to edges. In fact, as expected, a path's routing metric depends on the routing metrics of its constituent edges. In practice, routing metrics represent performance measurement quantities such as bandwidth, delay, power, hop count, reliability, economic cost, etc.

Chapter 2 provides an overview of Internet routing and Dijsktra's shortest path algorithm which are used by many routing protocols. This review chapter is the only chapter in this thesis that does not contain our original contribution. Subsequent chapters contain original contributions that, as the title suggests, can be divided into two major categories: (1) generalized network routing metrics, and (2) generalized network routing algorithms.

1.1 GENERALIZED NETWORK ROUTING METRICS

As mentioned above, routing metrics are real-valued quantities. Hence, they can be added and compared with the standard real number operators $+$ and $\leq$. However, many modern communication networks are modeled with graphs whose edges have routing metrics that are probability density parameters of communication channel models. These routing metrics, which we call the *generalized routing metrics*, need to be manipulated with a generalized routing metric algebra, not real number operators. .

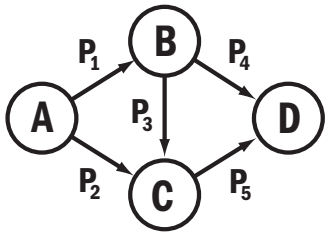


FIGURE 1.1 A simple BEC network

For example, a TCP/IP network can be modeled with a network of Gilbert channels. The routing metric for each edge in such a network is the transition parameters of the Gilbert channel associated with the edge.

For the purpose of illustration, we can assume that the links in the network shown in Figure 1.1 are Binary Erasure Channels (BECs). Let node A be the source node, and node D be the destination node. Each edge e_i has a routing metric representing the edge's BEC error probability p_i.

A packet going through the BEC labeled by AB is lost with a probability of p_1, and through BD with a probability of p_4, etc. We can combine AB and BD into a composite link ABD where packets are lost with a probability p:

$$p = p_1 \oplus p_4 = 1 - (1 - p_1)(1 - p_4). \tag{1.1}$$

Equation (1.1) defines the rule for "adding" two BEC parameters. Two BEC parameters can be compared with the standard $\leq$ operator. These two rules can be used to define

the generalized routing metric algebra for BEC networks. Using these operators, the *Generalized Dijkstra's Algorithm* (GDA) [8] can then be used to find the best path with the least p.

Chapter 3 contains the first set of contributions from this thesis, which is a detailed analysis of the generalized routing metric algebras for (a) the q-ary Symmetric Channel (q-SC) [1, 2]; (b) the q-ary Erasure Channel (q-EC) [3–5]; and (c) the Gilbert-Elliot Channel (GEC) [6], and (d) the constrained Additive White Gaussian Noise (AWGN). The GEC is a very important for modeling correlated errors in channels such as the ubiquitous TCP/IP links and the wireless fading channels. In this thesis, we prove that these channels are compatible with the generalized Dijkstra's algorithm, i.e., the algorithm is guaranteed to produce no routing loop.

1.2 Generalized Network Routing Algorithms

Another set of contributions made by this thesis falls under the category of generalized network routing algorithms. In turn, contributions in this category can be further divided into subcategories where we introduce: (1) enhancements and modifications of the existing routing algorithms, and (2) theoretical analysis on performance bounds and algorithmic correctness of auxilliary synchronization algorithms used in popular routing protocols.

Chapter 4 presents the second contribution of this thesis, which is another type of Generalized Dijkstra's Algorithm that can solve a modified shortest-path problem that features: (1) a subset of network nodes that are capable of reducing the accumulated path cost down to zero, and (2) a constraint that the cumulative cost of any feasible path must never exceed a prespecified maximum value. We call this problem the Gas Station Problem, and its solution the Gas Station Algorithm. The algorithm can be applied in many different areas: transportation network, vehicle routing, project management,

and most importantly, network communication.

In the same chapter, we also present a related contribution of an application of the Gas Station Algorithm to the very important problem of (1) estimating the *worst-case* performance of mission-critical communication networks and (2) finding the worst-case optimal route that is measured in an entirely new class of routing metric called the worst-case routing metric.

Chapter 5 contains another contribution in the category of generalizing network routing algorithms. Previously, we have mentioned that routing metrics are used by routing algorithms to compute the best network path. Inside these routing algorithms, routing metrics are used to build what is called the *routing table* (all these terms are covered in Chapter 2). Each entry in the table consists of a destination node and the path (or the next router) that can take a packet from the current router to that destination. The table is built from the network's topology database. As its name suggests, the topology database encodes the networks' actual topology, which in the case of the Internet, is constantly changing.

Regardless of network changes, all network nodes have to maintain identical copies of the routing table — not a trivial task, considering unreliabilities of network links and nodes, and possible malicious attacks. In other words, the routing tables of all the nodes have to be synchronized at all times. Synchronization is very important because both Open Shortest Path First (OSPF) and Optimized Link State Routing (OLSR) protocols — which are two of the most ubiquitous interior network routing protocols on the Internet — adopt this stringent requirement. The only way to keep these tables synchronized is by using message-passing algorithms that "flood" the entire network with identical copies of table. Thus, the relevant question is: under what assumptions can one guarantee that such synchronization methods will work?

Our contribution is the solution to what we call the Highly Dynamic Network problem. The solution is a set of assumptions that guarantees a proper message synchronization

between a pair of nodes in a network where flooding is used and where connectivity changes are asynchronous. We also obtain the upper bounds to quantities such as transmission period, memory requirement, etc. These bounds can be used to rate network performance.

Still in Chapter 5, we present a related contribution that analyzes message synchronization with a more sophisticated model where a message is retransmitted only if the number of identical messages received exceeds a certain threshold. The Chinese Generals Problem is defined as the problem of identifying the set of assumptions under which synchronization is guaranteed. We prove that the threshold function provides the additional benefits of a tunable gain and a higher noise resistance.

In this chapter, we also discuss our contribution of an application of the threshold-based message passing algorithm into error-correction decoding. Traditionally, Low-Density Parity Check (LDPC) codes have been encoded and decoded using linear algebraic operators in finite fields. In the encoding mode, each parity check node computes the parity bit using the XOR $\oplus$ operator. In the decoding mode, the parity bit is then transmitted to the neighboring variable nodes, each of which applies a selection rule (mostly majority voting) to determine which bit to send to the parity check node, and so on.

Our algorithm uses the same message passing infrastructure as before, but instead of using the XOR $\oplus$ operator, we use a threshold function that is quite similar to the one defined in the Chinese Generals Problem. Of course, the exact definition of this threshold function determines the actual error-correction code. In Chapter 5 we offer an example threshold function that defines a binary code where each codeword corresponds to a unique Sudoku puzzle. A successful decoding of this code is equivalent to obtaining a valid solution to a Sudoku puzzle. This contribution opens up the possibility of applying the threshold function to define other types of nonlinear codes.

1.3 Additional Topics

Chapter 6 concludes this thesis with two contributions that extend the ideas already presented in the previous chapters. Chapter 5 describes the relationship between a generalized LDPC code and a discrete combinatorial structure more popularly known as the Sudoku square. In Chapter 6, we prove that every q-ary MDS code of length n and distance 2 is equivalent to a ternary Latin hypercube of dimension n. Further, we derive a closed-form expression of the total number such structures.

Our final contribution is a demonstration of how message-passing algorithms can be applied in games of perfect information. For our demonstration, we choose the popular game Tic-Tac-Toe. The game has both enough complexity to show the benefits of message-passing, and enough simplicity to keep the presentation manageable. Compared to conventional algorithm implementations that use game tree search, recursive backtracking, and other types of heuristics, our algorithm requires very few operation types and execution steps. Local functions count the number of enemy and friendly pieces in the neighboring cells and compute the decision value using simple `if` statements. We believe that this hierarchical approach can be adapted into network routing problems.

Bibliography

[1] E. Soedarmadji, "Minimum Worst-Case-Erasure QOS Routing," *Proc. of 10th IEEE S'pore ICCS 2006.*

[2] E. Soedarmadji, "Optimal Routing in the Worst-Case-Error Metric," *Proc. of IEEE GLOBECOM '06.*

[3] E. Soedarmadji, R.J. McEliece, "Optimal Worst-Case QoS Routing in Constrained AWGN Channel Network," *Proc. of IEEE ICC '07.*

[4] E. Soedarmadji, "Worst-Case Routing Performance Metrics for Sensor Networks," *Proc. of IEEE PerCom Workshops '07,* pp. 313–317, White Plains, NY, 2007.

[5] E. Soedarmadji, "Worst-Case Routing Performance Metrics for Sensor Networks," , "Worst-Case Routing Performance Evaluation of Sensor Networks," *Proc. of IEEE SAS '07.*

[6] E. Soedarmadji, "Worst-Case Routing Performance Metrics for Sensor Networks," , "Finding the Best QoS Path in a Gilbert Channel Network," *Proc. of the 67th IEEE VTC '08,* Singapore, 2008.

[7] S.A. Thomas, *IP Switching and Routing Essentials*, Wiley, 2001.

[8] J.L. Sobrinho, "Algebra and algorithms for QoS path computation and hop-by-hop routing in the Internet," *ACM ToN.*, 10:541–550, 2002.

2

Standard and Generalized Shortest Path Algorithms in Network Routing

THE diversity of protocols, equipments, and topologies coexisting in the Internet makes network routing a very complex topic. In this thesis, we limit our discussion and analysis only to a specific class of routing algorithms. To select the particular class to focus on, we order our preference based on the algorithm's popularity. Let us now describe the taxonomy of Internet routing.

By its "delivery semantic" (which is a fancy word for addressing scheme), Internet routing can be categorized into unicast, anycast, broadcast, or multicast. This thesis focuses only on *unicast* routing protocols, which are by far the most dominant protocols in today's Internet.

Routing protocols can also be static or dynamic. Static routing protocols assume that the routing table — i.e., the network "address book" and topology — never changes. This type of protocols is common in Public Switched Telephone Networks (PSTNs). In contrast, dynamic routing protocols assume that the network constantly evolves. The Internet obviously belongs in this category. This factor motivates us to focus on *dynamic* unicast routing protocols. In this case, routing tables have to be continuously updated to reflect the network topology and condition.

Next, dynamic unicast Internet routing protocols can be further categorized into interior and exterior protocols. As we discussed before, the Internet can be thought of an interconnected federation of autonomous systems. Within each autonomous system, the so-called Interior Gateway Protocols (IGP) are used to route data packets. Between autonomous systems, the Exterior Gateway Protocols (EGP) — such as the Border Gateway Protocol, or BGP — are used to glue together the networks. Predictably, the population of IGP routers running in autonomous systems vastly outnumbers the population of EGP routers. Therefore, we focus our analysis on IGP routers.

Finally, based on their route discovery mechanisms, we can categorize the Interior Gateway Protocols into different classes. Two of the most important classes are the Link State Routing (LSR) and the Distance Vector Routing (DVR) protocols. LSR protocols are more readily scalable compared to the older DVR protocols. Correspondingly, this thesis focuses on LSR protocols, which are also the dominant Interior Gateway Protocols. LSR protocols include the Open Shortest Path First (OSPF) protocol that is designed for wired networks, the Optimized Link State Routing (OLSR) protocol that is designed for wireless networks, and Intermediate System to Intermediate System (IS-IS) protocol that is designed for Internet Service Providers (ISP).

Both OSPF and OLSR protocols are rich research platforms. These LSR protocols feature two components: (1) a *topology database* algorithm, which is responsible for propagating local changes in the network topology and inferring the topology database therefrom, and (2) a *route calculation* algorithm, which computes the best next hop for each known destination node. A brief overview of the OSPF protocol is given in the first section of this chapter.

At the heart of route calculation is Dijkstra's Algorithm (DA) [1]. Dijkstra's algorithm calculates the solution to a single-source shortest-path problem in a directed graph $G = (V, E)$ with nonnegative edge weights, i.e., $d(u, v) \geq 0$ for each edge $(u, v) \in E$. The nonnegative condition ensures that DA does not produce routing loops. In the second

section of this chapter, we provide a brief review of DA based on a more detailed exposition in [1].

Dijkstra's algorithm adds, compares, and selects real-valued edge weights. In many network problems, adjacent edge weights can be added and compared using real number $+$ and $\leq$ operators. However, in many important network classes, edge weights represent probability density parameters. Hence, adjacent edge weights cannot be added using these standard operators.

Recent work by Sobrinho [2] shows that DA can be generalized by substituting the standard operators with customized operators defined over these probability density parameters. DA is guaranteed to produce no routing loop if these edge weights — also known as routing metrics — obey a certain set of algebraic properties which in this thesis is denoted by **P**. In his paper, Sobrinho shows several applications of what he calls the Generalized Dijkstra's Algorithm (GDA) to solve network optimization problems where customized operators $\oplus$ and $\preceq$ are used in place of the standard operators $+$ and $\leq$.

For example, to solve the network connectivity problem, edge weights are either 0 or 1, the $\oplus$ operator is the min operator, and the $\preceq$ operator is the standard $\geq$ operator. To solve the standard shortest-path problem, the edge weights are taken from the set $\{\mathbb{R}^+ \cup \infty\}$, the $\oplus$ operator is the standard $+$ operator, and the $\preceq$ operator is the standard $\leq$ operator. Although these examples are useful, the paper does not address path optimization problems in networks where edges represent communication channels.

In this thesis, we go beyond abstract examples and propose a set of new routing metrics that are based on popular channel models that include the q-ary erasure channel, the q-ary symmetric channel, the Gilbert channel, and a constrained version of the additive white Gaussian noise channel. The result is a general, and yet practical, method for improving network communication by extending the LSR protocol to optimize path selection over physical layer (PHY) channel parameters. For completeness, we provide a brief review of the GDA in the third section of this chapter.

The topology database calculation algorithm is the second component of LSR protocols. The topology database calculation algorithm disseminates local connectivity observations to the entire network through a cascade of router broadcasts. At each iteration, information is amplified by a factor that depends on the average outdegree of network vertices. Although this method is very effective, it requires proper *gain* control and *error control*. Without gain control, the cascade can paralyze the entire network by consuming too much bandwidth and resources. On the other hand, without error control, a single faulty observation can corrupt the entire network topology database. In Chapter 5, we discuss our method to overcome this problem.

2.1 Open Shortest Path First (OSPF) Basics

Today, most Internet routers run the Open Shortest Path First (OSPF) protocol. Major communication network equipment vendors support this routing protocol because it is based on an open standard. The only competitors to OSPF are RIPv1 and RIPv2, which belong in the DVR protocol category. DVR protocols are found mostly in older routers and networks, and are phased out in larger networks where scalability problems become increasingly severe.

In the next few paragraphs, we provide a brief list of definitions that describe the OSPF terminology. More detailed explanation can be found in basic networking literature published by communication network equipment manufacturers. The following definitions are quoted from [3]:

Link. A link is a network or router interface assigned to any given network. When an interface is added to the OSPF process, it becomes a link.

Neighbor. Neighbors are two or more routers with interfaces on a common network, e.g., two routers connected on a point-to-point serial link.

Adjacency. An adjacency is a relationship between two OSPF routers that permits di-

rect exchange of route updates. Adjacent neighbors must have compatible network type and router configuration. OSPF directly shares routes only with neighbors that have also established adjacencies.

Hello protocol. The OSPF Hello protocol provides dynamic neighbor discovery and maintains neighbor relationships. Hello packets and Link State Advertisements (LSAs) build and maintain the topological database.

Topological database. The topological database contains information from all Link State Advertisement packets that have been received for an area. The router uses the topology database as an input into the Dijkstra algorithm that uses it to compute the shortest path to every network.

Link State Advertisement. A Link State Advertisement (LSA) is an OSPF data packet containing link-state and routing information that is shared among OSPF routers. An OSPF router will exchange LSA packets only with routers to which it has established adjacencies.

OSPF areas. An OSPF area is a grouping of contiguous networks and routers. All routers in the same area have the same topology database. OSPF areas allow hierarchical network organization and greater scalability.

For an open standard that has to be vendor-neutral, OSPF converges very quickly compared to the proprietary, vendor-specific protocols such as EIGRP, which converges slightly faster. The most important features of OSPF are:

- Support for hierarchy based on areas and autonomous systems
- Emphasis on minimal routing update traffic
- Greater scalability
- Unlimited hop count
- Support for multi-vendor deployment (open standard)

OSPF also owes its popularity to the fact that it was the first link-state routing protocol available to network engineers who were upgrading their networks from the more

Characteristic	OSPF	RIPv2 / RIPv1
Type of protocol	Link state	Distance vector
Classless support	Yes	Yes / No
VLSM support	Yes	Yes / No
Auto-summarization	No	Yes
Manual summarization	Yes	No
Discontiguous support	Yes	Yes / No
Propagation trigger	On change	Periodic
Route propagation	Multicast	Multicast / broadcast
Path metric	Bandwidth	Hops
Hop count limit	None	15
Convergence	Fast	Slow
Peer authentication	Yes	Yes / No
Hierarchical network	Yes (using areas)	No (flat only)
Updates	Event triggered	Route table updates
Route computation	Dijkstra	Bellman-Ford

Table 2.1: Comparison between OSPF and DVR protocols

traditional DVR protocols such as RIPv1 and RIPv2. The motivation to upgrade is very obvious from Table 2.1.

There are still many other OSPF features other than the few listed in Table 2.1. These features make OSPF a fast and robust protocol that can scale up to thousands of actively deployed production networks. One of the most important features of OSPF is its support for routing hierarchy, which allows separation of a larger network into smaller routing subnetworks called areas.

In the heart of OSPF is the Shortest Path First (SPF) algorithm, which is based on Dijkstra's algorithm. This algorithm calculates an area shortest path tree and populates the routing table with the best paths to each network in the area. This calculation is done by every router in the area using the topology database as the primary source of information. The result is a tree with the router at the root position, and all other networks occupying the branches and leaves. The routing table is then calculated from this shortest-path tree. The tree contains only networks in the same area as the router. Separate trees are constructed for each area connected to the router.

The algorithm performs route selection primarily based on the path metric toward

the destination network. At each link, the metric used is the cost associated with every outgoing router interface in the tree. The sum of the costs of the outgoing interfaces along the path makes up the cost of the entire path. Unfortunately, the Internet standards only specify interface costs as arbitrary values assigned to the router interfaces. Different vendors implement different rules to compute these arbitrary values, and give different configuration suggestions. For example, according to the HP-UX manual for OSPF [4]:

> [Interface c]ost values are assigned at the discretion of the network or system administrator. While the value is arbitrary, it must be a function of throughput or capacity of the interface: the higher the value, the lower the throughput or capacity. Thus, the interfaces with the highest throughput or capacity must be assigned lower cost values than other interfaces.

Many OSPF routers provide a default — and hence, identical — cost for all outgoing interfaces. Cisco routers use a simple formula $\frac{R}{B}$ where R is a reference bandwidth (10Mbps for most Cisco routers), and B is the *configured* bandwidth of the network interface in question [5, 6]. Using this equation, a 10Mbps Ethernet interface has a default OSPF cost of 10; a 100Mbps Fast Ethernet interface has a cost of 1; and a 64Kbps interface has a cost of 1,563. The problem is that B is hardly a good representation of the actual channel quality, especially in wireless networks.

In this thesis, we propose a method of calculating the interface cost from physical (PHY) layer parameters of the corresponding interface. These parameters are often probability density parameters of the channel models used to model the link between neighboring routers. If these density parameters satisfy the compatibility properties of the generalized Dijkstra's algorithm, they can then be used as more accurate routing metrics.

2.2 DIJKSTRA'S ALGORITHM

In this section, we provide a brief overview of Dijkstra's algorithm based on the material in [1]. In a nutshell, Dijkstra's algorithm computes the solution to the shortest-path problem (SPP). In this problem, we consider a directed graph $G = (V, E)$ with a function $\boldsymbol{m}$ that maps each edge $e_i \in E$ into real-valued edge weight λ_i. Suppose E_π denotes those edges in E that are part of π. The weight of a path π is the sum of the weights of its constituent edges:

$$\boldsymbol{m}(\pi) = \sum_{e \in E_\pi} \boldsymbol{m}(e). \tag{2.1}$$

Let $u \overset{\pi}{\rightsquigarrow} v$ denote that the nodes u and v are connected through a path π. The shortest path weight from u to v is defined by Equation (2.2). A shortest path from u to v is defined as any path π^* such that $\boldsymbol{m}(\pi^*) = \pi^*(u, v)$. A shortest path always contains optimal substructures, which means a shortest path between two nodes always contains other shortest paths within it.

$$\pi^*(u, v) = \begin{cases} \min_\pi \{\boldsymbol{m}(\pi) \mid u \overset{\pi}{\rightsquigarrow} v\} & \{\pi \mid u \overset{\pi}{\rightsquigarrow} v\} \neq \varnothing \\ \infty & \text{otherwise} \end{cases} \tag{2.2}$$

We have to define a special arithmetic rule for ∞. For any $a \in \mathbb{R}$, we define $a + \infty = \infty + a = \infty$. Since a graph might also contain negative-weight cycles, we also define the quantity $-\infty$ such that $a + (-\infty) = (-\infty) + a = -\infty$.

Edge weights are often interpreted as quantities that accumulate linearly along a path such as distance, time, cost, penalty, loss, etc. In most cases, these quantities are non-negative. Therefore, in many optimization problems, the objective is to minimize the shortest path weight.

One might ask if the shortest path weight to a node $v \in V$ can be negative. As long as the graph contains no negative-weight cycles that can be reached from the source node

s, then all path weights from s to v remain well defined. However, if there is a negative-weight cycle that can be reached from s, shortest path weights are not well defined. Dijkstra's algorithm simply assumes that all edge weights are nonnegative. Therefore, under this assumption, negative-weight cycles cannot exist.

Although we have defined the shortest path weight in Equations (2.1) and (2.2), we are more interested in obtaining the shortest path itself. To do this, we define a predecessor $\pi[v]$ for each vertex $v \in V$ that is either another vertex or NIL. If from v we trace back along its chain of predecessors, we will obtain the shortest path from s to v.

From the predecessor values $\pi[v]$, we can build a shortest path tree rooted at s, which is a directed subgraph $G' = (V', E')$, with $V' \subset V$ and $E' \subset E$, such that: (1) V' is the set of all nodes reachable from s in G, (2) G' represents a tree with root s, and (3) for all $v \in V'$, the unique simple path from s to v in G' is a shortest path from s to v in the original graph G.

2.2.1 Relaxation

The main technique used by shortest path algorithms is *relaxation*. Using this method, an upper bound $l[v]$ on the actual shortest path weight of each node v is repeatedly reduced until the upper bound cannot be reduced any further. At this point, the bound is equal to the shortest path weight from s. Before relaxation can be used, the shortest path bounds and predecessors have to be initialized to make $\pi[v] = \text{NIL}$ and $l[v] = 0$ for all $v \in V - \{s\}$.

In short, relaxing an edge (u, v) means testing whether $l[v]$ can be improved by going through u. If this is the case, then $l[v]$ and $\pi[v]$ are updated to point to u with the new path weight. Otherwise, the shortest-path estimate and predecessor field values remain the same.

1: **procedure** RELAX $(u, v, \boldsymbol{m})$

2: **if** $l[v] > l[u] + \boldsymbol{m}(u, v)$ **then**

3: $l[v] \leftarrow l[u] + \boldsymbol{m}(u, v)$

4: $\pi[v] \leftarrow u$

5: **end if**

6: **end procedure**

Despite its name, the term "relaxation" actually tightens the upper bound $l[v]$. The term relaxation refers to a relaxation of the constraint $l[v] \leq l[u] + \boldsymbol{m}(u, v)$. If $l[v] \leq l[u] + \boldsymbol{m}(u, v)$, then the constraint is "relaxed."

2.2.2 Dijkstra's Algorithm

Assume that in G, all edge weights are nonnegative, i.e., for all $(u, v) \in E$, we have $\boldsymbol{m}(u, v) \geq 0$. In such a graph, Dijkstra's algorithm provides us with the optimal path(s) that solve(s) the single-source shortest-path problem. Internally, the algorithm assigns into a set S all the nodes whose final shortest path weights from the source s have already been calculated.

1: **procedure** DIJKSTRA ($G, \boldsymbol{m}, s$)

2: **for all** $v \in V$ **do**

3: $l[v] \leftarrow \infty$

4: $\pi[v] \leftarrow$ NIL

5: **end for**

6: $Q \leftarrow V$

7: $l[s] \leftarrow 0$

8: **while** $Q \neq \varnothing$ **do**

9: $u \leftarrow$ EXTRACT-MIN(Q)

10: $Q \leftarrow Q \backslash u$

11: **for all** node $v \in N(u)$ **do**

12: RELAX$(u, v, \boldsymbol{m})$

13: **end for**

14: **end while**

15: **end procedure**

Mathematically, for all nodes $v \in S$, we have $l[v] = \pi^*(s, v)$. At each iteration, the node $u \in V - S \equiv Q$ that has the minimum shortest-path estimate $l[u]$ is inserted into S (and removed from Q). At the same time, all edges leaving u are relaxed. In the following, $N(v)$ denotes all the neighbors of v in G.

First, the algorithm performs the initialization of $l[v]$ and $\pi[v]$ values. Next, the set S is initialized to the empty set, and conversely the set Q is initialized to contain all the nodes in G. In each iteration of the while loop, a node $u \in Q$ with the minimum shortest path estimate is extracted from $Q = V - S$ and moved into S. Finally, each edge (u, v) leaving u is relaxed. If the shortest path to v can be improved by going through u, the values of $l[v]$ and $\pi[v]$ are updated. Note that the nodes only move from Q to S, and not in the other direction. Therefore, since Q originally contains V nodes, the while loop is guaranteed to iterate exactly V times.

Running Dijkstra's algorithm on a graph $G = (V, E)$ with nonnegative weight function $\boldsymbol{m}$ and source s produces the shortest path weights $l[u] = \pi^*(s, u)$ for all nodes $u \in V$. The resulting predecessor subgraph $G' \subset G$ constructed from by $\pi[u]$ is the shortest path tree rooted at s.

2.2.3 Complexity Analysis of Dijkstra's Algorithm

Next, we discuss the complexity of Dijkstra's algorithm. First, let us assume that the priority queue $Q = V - S$ is a linear array. Implemented this way, each call to EXTRACT-MIN triggers a scan across the array and consumes $O(V)$ units of time. Since there are $|V|$ calls, the total amount of time consumed by EXTRACT-MIN is $O(V^2)$. Next, from the definition of G, the total number of edges in all the adjacency lists is $|E|$. Hence, there must be a total of $|E|$ relaxation steps. Each step consumes $O(1)$ units of time. Therefore, the entire algorithm runs in $O(V^2 + E) = O(V^2)$ units of time.

On the other hand, if G is a sparse graph, it is more efficient to implement Q with a binary heap. The total running time for this implementation is generally given by $O((V+E)\log V)$, or $O(E\log V)$ if all nodes can be reached from the source s. If the Fibonacci heap is used instead of the binary heap, a running time of $O(V\log V+E)$ is possible.

2.3 Generalized Dijkstra's Algorithm

We will now briefly introduce the generalized Dijkstra's algorithm (GDA) [2]. Unlike Dijkstra's algorithm, which is restricted to routing metrics that are real numbers, the GDA can operate on any metric of choice without producing a routing loop as long as the metric obeys a set of algebraic properties which we denote by **P**. This feature is very important for solving optimization problems in communication networks where edge weights represent probability density parameters of channel models that do not obey the standard real number additions and comparisons using the $+$ and $\leq$ operators.

The GDA is practically identical to Dijkstra's algorithm (DA) except for the relaxation step, where the $\oplus$ and $\preceq$ operators act on a metric space $\mathcal{M}$. In DA the relaxation step uses the $+$ and $\leq$ operators acting on $\mathbb{R}$.

1: **procedure** GDA (G, $\boldsymbol{m}$, s)
2: **for all** $v \in V$ **do**
3: $l[v] \leftarrow \infty$
4: $\pi[v] \leftarrow$ NIL
5: **end for**
6: $Q \leftarrow V$
7: $l[s] \leftarrow 0$
8: **while** $Q \neq \varnothing$ **do**
9: $u \leftarrow \textsc{Min}(Q)$
10: $Q \leftarrow Q \backslash u$

11: **for all** node $v \in N(u)$ **do**

12: **if** $l[v] \succ l[u] \oplus \boldsymbol{m}(u,v)$ **then**

13: $l[v] \leftarrow l[u] \oplus \boldsymbol{m}(u,v)$

14: $\pi[v] \leftarrow u$

15: **end if**

16: **end for**

17: **end while**

18: **end procedure**

On line 11, $N(u)$ denotes the set of all nodes adjacent to u. The argument $\boldsymbol{m}$ is the edge weights of G, each of which is an element in $\mathcal{M}$, and $\boldsymbol{m}(u,v)$ is the length of (u,v). Lines 10–12 perform the relaxation step of the GDA. This step depends on the definitions of $\mathcal{M}$, $\oplus$, and $\preceq$. If $(\mathcal{M},\oplus)$ and $\preceq$ satisfy the properties **P** below, then they are *compatible* with the GDA.

P1 is a commutative *monoid*, that is, for $a,b,c \in \mathcal{M}$:

- $\mathcal{M}$ is *closed* under $\oplus$: $a \oplus b \in \mathcal{M}$;
- $\oplus$ is *associative* : $a \oplus (b \oplus c) = (a \oplus b) \oplus c$;
- 0 is the *identity* : $a \oplus 0 = 0 \oplus a = a$;
- $\oplus$ is *commutative* : $a \oplus b = b \oplus a$.

P2 There exists $\infty \in \mathcal{M} \mid a \oplus \infty = \infty \oplus a = \infty$.

P3 $\preceq$ is a *total order* on $\mathcal{M}$, i.e., $\preceq$ is :

- *reflexive*: $a \preceq a$;
- *anti-symmetric*: if $a \preceq b$ and $b \preceq a$ then $a = b$;
- *transitive*: if $a \preceq b$ and $b \preceq c$ then $a \preceq c$;
- *total*: for every $a,b \in \mathcal{M}$ either $a \preceq b$ or $b \preceq a$.

P4 There exists the least element 0 that satisfies $0 \preceq a$.

P5 $a \oplus c \prec b \oplus c$ if $a \prec b$ and $c \in \mathcal{M} - \{\infty\}$.

A complete proof of how the set of compatibility properties **P** guarantees the correct-

ness of the GDA is provided in [2]. The time complexity of the GDA is of course identical to the original Dijkstra's algorithm, which is $O(V^2)$.

2.4 Engineering Applications of Generalized Routing Metrics

Amidst the rise of mass-market digital multimedia and wireless networking technologies, the need for more effective and reliable ways of delivering large data files over the Internet becomes urgent, largely due to the fact that a large segment of the economy is now modeled (and financed) on the assumption that such a delivery is feasible. The underlying engineering problem is twofold: (1) multimedia files are mostly large (and only becoming larger), and (2) wireless media are inherently unreliable.

The engineering solution is the usual pair of: (1) data compression, and (2) error correction. Shannon's fundamental source-channel separation theorem [7] states that source coding (data compression) and channel coding (error protection) can be performed separately, sequentially, *and* optimally.

Indeed, in today's Internet, multimedia distribution follows the sequential process of producing a compressed data stream, breaking down the compressed stream into data packets, fortifying the packets with error-protection, and finally streaming the fortified packets to the receiving node for decoding.

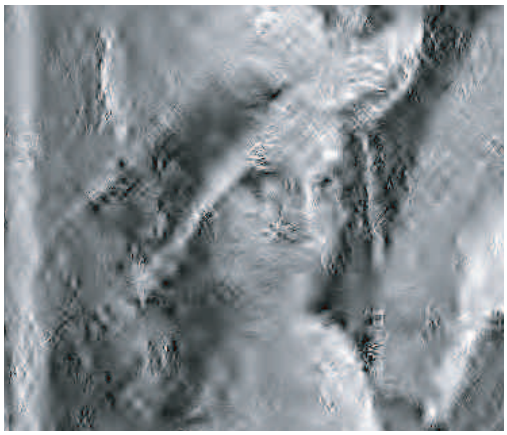


FIGURE 2.1 A corrupted JPEG image

This approach has many problems that are preventing their successful implementations in low-latency, error-resilient video-conferencing systems and future videophones. On a theoretical level, the problem is that Shannon's separation theorem does not apply unless the block lengths are asymptotically long. Thus, sequential processing is not optimal. On a practical level, the big problem stems from the assumption — made by current error protection schemes — that all packets are equally important.

This assumption does not agree with what is observed in practice. In H.323 (H.263) video streams, packets containing macroblocks are extremely important. Losing one such packet wreaks havoc on stream synchronization and forces the decoder to discard several frames worth of data until it can lock into a Picture Start Code (PSC)[2], resulting in a very severe image degradation.

This severe problem persists even in the newest H.264/AVC [10, 11] and *all* other prominent progressive [12] (i.e., multiscale) image and video compression formats such as JPEG, JPEG2000, MPEG-2 (used in DVDs), and MPEG-4 AVC (used in HD-DVDs). Figure 2.1 [13, 14] shows a heavily degraded image decoded from a 1-bit-per-pixel JPEG stream with a mere 0.0001 error rate!

The most promising method uses joint source-channel coding [13–18], in which coding algorithms adapt to the channel parameter of the actual *path* used for data transport. The GDA operating on generalized routing metrics derived from physical layer parameters can be used with existing routing technologies to improve joint source channel coding.

Once the optimal path is computed, the source node can obtain the physical layer parameters from each router in the path using similar techniques used in the *traceroute* program. These parameters can then be fed into the joint source-channel encoder at the source.

[2] This code is a 22-bit pattern `0000-0000-0000-0000-1-000000` [8, 9]

BIBLIOGRAPHY

[1] T.H. Cormen, C.E. Leiserson, R.L. Rivest, *Introduction to Algorithms*, MIT Press, MA, 1990.

[2] J.L. Sobrinho, "Algebra and algorithms for QoS path computation and hop-by-hop routing in the Internet," *IEEE/ACM ToN.*, vol. 10, pp. 541–550, 2002.

[3] T. Lammle, *CCNA: Cisco Certified Network Associate, Study Guide, 6th Edition*, Wiley Publishing Inc., 2007.

[4] Hewlett Packard, Inc., *HP-UX Routing Services: Configuring the OSPF Protocol*, 2007.

[5] Cisco Systems, Inc., *Cisco IOS Release 12.0 Network Protocols Configuration Guide, Pt. 1*, 2004.

[6] Cisco Systems, Inc., *Internetworking Technologies Handbook*, pp. 81; Cisco Press, 2004.

[7] C.E. Shannon, "A Mathematical Theory of Communication," *Bell System Technical Journal*, Vol. 27, pp.379–423, 623–656, 1948.

[8] International Telecommunication Union, "ITU-T H.263 Video coding for low bit rate communication," January 2005.

[9] S. Wenger, G.D. Knorr, J. Ott, F. Kossentini, "Error resilience support in H.263+" *IEEE Trans. on Circ. and Syst. for Video Tech.*, vol.8, no.7, pp.867–877, 1998.

[10] J.H. Zheng, L.P. Chau, "Error-resilient coding of H.264 based on periodic macroblock," *IEEE Trans. on Broadcasting*, vol.52, no.2, pp.223–229; 2006.

[11] S.K. Im, A.J. Pearmain, "Unequal error protection with the H.264 flexible macroblock ordering," *Proc. of the SPIE, VCIP 2005*, vol.5960, pp.1033–1040.

[12] International Telecommunication Union, "Recommendation T.86 – Digital compression and coding of continuous-tone still images (JPEG standard)."

[13] J. Lu, A. Nosratinia, B. Aazhang, "Progressive source-channel coding of images over bursty error channels," *Proc. IEEE ICIP*, Chicago, IL, October 1998.

[14] J. Lu, A. Nosratinia, B. Aazhang, "Progressive source-channel coding of images for feedback channels," *Proc. Data Comp. Conf.*, 1999.

[15] M. Gastpar, M. Vetterli, "Source-Channel Communication in Sensor Networks," *Proc. IEEE IPSN 2003*, LNCS 2634, pp.162–177.

[16] D. Gündüz, E. Erkip, "Joint Source-Channel Codes for MIMO Block-Fading Channels," *IEEE Trans. on Information Theory,* vol.54, no.1, pp.116–134, 2008.

[17] A. Nosratinia, J. Lu, B. Aazhang, "Source-Channel Rate Allocation for Progressive Transmission of Images," IEEE Trans. on Comm., vol.51, no.2, 2003.

[18] A. Hedayat, A. Nosratinia, "Iterative List Decoding of Concatenated Source-Channel Codes," *EURASIP J. on App. Sig. Proc.* Vol.6, pp.954–960; 2005.

[19] E. Soedarmadji, "Minimum Worst-Case-Erasure QOS Routing," *Proc. of 10th IEEE Singapore ICCS 2006.*

[20] E. Soedarmadji, "Optimal Routing in the Worst-Case-Error Metric," *Proc. of IEEE GLOBECOM '06.*

[21] E. Soedarmadji, R.J. McEliece, "Optimal Worst-Case QoS Routing in Constrained AWGN Channel Network," *Proc. of IEEE ICC '07.*

[22] E. Soedarmadji, "Worst-Case Routing Performance Metrics for Sensor Networks," *Proc. of IEEE PerCom Workshops '07.* pp.313–317; White Plains, NY, 2007.

[23] E. Soedarmadji, "Worst-Case Routing Performance Evaluation of Sensor Networks," *Proc. of IEEE SAS '07.*

[24] E. Soedarmadji, "Finding the Best QoS Path in a Gilbert Channel Network," *Proc. of the 67th IEEE VTC '08,* Singapore, 2008.

3

Generalized Dijkstra's Algorithm on Networks of Communication Channels

THE generalized Dijkstra's algorithm (GDA) covered in the previous chapter allows the use of unconventional edge weights, and customized addition and comparison operators that are useful in networks where edges do not necessarily represent real numbers, but rather probability density parameters of communication channel models. For example, Figure 3.1 shows a network whose edges e_i represent q-ary symmetric channels $\boldsymbol{X}^i$ with error probabilities p_i. This chapter describes a customized addition operator with which the probability distribution parameters p_i from adjacent edges can be "added" together.

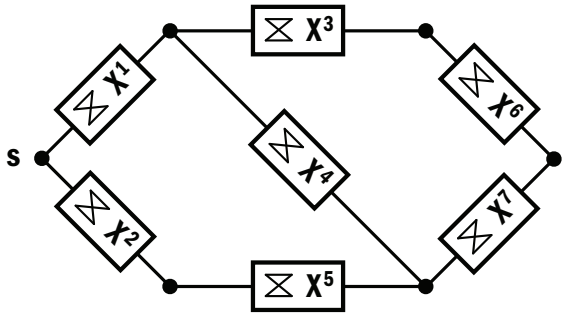


FIGURE 3.1 A network of communication channels

In this chapter, we discuss three different channel models: the q-ary erasure channel, the q-ary symmetric channel, and the Gilbert channel. We organize our analysis of these channels into separate sections. In each section, we begin with a mathematical definition of the channel and continue with a graphical representation of this definition. The mathematical rule for combining two such channels in series and calculating the equivalent parameter is covered next. Finally, each section is concluded with a proof that shows that the channel is compatible to the GDA properties.

3.1 q-ary Erasure Channels

It is beneficial to start our analysis by first considering the simplest type of communication channel, the q-ary erasure channel (q-EC). A q-EC can be used to model channels operating on input and output symbols that are drawn from a common alphabet $\mathcal{Q}$ with $q+1$ letters. At first, the definition and the name sound contradictory. From its name, a q-EC suggests that the channel operates on an alphabet with only q letters. In addition to the standard q letters $0 \ldots q-1$, the alphabet contains an *erasure* symbol ϵ that carries a special meaning.[1]

The q-EC only allows one type of error, namely the one in which the input symbol transmitted is one of the q letters $0 \ldots q-1$ in $\mathcal{Q}$, and the output symbol received is the erasure symbol ϵ.

There are at least two interpretations of the erasure symbol. In the first interpretation, an erasure symbol simply indicates a symbol that is lost during transmission. This interpretation mainly highlights the undeterministic nature of communication channels, i.e., channels unpredictably throw away symbols and replace them with erasure symbols.

[1] Note that our definition for the q-EC differs slightly from the standard definition in which the input and output symbols are not drawn from a common alphabet, but rather from two different alphabets: an input alphabet that contains the q standard letters, and an output alphabet with the same q letters plus the additional erasure symbol ϵ.

In another interpretation, an erasure symbol is itself a "flag" from the channel that an error has occured during transmission. While this interpretation still assumes that the channel is unpredictable, it also implies that the channel can intelligently *detect* any unrecoverable error and *report* any such error by substituting the erroneous symbol with the erasure symbol.

Let us now define the q-EC mathematically. Consider a q-EC which we shall denote by $\boldsymbol{X}$. The random variables corresponding to the input and output symbols of $\boldsymbol{X}$ are denoted by X and X', respectively. Suppose that i and j are two letters in a q-ary alphabet $\mathcal{Q}$, and that $X = i$ and $X' = j$. The following is the mathematical definition of the q-EC.

$$P(X' = j \mid X = i) = \begin{cases} 1-p\,, & i = j \neq \epsilon \\ p \quad\;\; , & i \neq j = \epsilon \\ 1 \quad\;\; , & i = j = \epsilon \end{cases} \tag{3.1}$$

With probability $1-p$, the output symbol j is equal to i. Otherwise, with *erasure probability* $p \in \Lambda \equiv [0,1]$, the output symbol j is the erasure symbol ϵ in $\mathcal{Q}$. The equation explicitly treats the erasure symbol itself as a valid input symbol. There is a reason for this. In the literature, a q-EC is always analyzed in isolation from other q-ECs. In contrast, we are interested in the series combination of q-ECs, which requires the input and output symbol alphabets to be identical.

For example, consider a q-EC with $q = 5$ and $p = 0.05$ represented by the following 6×6 matrix $\boldsymbol{X}$. The row index $i \in \{0 \ldots 5\}$ corresponds to the input symbol $X \in \mathcal{Q} = \{0 \ldots 4, \epsilon\}$, and the column index $j \in \mathcal{Q}$ corresponds to the output symbols $X' \in \mathcal{Q}$. Each entry $\boldsymbol{X}_{ij}$ represents the probability $P(X' = j \mid X = i)$. Note the difference in the last row

for $X = \epsilon$.

$$\begin{matrix} 0.95 & 0.00 & 0.00 & 0.00 & 0.00 & 0.05 \\ 0.00 & 0.95 & 0.00 & 0.00 & 0.00 & 0.05 \\ 0.00 & 0.00 & 0.95 & 0.00 & 0.00 & 0.05 \\ 0.00 & 0.00 & 0.00 & 0.95 & 0.00 & 0.05 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.95 & 0.05 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 1.00 \end{matrix}$$

The input and output symbols of a q-EC are often shown as two columns of $q+1$ dots each. From top to bottom, the dots on the left and right columns represent the input and output letters $0 \ldots q-1, \epsilon$ of the alphabet $\mathcal{Q}$.

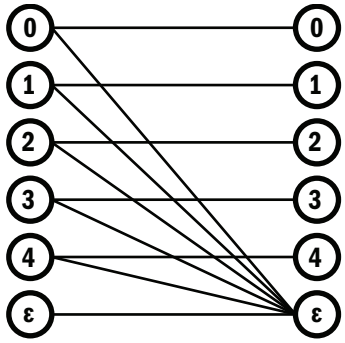


FIGURE 3.2 A 5-ary erasure channel

Symbols on the left are connected to the identical symbols on the right with a set of horizontal lines, as shown in Figure 3.2. These connections correspond to the diagonal entries $\boldsymbol{X}_{ii}$. In addition, the symbols on the left, with the exception of ϵ, are also connected to the symbol ϵ on the right. These connections correspond to last matrix column $\boldsymbol{X}_{iq}$.

We briefly mentioned in the beginning of this section that the reason for including the erasure symbol ϵ in the input alphabet $\mathcal{Q}$ is to allow a series combination of two q-ECs. In general, any two channels $\boldsymbol{Y}^1$ and $\boldsymbol{Y}^2$ — which could be of different types — can be combined in series as long as the output alphabet of $\boldsymbol{Y}^1$ and the input alphabet of $\boldsymbol{Y}^2$ are the same. What is interesting and not immediately clear with the q-ECs is that any series combination of two q-ECs is also a q-EC. In other words, the operation of

combining two q-ECs in series satisfies a kind of closure property.

We shall prove that $\boldsymbol{X}$, the series combination of two q-ECs denoted by $\boldsymbol{X}^1$ and $\boldsymbol{X}^2$, is itself a q-EC by showing that $\boldsymbol{X}$ can be defined by an equation similar to (3.1). This also implies that from the erasure probabilities p_1 and p_2 of $\boldsymbol{X}^1$ and $\boldsymbol{X}^2$, we can compute the equivalent erasure probability p for $\boldsymbol{X}$.

Figure 3.3 shows a series combination of $\boldsymbol{X}^1$ and $\boldsymbol{X}^2$ with $q = 5$. The erasure probabilities p_1 and p_2 of these channels could be different. In the figure, there are three columns of dots: the leftmost column corresponding to the input symbol of $\boldsymbol{X}^1$, the middle column corresponding to either the output symbol of $\boldsymbol{X}^1$ or the input symbol of $\boldsymbol{X}^2$, and the rightmost column corresponding to the output symbol of $\boldsymbol{X}^2$. The random variables corresponding to these three columns are denoted by X, X', and X''.

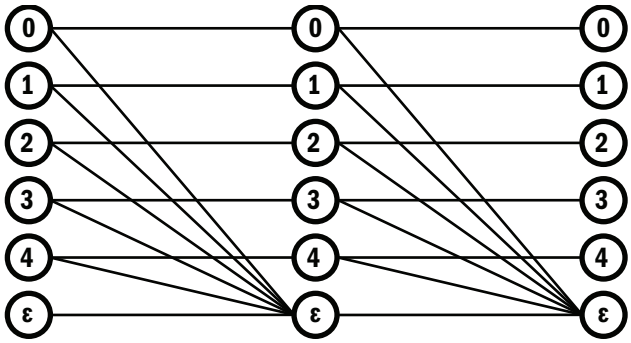


FIGURE 3.3 A series combination of two 5-ECs

From Figure 3.3, we can also see why $\boldsymbol{X}$ is also a q-EC. Remember that in a q-EC, each symbol on the left column is connected to the identical symbol on the right column with a horizontal line. In the figure above, we can also find these horizontal lines that connect identical symbols on the leftmost column and the rightmost column. The difference is that in $\boldsymbol{X}$, one half of the horizontal line is in $\boldsymbol{X}^1$, while the other half is in $\boldsymbol{X}^2$.

In addition, in a q-EC, the dots on the left column, with the exception of ϵ, are also connected to the dot representing ϵ on the right column. Similarly, in Figure 3.3, each dot $X = i$ on the leftmost column in $\boldsymbol{X}$ is also connected to the dot representing ϵ on the rightmost column through two distinct paths: (1) the path going through $X' = i$, and (2)

another one going through $X' = \epsilon$.

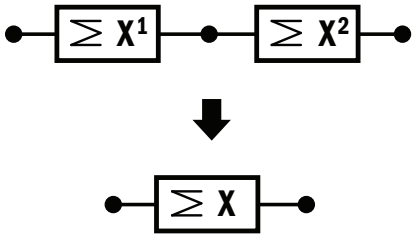


FIGURE 3.4 Series q-ECs and the equivalent q-EC

Although Figures 3.2 and 3.3 offers the clearest depiction of the q-EC and the series combination of two such channels, they are quite cumbersome to reproduce. Figure 3.4 shows a shorthand symbol for the q-EC in which the q input and output letters are drawn as q-ary input and output terminals.

By hiding much of the channel's internal connections, the shorthand symbol looks like a generic system element, which is much easier to use in diagrams containing a large number of interconnected q-ECs.

Now we are ready to prove that an equivalent erasure probability p for $\boldsymbol{X}$ can be computed from the erasure probabilities p_1 and p_2 for $\boldsymbol{X}^1$ and $\boldsymbol{X}^2$, and more importantly, the final expression for p can also be described by an equation similar to (3.1). Rather than deriving p directly, we will first derive $1-p$ in terms of p_1 and p_2, which is much easier to compute.

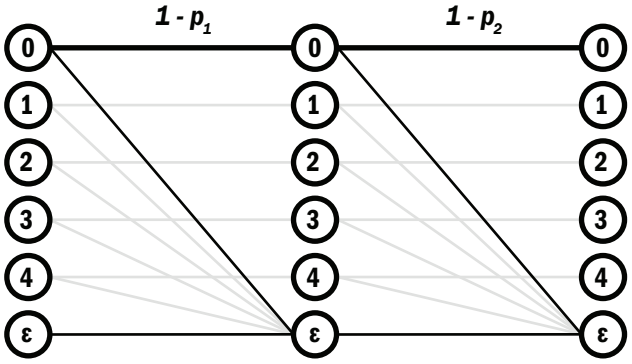


FIGURE 3.5 $P(X'' = 0 \mid X = 0)$

Suppose the input symbol is not the erasure symbol. By its definition, $1-p$ is the

probability of the input and output symbols of $\boldsymbol{X}$ being identical. This probability is simply the probability of both $\boldsymbol{X}^1$ and $\boldsymbol{X}^2$ not producing any error, which is $(1-p_1)(1-p_2)$. For $X=0$, this probability corresponds to the horizontal line on the top of Figure 3.5 above. If the input symbol is the erasure symbol, then by definition $1-p=1$, as shown by the lower horizontal line. Mathematically, the conditional probabilities of X'' given X is given by:

$$P(X''=j \mid X=i) = \begin{cases} (1-p_1)(1-p_2) & , \quad i=j\neq\epsilon \\ 1-(1-p_1)(1-p_2) & , \quad i\neq j=\epsilon \\ 1 & , \quad i=j=\epsilon \quad . \end{cases} \tag{3.2}$$

Equation (3.2) has the exact same form as (3.1). By direct comparison of the two equations, we can infer that for $\boldsymbol{X}$, the erasure probability p:

$$\begin{aligned} p &= 1-(1-p_1)(1-p_2) && (3.3)\\ &= p(p_1,p_2) && (3.4)\\ &\triangleq p_1 \oplus p_2 \quad . && (3.5) \end{aligned}$$

Equation (3.3) expresses the relationship between p, p_1, and p_2 as an algebraic equation that stems directly from the definition of erasure probability for a q-EC. Alternatively, equation (3.4) explicitly expresses p as a function of both p_1 and p_2. Finally, equation (3.5) expresses p as the result of a binary algebraic operation *o-plus* — symbolically denoted by $\oplus$ — on p_1 and p_2.

The formulation of p in terms of the binary operator $\oplus$ is the most relevant to our analysis. The binary operator is a convenient algebraic shorthand for the operation of combining two q-EC in series, hiding much of the algebraic complexity in a similar way that the visual shorthand we introduced earlier in Figure 3.4 hides the internal complexity of a q-EC. These shorthands make the task of analyzing a large communication

system with a large number of interconnected q-ECs much easier. For example, we can conveniently denote the series connection of n different q-ECs by:

$$p = p_1 \oplus p_2 \oplus \cdots \oplus p_{n-1} \oplus p_n \tag{3.6}$$

Naturally, the next question that arises is: does the order of evaluation of $\oplus$ matter? Can the operands be interchangeable? Is there such a thing as the identity element for $\oplus$? These algebraic questions are not only theoretically interesting, but also very important to prove that the q-EC satisfies all the compatibility property **P** of the Generalized Dijkstra's Algorithm (GDA). We will provide a detailed proof of compatibility in the next subsection.

Besides the ability to combine two q-ECs in series, the GDA also includes as its integral feature the ability to compare two q-ECs. For this reason, we also need to define a relational binary operator, which we shall denote by $\preceq$. In general, the semantics of $\preceq$ depends on its context, but we define the statement $A \preceq B$ to always read: "A is no less preferred than B". The following equations show $\preceq$ in two different contexts:

$$\boldsymbol{X}^1 \preceq \boldsymbol{X}^2 \tag{3.7}$$

$$p_1 \preceq p_2 \quad . \tag{3.8}$$

Defined within the context of q-ECs, equation (3.7) states that the q-EC $\boldsymbol{X}^1$ is no less preferred than $\boldsymbol{X}^2$. On the other hand, equation (3.15) states that the erasure probability p_1 of $\boldsymbol{X}^1$ is no worse than p_2 of $\boldsymbol{X}^2$. Fortunately for us, in both examples, the semantics of the $\preceq$ operator can be made consistent. For the erasure probabilities $p_1, p_2 \in \mathbb{R}$, the only logical interpretation for $p_1 \preceq p_2$ is that $p_1 \leq p_2$, where $\leq$ is the standard *less-than-or-equal-to* operator for real numbers. We can then define the $\preceq$ operator unambiguously as follows:

$$\boldsymbol{X}^1 \preceq \boldsymbol{X}^2 \quad \Leftrightarrow \quad p_1 \leq p_2 \quad . \tag{3.8'}$$

Of course, there are other ways to define the $\preceq$ operator. For example, in theory, we could use the q-EC Shannon capacity, which also happens to be given by $1-p$, to rank our preference for different q-ECs.

Algebraic Properties of $\oplus$ and $\preceq$

Previously, we claimed without proof that the operator $\oplus$ satisfies the closure property on Λ, the space of all erasure probabilities. Here, we shall prove not only the closure property, but also all the algebraic properties of $\oplus$, $\preceq$, and Λ in **P** that are required to ensure their compatibility with the GDA.

We begin by proving the properties of $\oplus$. In the following proofs, let Λ be the set of all possible erasure probabilities, which is the range of real numbers $[0,1]$, and let a, b and c be three elements of Λ.

① *Closure.* Λ satisfies closure if $a, b \in \Lambda$ implies $p = a \oplus b \in \Lambda$. Proof:

$$\begin{aligned} & a, b \in [0,1] \\ \Leftrightarrow \quad & (1-a), (1-b) \in [0,1] \\ \Leftrightarrow \quad & (1-a)(1-b) \in [0,1] \\ \Leftrightarrow \quad & 1-(1-a)(1-b) \in [0,1] \end{aligned}$$ (Q.E.D.)

② *Associativity.* The order in which $\oplus$ is evaluated does not matter. Proof:

$$\begin{aligned} (a \oplus b) \oplus c &= (1-(1-a)(1-b)) \oplus c \\ &= 1-(1-(1-(1-a)(1-b)))\,(1-c) \\ &= 1-(1-a)(1-b)(1-c) \\ &= 1-(1-a)\,(1-(1-(1-b)(1-c))) \\ &= 1-(1-a)\,(1-(b \oplus c)) \\ &= a \oplus (b \oplus c) \end{aligned}$$ (Q.E.D.)

③ *Commutativity.* The operand position does not matter. Proof:

$$\begin{aligned} a \oplus b &= 1 - (1-a)(1-b) \\ &= 1 - (1-b)(1-a) \\ &= b \oplus a \end{aligned} \qquad \text{(Q.E.D.)}$$

④ The identity element 0 satisfies $a \oplus 0 = a, \ \forall a \in \Lambda$. Proof:

$$\begin{aligned} a \oplus 0 &= 1 - (1-a)(1-0) \\ &= 1 - (1-a) \\ &= a \end{aligned} \qquad \text{(Q.E.D.)}$$

⑤ The absorptive element $\infty \equiv 1$ satisfies $a \oplus \infty = \infty, \ \forall a \in \Lambda$. Proof:

$$\begin{aligned} a \oplus \infty &= 1 - (1-a)(1-\infty) \\ &= 1 - (1-b)(1-1) \\ &= 1 = \infty \end{aligned} \qquad \text{(Q.E.D.)}$$

⑥ The operator $\preceq$ establishes a total order on Λ, i.e., for every pair of elements $a, b \in \Lambda$, either $a \preceq b$ or $a \preceq b$ (or both, in which case $a = b$). In other words, total order allows any elements to be compared without contraditions. Since for q-ECs the space Λ is just $[0,1] \in \mathbb{R}$, and $\preceq$ is the standard operator $\leq$, this property is automatically satisfied.

Total order property of $\preceq$ is a result of the operator being reflexive (i.e., $a \preceq a$), anti-symmetric (i.e., $a \preceq b$ and $b \preceq a$ implies $a = b$), and transitive (i.e., $a \preceq b$ and $b \preceq c$ implies $a \preceq c$). These properties are automatically satisfied by the standard $\leq$ operator. (Q.E.D.)

⑦ The least element 0 satisfies $0 \preceq a \ \ \forall a \in \Lambda$. The fact that the notation for the least element is the same as the notation for the identity element is not a coincidence. Since the identity element 0 does not change the "magnitude" of any operand it is added to, intuitively, we can think of the identity element as also the smallest element. The proof simply uses the fact that when $\preceq$ is simply the standard operator $\leq$, the real number 0 is the least element in Λ. (Q.E.D.)

⑧ *Strict isotonicity.*

$$a < b$$

$$(1-a) > (1-b)$$

$$(1-a)(1-c) > (1-b)(1-c)$$

$$1-(1-a)(1-c) < 1-(1-b)(1-c)$$

$$a \oplus c < b \oplus c \qquad \text{(Q.E.D.)}$$

3.2 q-ary Symmetric Channels

Next, we discuss the q-ary symmetric channel (q-SC). Unlike the q-EC, the q-SC does not have the ability to produce the erasure symbol. In other words, the channel's alphabet, which we shall denote by $\mathcal{Q}$, does not include the erasure symbol. As previously mentioned, the erasure symbol is essentially an explicit hint from the channel that an error has occured. Without an erasure symbol, the q-SC cannot provide the additional information of whether or not an error actually occurs. As a result, without knowing the actual input symbol transmitted, each output symbol looks perfectly fine.

This fact alone makes the q-SC a more challenging channel to protect against errors. In the case of the q-EC, the receiver can be sure that if the channel does not replace an input symbol with the erasure symbol, then the output symbol is guaran-

teed to be correct. In the case of the q-SC, however, there is always some probability that a seemingly legitimate output symbol produced by the channel is actually a symbol produced in error. In the following paragraphs, we describe how channel error is defined in a q-SC.

First, let us consider a q-SC denoted by $\boldsymbol{X}$. Denote the random variables corresponding to the input and output symbols of $\boldsymbol{X}$ by X and X', respectively. Suppose $X = i$ and $X' = j$, where i and j are letters taken from the q-ary alphabet $\mathcal{Q}$. With probability $1-p$, the output symbol j is equal to i. Otherwise, with *error probability* $p \in \Lambda \equiv [0, \frac{q-1}{q}]$, the output symbol j is one of the $q-1$ letters in $\mathcal{Q}$ that are different from i. It is reasonable to assign an equal probability to each of these $q-1$ letters, or mathematically:

$$P(X' = j \mid X = i) = \begin{cases} 1-p & , i = j \\ p/(q-1) & , i \neq j \end{cases} .$$

The channel is "symmetric" because the above expression applies equally to all input symbols i in $\mathcal{Q}$. In addition, for each i, the probabilities of $i = j$ are equal for all $j \in \mathcal{Q}$. The symmetry also becomes apparent when the probability values are presented in a matrix format.

$$\begin{array}{cccccc} 0.95 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 \\ 0.01 & 0.95 & 0.01 & 0.01 & 0.01 & 0.01 \\ 0.01 & 0.01 & 0.95 & 0.01 & 0.01 & 0.01 \\ 0.01 & 0.01 & 0.01 & 0.95 & 0.01 & 0.01 \\ 0.01 & 0.01 & 0.01 & 0.01 & 0.95 & 0.01 \\ 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.95 \end{array}$$

For example, consider a particular q-SC with $q = 6$ and $p = 0.05$ represented by the above 6×6 matrix $\boldsymbol{X}$. Compare this q-SC matrix with the q-EC we discussed before. The row index $i \in \{0 \ldots 5\}$ corresponds to the input symbol $X \in \mathcal{Q} = \{0 \ldots 5\}$, and the column index $j \in \mathcal{Q}$ corresponds to the output symbols $X' \in \mathcal{Q}$. Each entry $\boldsymbol{X}_{ij}$ represents

the probability $P(X' = j \mid X = i)$.

Visually, it is customary to depict the q-SC with two columns of dots drawn in such a way that each dot on the left column is connected to every dot on the right column (and vice versa), as shown in Figure 3.6. From top to bottom, each dot on the left and right columns of dots represent the input and output letters $0 \ldots q-1$ from the alphabet $\mathcal{Q}$. The horizontal lines connecting identical input and output letters correspond to the diagonal entries $\boldsymbol{X}_{ii}$.

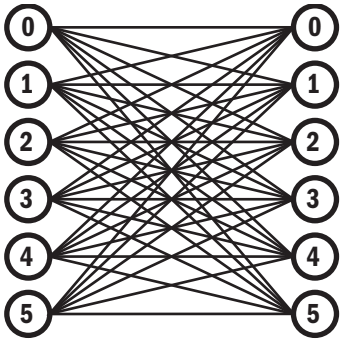


FIGURE 3.6 A 6-ary symmetric channel

We note that any two q-SCs can also be combined in series into a single q-SC. Let us denote two such q-SCs by $\boldsymbol{X}^1$ and $\boldsymbol{X}^2$, and their error probabilities by p_1 and p_2, respectively. To avoid unnecessary clutter in visualizing this arrangement, we introduce a shorthand symbol for the q-SC that combines the q input and output dots into q-ary input and output terminals shown in Figure 3.7 below and hides the connecting lines inside the boxes.

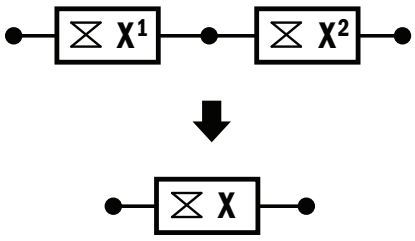


FIGURE 3.7 Series q-SCs and the equivalent q-SC

In Figure 3.7, the first terminal represents the input letters from $\mathcal{Q}$ of $\boldsymbol{X}^1$. The second

terminal represents both the output letters of $\boldsymbol{X}^1$ and the input letters of $\boldsymbol{X}^2$. Finally, the third terminal represents the output letters of $\boldsymbol{X}^2$.

Just like in the case for q-ECs, the series combination of $\boldsymbol{X}^1$ and $\boldsymbol{X}^2$ forms a channel that we shall denote by $\boldsymbol{X}$. This channel is itself a q-SC, an assertion that can be proven by showing that an equivalent error probability p for $\boldsymbol{X}$ can be computed from the error probabilities p_1 and p_2 for $\boldsymbol{X}^1$ and $\boldsymbol{X}^2$.

Figure 3.8 shows the idea behind the derivation of p for two 6-SCs in series. To derive p, one must consider all combinations of X, X', and X'', such that $X \neq X''$. In general, the number of such combinations is large.

In contrast, as shown in Figure 3.8, deriving $1-p$ involves a much smaller number of combinations, and thus can be considered as a much simpler approach that can be applied for a general value of q.

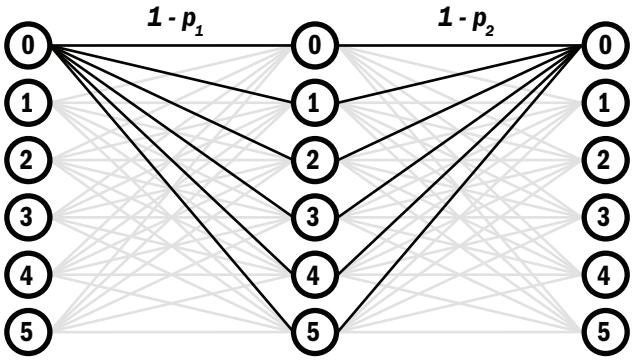


FIGURE 3.8 $P(X'' = 0 \mid X = 0)$

By its definition, $1-p$ is the probability that the output letter X'' of $\boldsymbol{X}^2$ is the same as the input letter X of $\boldsymbol{X}^1$. Visually, the definition for $1-p$ corresponds to the probability associated with all the *paths* connecting identical letters on X and X''. For example, in Figure 3.8, the bold lines show all the paths from $X = 0$ to $X'' = 0$ going through various possible values of X'.

Mathematically, the probability $1-p$ is defined by the expression $P(X'' = i \mid X = i)$. As illustrated in Figure 3.8, this expression is the probability of observing $X'' = i$ given

that $X = i$. The expression can be expanded:

$$\begin{aligned} 1-p &\triangleq P(X'' = i \mid X = i) \\ &= \textstyle\sum_{j\in\mathcal{Q}} P(X' = j \mid X = i)\, P(X'' = i \mid X' = j) \quad . \end{aligned} \tag{3.9}$$

This expansion shows the relationship between Figure 3.8 and the mathematical expression for $1-p$ defined in (3.9). If we assume $i = 0$, then the factor $P(X' = j \mid X = 0)$ in the summation is visualized in Figure 3.8 by the connections from the dot $i = 0$ in the first column to all the dots in the second column. Likewise, the factor $P(X'' = 0 \mid X' = j)$ is visualized by the connections from the second column to the third column. Continuing (3.9),

$$\begin{aligned} 1-p &= (1-p_1)(1-p_2) + (q-1)\frac{p_1}{q-1}\frac{p_2}{q-1} \\ &= (1-p_1)(1-p_2) + \frac{p_1 p_2}{q-1} \quad . \end{aligned} \tag{3.10}$$

The first term in (3.10) corresponds to an error-free transmission through $\boldsymbol{X}^1$ and $\boldsymbol{X}^2$. The second term in (3.10) sums up the $q-1$ probability values of producing a random error in $\boldsymbol{X}^1$ immediately followed by an equally random correction in $\boldsymbol{X}^2$. Due to symmetry, we know that all of these $q-1$ paths must have an identical probability of $\frac{p_1}{q-1}\frac{p_2}{q-1}$.

As we previously mentioned, equation (3.10), which is not overly complicated, is the entry point to our definition of the error probability p of the combined channel $\boldsymbol{X}$. The simplest definition is an algebraic relation that states p in terms of p_1 and p_2, as shown in (3.11). The same algebraic relation can be also cast as a function p with arguments p_1 and p_2, as shown in (3.12). As in the case of the q-EC, the most useful definition of p is cast in terms of the binary operator *o-plus*, which is denoted by $\oplus$, shown in (3.13) that

conceptually "adds" p_1 and p_2 into a single value.

$$p = 1-(1-p_1)(1-p_2)-(p_1 p_2)/(q-1) \tag{3.11}$$

$$= p(p_1, p_2) \tag{3.12}$$

$$\triangleq p_1 \oplus p_2 \tag{3.13}$$

With the $\oplus$ operator, we can *combine* two adjacent q-SCs. To *compare* two different (not necessarily connected) q-SCs, we need to define a relational binary operator denoted by $\preceq$. Although the semantics of $\preceq$ is context-sensitive, the statement $A \preceq B$ reads: A is no less preferred than B. Two examples of different contexts for $\preceq$ are shown in (3.14) and (3.15) below.

$$\boldsymbol{X}^1 \preceq \boldsymbol{X}^2 \tag{3.14}$$

$$p_1 \preceq p_2 \tag{3.15}$$

Equation (3.14) states that the q-SC $\boldsymbol{X}^1$ is no less preferred than $\boldsymbol{X}^2$, while equation (3.15) states that the channel parameter p_1 of $\boldsymbol{X}^1$ is no worse than p_2 of $\boldsymbol{X}^2$. At the first glance, the first statement seems to allow a great deal of latitude for the exact interpretation of "preferred". However, the second statement leaves no such ambiguity. For the error probabilities $p_1, p_2 \in \mathbb{R}$, the only logical interpretation is that $p_1 \preceq p_2$ if and only if $p_1 \leq p_2$, where $\leq$ is the standard *less-than-or-equal-to* operator for real numbers. This then implies the following non-ambiguous definition:

$$\boldsymbol{X}^1 \preceq \boldsymbol{X}^2 \quad \Leftrightarrow \quad p_1 \leq p_2 \quad . \tag{3.15'}$$

In the next subsection, we will explore in great detail the algebraic properties of $\oplus$ and $\preceq$ in our attempt to prove that just as their q-EC counterparts, the two q-SC operators also satisfy the properties in **P** and are thus compatible with generalized Dijkstra's

algorithm (GDA).

Algebraic Properties of $\oplus$ and $\preceq$

Using Figure 3.8, we discussed how a series combination of any two q-SCs produces another q-SCs, and showed that the error probability p for this combination has the same mathematical expression for a q-SC. There are other algebraic properties in **P** that need to be satisfied by $\oplus$, $\preceq$, and Λ to ensure their compatibility with the GDA. Here, we shall prove all of them.

Let us begin by proving the properties of $\oplus$. The set Λ contains all possible value for error probability, which lies in the range of real numbers $[0, \frac{q-1}{q}]$. Throughout the proofs, a, b, and c denote any three elements of Λ.

① *Closure.* Λ satisfies closure if $a, b \in \Lambda$ implies $p = a \oplus b \in \Lambda = [0, \frac{q-1}{q}]$. Although one might be inclined to use the same algebraic approach used in proving closure for q-ECs, the much simpler route is to utilize calculus. Recall that as shown in equation (3.12), the operator $\oplus$ can also be thought of as a function $p(a,b)$. If we can show that for $(a,b) \in \Lambda^2$ the minimum and maximum values of $p(a,b)$ are also contained in Λ, we prove closure. First, we compute the partial derivatives of p:

$$\begin{aligned} p'(b) = \frac{\partial p}{\partial a} &= 1 - b\frac{q}{q-1} \\ p'(a) = \frac{\partial p}{\partial b} &= 1 - a\frac{q}{q-1} \quad . \end{aligned} \tag{3.16}$$

The partial derivatives are linear functions of single variables. At $(a,b) = (0,0)$, the slope $(p'(a), p'(b))$ is $(1,1)$. The minimum and maximum of $p(a,b)$ are located at either (a,b) where $p'(a) = p'(b) = 0$, which is $(\frac{q-1}{q}, \frac{q-1}{q})$ or at the corner points. Figure 3.9 shows a plot of $p(a,b)$ for $q = 6$. The plot for other values of q look very similar, except for the upper bounds of $\frac{q-1}{q}$ for a, b, and $p(a,b)$.

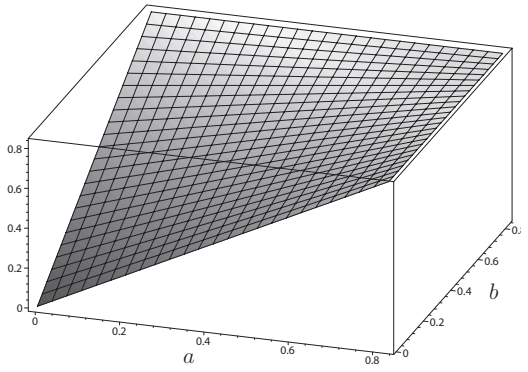


FIGURE 3.9 $p(a,b)$ for $q=6$

From Equation (3.11), we can compute the value of $p(0,0)=0$. Equation (3.16) tells us that because the partial derivatives are always positive for all values of a and b, from $(0,0)$, the value of $p(a,b)$ can only increase. In agreement with Equation (3.16), the maximum value is reached at $((q-1)/q,(q-1)/q)$. However, Equation (3.11) tells us that the maximum is also reached on the lines $a=(q-1)/q$ or $b=(q-1)/q$. Closure is thus satisfied. (Q.E.D.)

② *Associativity.* The order in which $\oplus$ is evaluated does not matter. For the sake of brevity, we do not expand the algebraic expressions in (3.17) and (3.18) to prove their equivalence. Proof:

$$
\begin{aligned}
(a\oplus b)\oplus c &= \left[1-(1-a)(1-b)-\frac{ab}{q-1}\right]\oplus c \\
&= 1-\left[(1-a)(1-b)+\frac{ab}{q-1}\right](1-c) \\
&\quad -\left[1-(1-a)(1-b)-\frac{ab}{q-1}\right](\frac{c}{q-1}) \qquad (3.17)\\
&= 1-\left[(1-a)((1-b)(1-c)+\frac{bc}{q-1})\right] \\
&\quad -\left[1-(1-b)(1-c)-\frac{bc}{q-1}\right](\frac{a}{q-1}) \qquad (3.18)\\
&= a\oplus\left[1-(1-b)(1-c)-\frac{bc}{q-1}\right] \\
&= a\oplus(b\oplus c) \qquad \text{(Q.E.D.)}
\end{aligned}
$$

③ *Commutativity.* The operand position does not matter. Proof:

$$\begin{aligned} a \oplus b &= 1 - (1-a)(1-b) - (ab)/(q-1) \\ &= 1 - (1-b)(1-a) - (ba)/(q-1) \\ &= b \oplus a \end{aligned} \qquad \text{(Q.E.D.)}$$

④ The identity element 0 satisfies $a \oplus 0 = a, \ \forall a \in \Lambda$. Proof:

$$\begin{aligned} a \oplus 0 &= 1 - (1-a)(1-0) - (a \cdot 0)/(q-1) \\ &= 1 - (1-a) - 0 \\ &= a \end{aligned} \qquad \text{(Q.E.D.)}$$

⑤ The absorptive element $\infty \equiv (q-1)/q$ satisfies $a \oplus \infty = \infty, \ \forall a \in \Lambda$. Proof:

$$\begin{aligned} a \oplus \infty &= 1 - (1-a)(1-\infty) - (a \cdot \infty)/(q-1) \\ &= 1 - (1-a)/q - a/q \\ &= (q-1)/q = \infty \end{aligned} \qquad \text{(Q.E.D.)}$$

⑥ The operator $\preceq$ introduces a total order on Λ that categorizes every pair of elements $a, b \in \Lambda$ into $a \preceq b$ or $a \preceq b$ (or both, in which case $a = b$). As with the q-ECs, the space Λ for the q-SCs is just $[0, \frac{q-1}{q}] \in \mathbb{R}$, with $\preceq$ defined as the standard operator $\leq$, which satisfies total order. In addition, the $\preceq$ is also reflexive (i.e., $a \preceq a$), anti-symmetric (i.e., $a \preceq b$ and $b \preceq a$ implies $a = b$), and transitive (i.e., $a \preceq b$ and $b \preceq c$ implies $a \preceq c$). The standard $\leq$ operator also automatically satisfies these properties. (Q.E.D.)

⑦ The least element 0 satisfies $0 \preceq a \ \ \forall a \in \Lambda$. For q-SCs, the real number 0 is the least element in the space $\Lambda = [0, \frac{q-1}{q}] \in \mathbb{R}$. (Q.E.D.)

⑧ *Strict isotonicity.*

$$
\begin{aligned}
a \oplus c &< b \oplus c \\
1-(1-a)(1-c)-(ac)/(q-1) &< 1-(1-b)(1-c)-(bc)/(q-1) \\
(1-a)(1-c)+(ac)/(q-1) &> (1-b)(1-c)+(bc)/(q-1) \\
(1-a)(1-c)-(1-b)(1-c) &> (bc-ac)/(q-1) \\
(1-c)(b-a) &> c(b-a)/(q-1) \\
(1-c) &> c/(q-1) \\
(q-1)(1-c)-c &> 0 \\
(q-1)-qc &> 0 \\
c &< (q-1)/q \qquad \text{(Q.E.D.)}
\end{aligned}
$$

3.3 Gilbert Channels

So far, in Sections 3.1 and 3.2, we have only discussed memoriless channels. To the first approximation, many different types of modern wired and wireless communication links can be considered memoriless, and depending on the types of signal used, these links can be mathematically modeled as binary erasure channels, binary symmetric channels, or AWGN channels, etc. However, in many cases, accurate behavior and performance analysis has to take into account the fact that in reality, channel error exhibits memory.

The Gilbert channel model [1] is one of the simplest models for channels with memory. Mathematically, it is nothing more than a Markov chain with two states: G and B, with their outgoing transition probability values denoted by g and b, respectively. In this subsection, we assign B (for "Bad") to be the state where undesirable events occur, e.g., packet loss, packet error, fading higher than a predetermined threshold, etc. Likewise,

we assign G (for "Good") to be the state where information is transmitted without error.

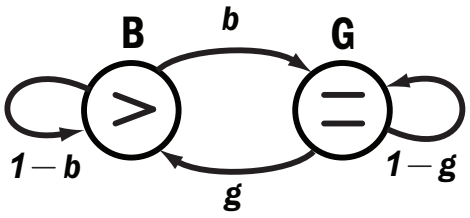


FIGURE 3.10 A Gilbert channel

Despite (also because of) its apparent simplicity, the Gilbert Channel Model (GCM) has been widely used in analyzing a wide variety of networks [2]. For example, GCM is used to analyze the performance of slotted ALOHA over fading communications channels [3], or correlated loss over TCP/IP networks [4]. Other relevant examples include performance analysis of real-time wireless communications [5]. For many wireless fading channels, GCM is a very attractive alternative to sophisticated models such as Hidden Markov Models (HMMs) [6]. For one, GCMs are analytically tractable. In addition, using GCMs, packet-level network QoS can be easily computed.

GCM becomes even more important with the advent of Universal Mobile Telecommunications Systems (UMTS) networks where multimedia (especially speech) and data packets will have to coexist in the underlying common Wideband Code-Division Multiple Access (WCDMA) networks.

The recent UMTS recommendations for QoS measures is based on a set of user satisfaction assessments of individual speech and data sessions [7]. For speech services, user satisfaction drops significantly in the presence of long spans (in terms of packets) of service outage. For data services, disruption and termination comes from successive retransmissions.

These new QoS measures require the next level of approximation of behavioral analysis that is not available from system outage probability analysis (which inherently assumes zero correlation between the outages) commonly found in CDMA literature [8, 9]. Recently, detailed analysis (in the contexts of QoS and capacity) of WCMDA [10, 11] that

incorporates correlated outage behavior through the use of GCM have been proposed.

Let us now define the discrete-time q-ary Gilbert Channel (q-GC) more precisely. First, let us consider a q-GC denoted by $\boldsymbol{X}$. As previously said, $\boldsymbol{X}$ is a Markov chain with two states, G (for "Good") and B (for "Bad"), with outgoing transition probabilities g and b, where $0 \leq g, b \leq 1$. All possible pairs of (g, b) form the space of all GCs denoted by $\Lambda = [0,1] \times [0,1]$.

With this definition, every possible instantiation of $\boldsymbol{X}$ corresponds to an element in Λ that encodes the appropriate probabilistic parameters g and b. Exceptions to this generalization are the elements with either $g = 0$ or $b = 0$. From Figure 3.10, we can prove that for these elements, the corresponding GCs quickly converge to their deterministic steady-state behaviors.

Just like a q-EC or a q-SC, a q-GC also takes input symbols and converts them into output symbols. To be consistent, let us denote the random variables corresponding to the input and output symbols of $\boldsymbol{X}$ by X and X', respectively. Unlike with a q-EC or a q-SC, however, at any time t, a GC also maintains an internal state which is a Markovian random variable denoted by $\mathfrak{x}_t$. As previously mentioned, the value of $\mathfrak{x}_t$ can be either G or B. In this notation, the probability of observing the current internal state $\mathfrak{x}_t$ is given by:

$$\begin{aligned} P(\mathfrak{x}_t = B \,|\, \mathfrak{x}_{t-1} = G) &= g \\ P(\mathfrak{x}_t = G \,|\, \mathfrak{x}_{t-1} = B) &= b \\ P(\mathfrak{x}_t = G \,|\, \mathfrak{x}_{t-1} = G) &= 1 - g \\ P(\mathfrak{x}_t = B \,|\, \mathfrak{x}_{t-1} = B) &= 1 - b \quad . \end{aligned} \tag{3.19}$$

The input and output symbols are drawn from a common alphabet Q that contains $q+1$ letters $0, \ldots, q-1, \epsilon$, where the letter ϵ again represents the erasure symbol. When the GC is in the B state, i.e., when $\mathfrak{x}_t = B$, the output symbol is always the erasure symbol

regardless of the input symbol. In contrast, when the GC is in the G state, i.e., when $\mathfrak{x}_t = G$, the input symbol is identical to the output symbol. Of course, the input symbol can also be ϵ. In this case, the "Good" channel will simply retransmit the erasure symbol.

$$\begin{aligned} P(X' = X \mid \mathfrak{x}_t = G) &= 1 \\ P(X' = \epsilon \mid \mathfrak{x}_t = B) &= 1 \end{aligned} \tag{3.20}$$

The last equation shows that unlike the q-EC or the q-SC, for a GC, the conditional probability equation relating the input and output symbols depends on $\mathfrak{x}_t$. Therefore, the entries of the corresponding matrix $\boldsymbol{X}$ depend on $\mathfrak{x}_t$. In the G and B states the *realizations* of $\boldsymbol{X}$ are denoted by $\boldsymbol{X}_G$ and $\boldsymbol{X}_B$.

For example, consider a particular GC with $q = 5$ and $p = 0.05$. For this parameter, the $\boldsymbol{X}_G$ and $\boldsymbol{X}_B$ matrices are shown in the left and right matrices below. The row index $i \in \{0 \ldots 5\}$ corresponds to the input symbol $X \in \mathcal{Q} = \{0 \ldots 4, \epsilon\}$, and the column index $j \in \mathcal{Q}$ corresponds to the output symbols $X' \in \mathcal{Q}$. Each entry $\boldsymbol{X}_{hij}$ represents the probability $P(X' = j \mid X = i\,, \mathfrak{x}_t = h)$. If $\mathfrak{x}_t = G$, then the left matrix $\boldsymbol{X}_G$ is used. Otherwise, if $\mathfrak{x}_t = B$, then $\boldsymbol{X}_B$ is used.

$$\begin{array}{cccccc} 1.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 1.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 1.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 1.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 1.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 1.00 \end{array} \qquad \begin{array}{cccccc} 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 1.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 1.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 1.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 1.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 1.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 1.00 \end{array}$$

The visual representation of $\boldsymbol{X}_G$ and $\boldsymbol{X}_B$ is undoubtedly more complex than what we have shown for q-ECs and q-SCs. Figure 3.11 shows the two realizations of $\boldsymbol{X}$, along with their transition probabilities. The figure also shows that for a q-GC, the parametrization

is on its transition probabilities (the values of b and g), and not on its channel parameters.

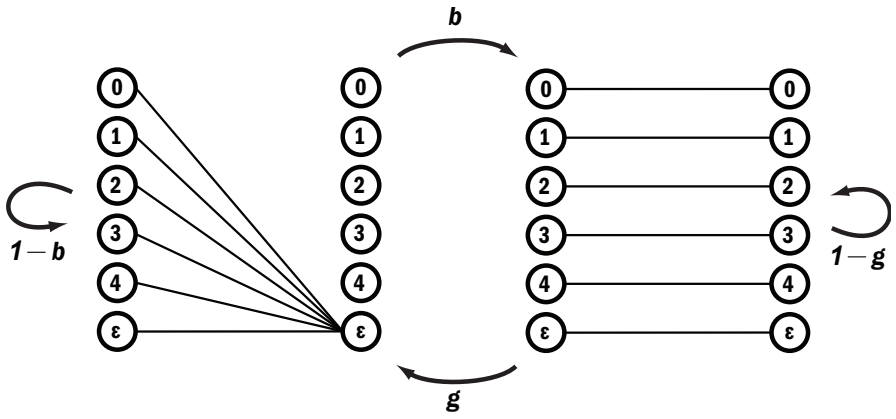


FIGURE 3.11 A visual representation of the 5-ary Gilbert channel

Figure 3.11 shows why a q-GC exhibits memory. If at time t the q-GC experiences a failure (i.e., it is in state B), then the probability of experiencing another failure at time $t+1$ is $1-b$. If the magnitude of $1-b$ is large (or small) enough, we will observe many long (or short) failure runs.

So far, we have not discussed the state at which a q-GC starts at $t=0$. For convention, let us decide that $\mathfrak{x}_0 = G$. With this convention, we can compute the probability of observing any particular failure pattern. While informative, the probabilities of individual failure patterns are not very useful because after a length of time t, there will be a very large number (2^{t-1}) of different patterns, all with very small probabilities. A far more useful quantity to compute is the "stationary" (which usually means long-term) probability of failure in a typical (very long) input pattern.

Let us denote the stationary failure and success probabilities by the symbols β and γ, respectively. The two stationary probabilities must add up to one: $\beta+\gamma=1$. Interestingly, β and γ are also just the ratios between the amount time spent by a q-GC dwelling in the B and G states and the total amount of time t. From the standard theory of Markov chain, we can compute β and γ as functions of b and g using the following

simple formulas:

$$\beta = \beta(b,g) = P(\mathfrak{x}_t = B) = \frac{g}{b+g}$$
$$\gamma = \gamma(b,g) = P(\mathfrak{x}_t = G) = \frac{b}{b+g} \quad . \tag{3.21}$$

This also means that to a really long sequence of letters of length t, a q-GC parametrized by (b,g) will look for a β fraction of the time like $\boldsymbol{X}_B$ (or the left channel diagram in Figure 3.11). For the remaining fraction of the time (which is γ), it will look like $\boldsymbol{X}_G$ (or the right channel diagram in Figure 3.11). This should remind us of the description for a q-EC with $p = \beta$. Although the two channels are obviously not equivalent, after a long period of time t, both converge to the same failure probability of β. Based on this observation, we develop a more concise diagram for $\boldsymbol{X}$ that is shown in Figure 3.12 below. To distinguish this diagram from that of a standard q-EC, we use a dashed horizontal line for the first connection.

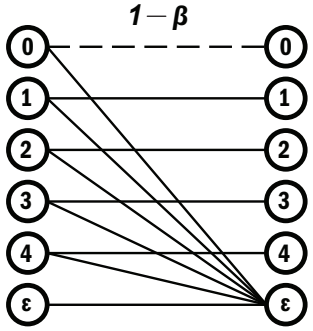


FIGURE 3.12 A more concise diagram for the 5-ary Gilbert channel

Figure 3.12 evokes our previous discussion about combining two q-ECs into a single q-EC. Naturally, this brings up several interesting questions about the q-GCs. First, can we combine two q-GCs in series into a single q-GC? Second, is it possible to compare two q-GCs? In the remainder of this section, we shall show that both intuitively and mathematically, the answer to both questions is indeed positive. To answer these questions using the same arguments we used in our discussions on the q-ECs, we have to make some adjustments due to the fact that Figure 3.12 uses the long-term probabilities

of failure and success of symbol transmission. We want to show that the equivalent transition probabilities (b, g) of the series combination denoted by $\boldsymbol{X}$ can be computed from the transition probabilities (b_1, g_1) and (b_2, g_2) of the two component q-GCs denoted by $\boldsymbol{X}^1$ and $\boldsymbol{X}^2$.

Figure 3.3 shows a series combination of $\boldsymbol{X}^1$ and $\boldsymbol{X}^2$ with $q = 5$. The transition probabilities (b_1, g_1) and (b_2, g_2) of these channels could be different. In the figure, there are three columns of dots: the leftmost column corresponding to the input symbol of $\boldsymbol{X}^1$, the middle column corresponding to either the output symbol of $\boldsymbol{X}^1$ or the input symbol of $\boldsymbol{X}^2$, and the rightmost column corresponding to the output symbol of $\boldsymbol{X}^2$. The random variables corresponding to these three columns are denoted by X, X', and X''.

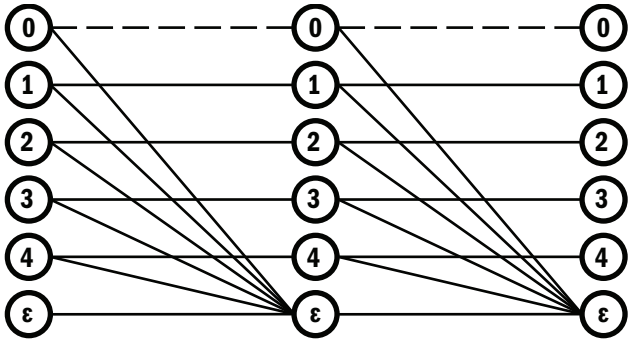


FIGURE 3.13 A series combination of two 5-GCs

Figure 3.13, provides a hint as to why $\boldsymbol{X}$ is also a q-GC. In a q-GC, just as in a q-EC, each symbol on the left column is connected to the identical symbol on the right column with a horizontal line. In Figure 3.13, we can also find horizontal lines connecting identical symbols on the left- and rightmost columns. The horizontal lines in $\boldsymbol{X}$ are of course composite lines. Each one of these lines are composed by one line in $\boldsymbol{X}^1$ and the other line in $\boldsymbol{X}^2$.

Another feature that is shared by both the q-EC and the q-GC is the connection between the dots on the left column, with the exception of ϵ, to the dot representing ϵ on the right column. Similarly, Figure 3.13 shows that this feature is present in a series combination of two q-GCs: each dot $X = i$ on the leftmost column in $\boldsymbol{X}$ are also connected

to the dot representing ϵ on the rightmost column through two distinct paths: (1) the path going through $X' = i$, and (2) another one going through $X' = \epsilon$.

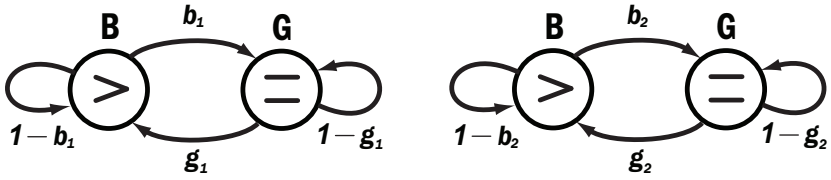


FIGURE 3.14 Two independent Gilbert channels

The major difference is, of course, that in a q-GC, the probabilities represented by these connections are not parameters of random variables associated with observation of individual symbol transmissions. Rather, they are the stationary probabilities γ_1, γ_1, β_2, and β_2 of observing the q-GCs in their various states over a large number of transmissions.

Recall that for a q-GC, transmission failure is a direct function of the channel's state : if the state is B, transmission fails, and vice versa if the state is G, transmission succeeds. For two q-GCs in series, the transmission succeeds if both channels are in the G state, whereas the transmission fails if either one of the channels is in the states B.

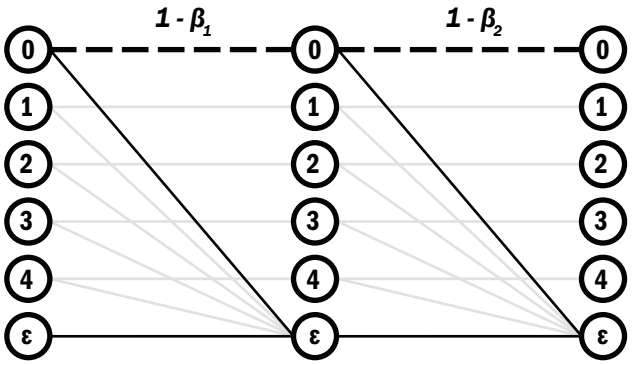


FIGURE 3.15 $P(\mathfrak{x}_t = G)$ and $P(\mathfrak{x}_t = B)$

The above statement notwithstanding, at any given time t, the observed states $\mathfrak{x}_{1t}$ and $\mathfrak{x}_{2t}$ of the two q-GCs are independent of each other. Given this, the probability of finding both $\boldsymbol{X}^1$ and $\boldsymbol{X}^2$ in a particular state configuration must then be equal to the product of the probability of finding the q-GCs in the individual states. Furthermore, the same

statement can be made for the stationary probabilities as well:

$$\begin{aligned} P(\mathfrak{x}_t = G) &= P(\mathfrak{x}_{1t} = \mathfrak{x}_{2t} = G) \\ &= P(\mathfrak{x}_{1t} = G) \cdot P(\mathfrak{x}_{2t} = G) \\ \therefore \qquad \gamma &= \gamma_1 \cdot \gamma_2 \quad . \end{aligned} \tag{3.22}$$

Of course, the preceding discussion has not touched upon how the combined parameter (g, b) of $\boldsymbol{X}$ can be calculated from the component parameters (g_1, b_1) and (g_2, b_2) of $\boldsymbol{X}^1$ and $\boldsymbol{X}^2$. On the surface, the discussion seems to merely provide us with a logical way for interpreting Figure 3.13. However, Equation 3.22 is a good starting place for deriving (g, b).

$$\begin{aligned} \gamma = \gamma_1 \cdot \gamma_2 &= \frac{b_1}{g_1 + b_1} \, \frac{b_2}{g_2 + b_2} \\ &= \frac{b_1 \, b_2}{(g_1 + b_1)(g_2 + b_2)} = \frac{b}{b + g} \end{aligned} \tag{3.23}$$

Equation (3.23) gives one equation for solving the two unknowns g and b. The other equation can be obtained from the definition of g or b, which is best explained by Figure 3.16, which shows two independent q-GCs running simultaneously, together with the transitions and their probabilities.

In Figure 3.16, the symbols g' and b' denote $1 - g$ and $1 - b$, respectively. The diagram provides us with several different ways to obtain the second equation. The simplest second equation turns out to be the definition of g' in terms of g'_1 and g'_2. First, let us describe what g' really means.

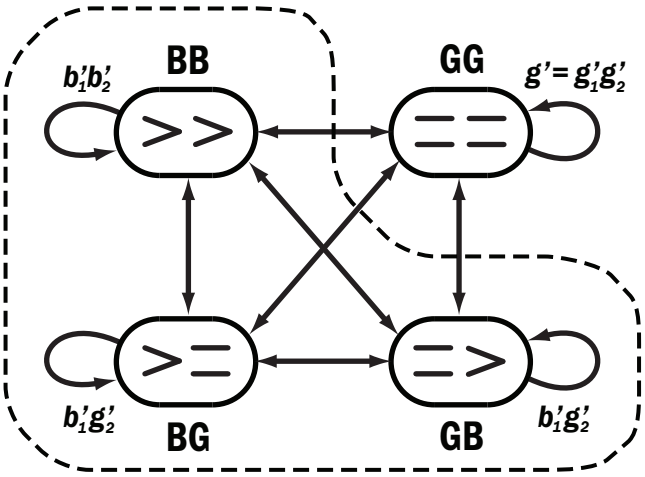


FIGURE 3.16 The derivation of g in terms of g_1 and g_2

The two q-GCs can be thought of as a single, composite q-GC. The binary states G and B of the individual channels are combined into the four states GG, GB, BG, and BB of the composite channel. Since the two channels are connected in series, the three composite states containing at least one B correspond to at least one transmission failure in the chain. The dashed line groups these states together into a composite "Bad" state which we shall denote by $\boldsymbol{B}$. Naturally, we can denote the other remaining "Good" state by $\boldsymbol{G}$. From the diagram, g' denotes the probability:

$$\begin{aligned} g' = 1 - g &= P(\mathfrak{x}_t = \boldsymbol{G} \,|\, \mathfrak{x}_{t-1} = \boldsymbol{G}) \\ &= P(\mathfrak{x}_{1,t} = G \,|\, \mathfrak{x}_{1,t-1} = G) \cdot P(\mathfrak{x}_{2,t} = G \,|\, \mathfrak{x}_{2,t-1} = G) \\ &= (1 - g_1)(1 - g_2) = g_1' g_2' \\ \therefore \qquad g &= 1 - (1 - g_1)(1 - g_2) \quad . \end{aligned} \tag{3.24}$$

Equation (3.24) provides us with the second to solve the two unknowns g and b. With

some manipulation of Equation (3.23), we can state the following:

$$\begin{aligned} g &= 1-(1-g_1)(1-g_2) \\ b &= g\cdot\frac{\gamma}{1-\gamma} = g\cdot\frac{b_1 b_2}{(g_1+b_1)(g_2+b_2)-b_1 b_2} \quad . \end{aligned} \tag{3.25}$$

With Equation (3.25), we can solve for b by substituting the first equation into the second one. These equations also serve as the algebraic definition of (g,b) in terms of the component parameters (g_1,b_1) and (g_2,b_2).

Alternatively, this algebraic definitions of g and b can also be recast as functions of the component parameters, as shown in Equation (3.27). The most useful of all, however, is the definition of (g,b) in terms of the binary operator *o-plus*, which is denoted by $\oplus$, as shown in (3.28). This operator conceptually "adds" (g_1,b_1) and (g_2,b_2) into a single value (g,b) using Equation (3.25).

$$(g,b) = \left(1-(1-g_1)(1-g_2)\,,\,\frac{b_1 b_2[1-(1-g_1)(1-g_2)]}{(g_1+b_1)(g_2+b_2)-b_1 b_2}\right) \tag{3.26}$$

$$= \big(g\,(g_1,b_1,g_2,b_2)\,,\,b(g_1,b_1,g_2,b_2)\big) \tag{3.27}$$

$$\triangleq (g_1,b_1)\oplus(g_2,b_2) \tag{3.28}$$

Visually, we can also define shorthand notations for Figures 3.10 and 3.13 in the same way we defined the $\oplus$ operator. Although the original figures offer the clearest picture of the q-GC and the series combination of two such channels, they are quite cumbersome to use. With the shorthand symbol, we can make a q-GC look like a typical system element. By hiding much of the channel's internal connections, the symbol is easy to use in diagrams containing a large number of interconnected q-GCs.

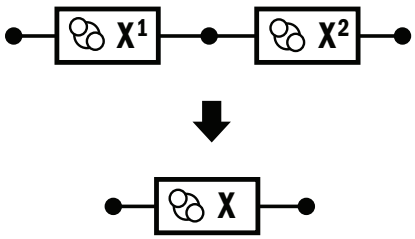


FIGURE 3.17 Series q-GCs and the equivalent q-GC

In the shorthand notation shown in Figure 3.17, the q input and output symbols of a q-GC are shown as terminals. In Figure 3.17, the first terminal represents the input letters of $\boldsymbol{X}^1$. The second terminal represents both the output letters of $\boldsymbol{X}^1$ and the input letters of $\boldsymbol{X}^2$, and the third terminal represents the output letters of $\boldsymbol{X}^2$. The letters are all drawn from a common alphabet $\mathcal{Q}$.

The $\oplus$ operator merely allows us to *combine* two adjacent q-GCs. To *compare* two different (not necessarily adjacent) q-GCs, we need to define a relational binary operator denoted by $\preceq$. The statement $A \preceq B$ reads: A is not less preferred than B, as illustrated in the following two equations:

$$\boldsymbol{X}^1 \preceq \boldsymbol{X}^2 \tag{3.29}$$

$$(g_1, b_1) \preceq (g_2, b_2) \tag{3.30}$$

The meaning of $\preceq$ depends on the context. Equation (3.29) states that the q-GC $\boldsymbol{X}^1$ is not less preferred than $\boldsymbol{X}^2$, while equation (3.30) states that the transition probabilities (g_1, b_1) of $\boldsymbol{X}^1$ are not less preferred than (g_2, b_2) of $\boldsymbol{X}^2$. The immediate question to ask is, what does "preferred" mean?

Although the first statement might suggest different ways to interpret the meaning of "preferred", only one of them is compatible with the second statement, leaving us with no ambiguity. For the transition probabilities $(g_1, b_1), (g_2, b_2) \in \Lambda$, the only logical interpretation is that $(g_1, b_1) \preceq (g_2, b_2)$ if and only if $\beta_1 \leq \beta_2$, where $\leq$ is the standard

less-than-or-equal-to operator for real numbers. From this, we can state the following:

$$\boldsymbol{X}^1 \preceq \boldsymbol{X}^2 \quad \Leftrightarrow \quad \beta_1 \leq \beta_2 \quad . \tag{3.30'}$$

In the next subsection, we will discuss the algebraic properties of $\oplus$ and $\preceq$ and prove that just as their q-EC counterparts, the two q-GC operators also satisfy the properties in **P** and are thus compatible with the GDA.

Algebraic Properties of $\oplus$ and $\preceq$

Using Figure 3.15, we proved how a series combination of any two q-GCs is also another q-GC with combined transition probabilities (g, b). In order to ensure the compatibility of $\oplus$, $\preceq$, and Λ with the generalized Dijkstra's algorithm (GDA), we need to prove the other algebraic properties in **P**.

Let us begin by proving the properties of $\oplus$. The set Λ contains all possible transition probability pairs lying in the range of real number pairs $[0,1] \times [0,1]$. Throughout the proofs, let $a, b,$ and c be the shorthand notations for (g_1, b_1), (g_2, b_2), $(g_3, b_3) \in \Lambda$ of the q-GCs denoted by $\boldsymbol{X}^1$, $\boldsymbol{X}^2$, and $\boldsymbol{X}^3$.

① *Closure.* The set Λ satisfies closure if $a, b \in \Lambda$ implies $a \oplus b \in \Lambda = [0,1] \times [0,1]$. We will prove that closure property is satisfied for each of the component. First, let us prove $0 \leq b\,(g_1, b_1, g_2, b_2) \leq 1$. From (3.25).

$$b = g \cdot \frac{\gamma}{1-\gamma} \leq 1 \tag{3.31}$$

Since $0 \leq \gamma \leq 1$, then $\gamma/(1-\gamma) > 0$. If we assume that closure on b is satisfied, then

$0 \leq b \leq 1$, and the following must be true:

$$
\begin{aligned}
& b \leq 1 \\
\Leftrightarrow \quad & g \leq (1-\gamma)/\gamma \\
\Leftrightarrow \quad & g_1 + g_2 - g_1 g_2 \leq [(g_1+b_1)(g_2+b_2) - b_1 b_2]/(b_1 b_2) \\
\Leftrightarrow \quad & g_1 b_1 b_2 + g_2 b_1 b_2 - g_1 g_2 b_1 b_2 \leq g_1 g_2 + g_1 b_2 + g_2 b_1 \\
\Leftrightarrow \quad & g_1 b_1 b_2 + g_2 b_1 b_2 \leq g_1 g_2 b_1 b_2 + g_1 g_2 + g_1 b_2 + g_2 b_1 \\
\Leftrightarrow \quad & g_1 b_1 b_2 + g_2 b_1 b_2 \leq g_1 b_2 + g_2 b_1 \\
\Leftrightarrow \quad & b_1(g_1 b_2) + b_2(g_2 b_1) \leq g_1 b_2 + g_2 b_1
\end{aligned}
$$

In the above, we first derived the second inequality using (3.31). Next, both sides of the inequality are expanded using (3.25). We then multiply both sides with $b_1 b_2 \geq 0$ and add $g_1 g_2 b_1 b_2$ to both sides. At this point, we observe the inequality involves only positive terms.

Removing two positive terms on the right-hand side does not change the inequality. Finally, by appropriately factoring the terms on the left-hand side, and using the fact that $b_1, b_2 \leq 1$, we reach the final inequality that is always true. Next, we claim that $0 \leq g(g_1, b_1, g_2, b_2) \leq 1$. Proof:

$$
\begin{aligned}
& g_1, g_2 \in [0,1] \\
\Leftrightarrow \quad & (1-g_1), (1-g_2) \in [0,1] \\
\Leftrightarrow \quad & (1-g_1)(1-g_2) \in [0,1] \\
\Leftrightarrow \quad & 1-(1-g_1)(1-g_2) \in [0,1] \qquad \text{(Q.E.D.)}
\end{aligned}
$$

② *Associativity*. The order in which $\oplus$ is evaluated does not matter. Instead of presenting the full algebraic proof, which is too long and tedious, we outline the method of proving the associativity property. With a symbolic algebraic manip-

ulator, one can verify that the following is true (for example, by subtracting the l.h.s. from the r.h.s.):

$$\begin{aligned} & g\,(g\,(g_1,b_1,g_2,b_2),b\,(g_1,b_1,g_2,b_2),g_3,b_3) = \\ & \quad g\,(g_1,b_1,g\,(g_2,b_2,g_3,b_3),b\,(g_2,b_2,g_3,b_3)) \\ & b\,(g\,(g_1,b_1,g_2,b_2),b\,(g_1,b_1,g_2,b_2),g_3,b_3) = \\ & \quad b\,(g_1,b_1,g\,(g_2,b_2,g_3,b_3),b\,(g_2,b_2,g_3,b_3)) \quad . \end{aligned} \tag{3.32}$$

③ *Commutativity*. The operand position does not matter:

$$\begin{aligned} & g\,(g_1,b_1,g_2,b_2) = g\,(g_2,b_2,g_1,b_1) \quad \text{and} \\ & b\,(g_1,b_1,g_2,b_2) = b\,(g_2,b_2,g_1,b_1) \quad . \end{aligned} \tag{3.33}$$

④ The identity element $\mathbf{0}$ satisfies $a \oplus \mathbf{0} = a,\ \forall a \in \Lambda$. We shall prove that $\forall b \in [0,1]$, the element $(0,b)$ satisfies the property of the $\mathbf{0}$ element. Proof:

$$\begin{aligned} (g,b) &= (g_1,b_1) \oplus (g_2,b_2) \\ &= (g_1,b_1) \oplus \mathbf{0} \\ &= (g_1,b_1) \oplus (0,b_2) \\ &= \left(1-(1-g_1)(1-g_2)\,,\,\frac{b_1 b_2[1-(1-g_1)(1-g_2)]}{(g_1+b_1)(g_2+b_2)-b_1 b_2}\right) \\ &= \left(1-(1-g_1)(1-0)\,,\,\frac{b_1 b_2[1-(1-g_1)(1-0)]}{(g_1+b_1)(0+b_2)-b_1 b_2}\right) \\ &= \left(1-(1-g_1)\,,\,\frac{b_1 b_2[1-(1-g_1)]}{(g_1+b_1)b_2-b_1 b_2}\right) \\ &= \left(g_1\,,\,\frac{b_1 b_2 g_1}{g_1 b_2}\right) \\ &= (\,g_1,b_1\,) \end{aligned} \tag{Q.E.D}$$

⑤ The absorptive element $\infty \equiv (1,0)$ satisfies $a \oplus \infty = \infty$ for all $a \in \Lambda$. Proof:

$$\begin{aligned}
(g,b) &= (g_1,b_1) \oplus (g_2,b_2) \\
&= (g_1,b_1) \oplus \infty \\
&= (g_1,b_1) \oplus (1,0) \\
&= \left(1-(1-g_1)(1-1),\, \frac{(b_1 \times 0)[1-(1-g_1)(1-1)]}{(g_1+b_1)(1+0)-b_1 \times 0}\right) \\
&= (1,0) \\
&= \infty \qquad \text{(Q.E.D)}
\end{aligned}$$

⑥ *Total order.* The operator $\preceq$ introduces a total order on Λ that categorizes every pair of elements $a,b \in \Lambda$ into $a \preceq b$ or $a \preceq b$ (or both, in which case $a = b$). Equation (3.25) defines b as a linear function of g with a slope of $\gamma/(1-\gamma)$. We can invert this function and obtain, for each γ, the corresponding line $b(g)$ that intersects the origin $(0,0)$.

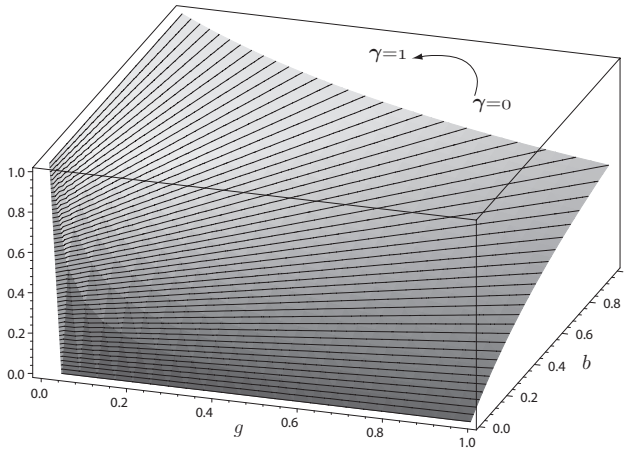


FIGURE 3.18 The contour lines $b(g,\gamma)$

These lines are contour lines for the different values of $\gamma \in [0,1]$. The expression $(g_1,b_1) \preceq (g_2,b_2)$ then simply compares the slopes of the two lines (the steeper the slope, the more preferred). Since $\gamma \in \mathbb{R}$ and we can use the standard *less-than-or-equal-to* $\leq$ operator on $\mathbb{R}$, total order is automatically satisfied. (Q.E.D.)

⑦ The least element $\mathbf{0}$ satisfies $\mathbf{0} \preceq a$ for all $a \in \Lambda$. For any $b \in [0,1]$, the element $(0,b) \in \Lambda$ satisfies the criteria for the $\mathbf{0}$ element. Let β be associated with $(0,b)$, and β' with any element $(g',b') \in \Lambda$. Proof:

$$\beta = \frac{g}{b+g} = \frac{0}{b+0} = 0 \leq \beta'$$

$$\Leftrightarrow \qquad (0,b) \equiv \mathbf{0} \leq (g',b') \qquad \text{(Q.E.D.)}$$

⑧ *Strict isotonicity.* Denote by $\gamma(a)$ and $\beta(a)$ the value of γ and β for $a \in \Lambda$. We use the fact that $\beta = 1-\gamma$ and recall that $\gamma, \beta \in \mathbb{R}$ to prove the isotonicity property as follows:

$$a \prec b$$

$$\beta(a) \leq \beta(b)$$

$$\gamma(a) \geq \gamma(b)$$

$$\gamma(a)\gamma(c) \geq \gamma(b)\gamma(c)$$

$$\gamma(a \oplus c) \geq \gamma(b \oplus c)$$

$$1-\gamma(a \oplus c) \leq 1-\gamma(b \oplus c)$$

$$\beta(a \oplus c) \leq \beta(b \oplus c)$$

$$a \oplus c \prec b \oplus c \qquad \text{(Q.E.D.)}$$

BIBLIOGRAPHY

[1] E.N. Gilbert, "Capacity of a burst-noise channel," *Bell Syst. Tech. J.*, vol. 39, pp. 1253–1265, September 1960.

[2] J.S. Swarts, H.C. Ferreira, "On the evaluation and application of Markov channel models in wireless communications," *Proc. of the 50th IEEE Vehicular Technology*

Conference.

[3] H.H. Tan, S.V. Hung, "Performance analysis of slotted ALOHA over fading communications channels," *INFOCOM '90. Ninth Annual Joint Conference of the IEEE Computer and Communication Societies. Proceedings., IEEE.*

[4] M. Zorzi, R.R. Rao, "The Effect of Correlated Errors on the Performance of TCP" *IEEE Communications Letters*, vol. 1, no. 5, September 1997

[5] K.K. Lee, S.T. Chanson, "Packet loss probability for real-time wireless communications," *IEEE Trans. Veh. Technol.*, vol. 51, no. 6, pp. 1569–1575, Nov. 2002.

[6] W. Turin. and R. Van Nobelen, "Hidden Markov modeling of fading channels," *Proc. IEEE Veh. Technol. Conf.,* May 1998, pp. 1234–1238.

[7] ETSI, "Selection Procedures for the choice of radio transmission technologies of the UMTS," *Universal Mobile Telecommunications System (UMTS);* V3.2.0 (1998–04), Annex B.

[8] K.S. Gilhousen, I.M. Jacobs, R. Padovani, A.J. Viterbi, L.A.Jr. Weaver, and C.E. Weatley, "On the capacity of a cellular CDMA system," *IEEE Trans. Veh. Technol.,* vol. 40, no. 2, pp. 303–312, May 1991.

[9] J.S. Evans, D. Everitt, "On the teletraffic capacity of CDMA cellular networks," *IEEE T. Veh. Tech.,* vol. 48, no. 1, pp. 153–166, Jan. 1999.

[10] T. Elshabrawy, R. Le-Ngoc, "Capacity of future WCDMA networks supporting multimedia services," *Selected Areas in Communications, IEEE Journal on,* vol. 24, no. 1, pp. 54–64, Jan. 2006.

[11] T. Elshabrawy, T. Le-Ngoc, "Gilbert channel approximation for downlink performance evaluation of WCDMA systems," *Proc. of 62nd IEEE Vehicular Technology Conference,* vol. 1, pp. 382–386, 28–25 Sept., 2005.

[12] Sobrinho, J.L., "Algebra and algorithms for QoS path computation and hop-by-hop routing in the Internet," *IEEE/ACM Transaction on Networking.*, vol. 10, pp. 541–550, August 2002.

[13] T.H. Cormen; C.E., Leiserson; R.L. Rivest, *Introduction to Algorithms*, MIT Press, MA and McGraw-Hill, NY, 1990.

[14] E. Soedarmadji, "Worst-Case Routing Performance Metrics for Sensor Networks," *Proc. of the 5th IEEE International Conference on Pervasive Computing and Communications Workshops,* pp. 313–317, 19-23 March 2007.

[15] E. Soedarmadji, "Optimal Routing in the Worst-Case-Error Metric", *Global Telecommunications Conference, 2006. GLOBECOM '06. IEEE* pp. 1–5, Nov. 2006.

[16] C. Jin, Q. Chen, S. Jamin, "Inet: Internet Topology Generator," *Tech Report UM-CSE-TR-433-00,* University of Michigan, Ann Arbor.

4

The Gas Station Problem

THE generalized Dijkstra's algorithm discussed in the previous chapters focuses on allowing a much more general class of edge weights and the corresponding algebraic operations of adding and comparing them. In this chapter, we introduce another type of Generalized Dijkstra's Algorithm that allows a subset of network nodes to reduce the accumulated path cost down to zero.

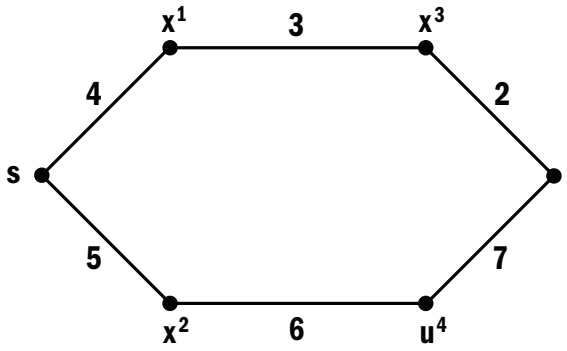


FIGURE 4.1 A network containing cost-resetting nodes

Figure 4.1 shows a network that contains such nodes. The two paths from s to d are $s-x^1-x^3-d$ and $s-x^2-u^4-d$, and the cost-resetting node is u^4. Path cost is referenced to the starting node s, which means that the cost at s is zero, and accumulates at nodes further away from s. Along the first path, the accumulated cost is 4 at x^1, and $4+3=7$ at x^3, and $4+3+2=9$ at d. Along the second path, however, the accumulated cost at x^2

is 5, but the cost at u^4 is zero. Therefore, the second path ends up costing only 7, thanks to u^4.

Called the "Gas Station Problem", our generalized shortest-path problem goes one step further. In this problem, a feasible path must never accumulate cost beyond a pre-specified maximum value. Although in the next section, we describe the problem in terms of vehicles and gas stations, the problem has general applications beyond transportation network optimization.

4.1 The Gas Station Problem

4.1.1 Introduction

In this section, we introduce a generalized shortest-path problem (SPP) that addresses the problem of finding the shortest travel path for a vehicle traveling from an origin node to a destination node in a transportation network, considering fuel consumption, limited onboard vehicle fuel capacity, and the presence of refueling facilities – "gas stations" – at some network nodes. We call this problem the *"Gas Station Problem"* (GSP).

In the GSP, the vehicle begins its journey at the source node with an initial amount of fuel in its tank. If this amount is large enough to allow the vehicle to reach any network node (including the destination) without refueling, then the problem is reduced to the standard shortest-path problem.

The GSP is more general than the SPP because in the GSP, the initial fuel amount might be such that the destination cannot be reached without visiting at least one gas station. The GSP attempts to answer two questions: (a) given an initial fuel amount, is travel from the origin to the destination possible? (b) if so, which path minimizes the travel distance?

The work presented in this section is primarily motivated by the problem facing drivers

of conventional vehicles operating in remote areas where refueling facilities are sparsely located. A similar problem is also faced by owners of alternative fuel vehicles who have to consider the sparseness of refueling facilities in planning their travel. The problem also occurs in project management where project milestones are connected by paths with associated time and cost element. Refueling stations can then be thought of as project milestones at which project funding is restored to its maximum capacity. Most importantly, however, as we will discuss in the next section, this problem has important applications in communication networks, where the "vehicle" becomes a representation of data packets and codewords.

The GSP also has an alternative formulation, which is relevant to the main topic of this section. Instead of focusing on fuel consumption and requiring that fuel level stay above zero at all time, this formulation focuses on accumulation of some quantity (waste, heat, etc.) that cannot exceed some maximum onboard capacity. Instead of gas stations, the network has disposal stations that reset the vehicle waste level back to zero.

The GSP belongs to a general class of SPP with side constraints [1], also referred to as the resource-constrained shortest-path problems (RCSPP) [3, 7, 9, 11]. There are many transportation problems that involve solving the SPP with constraints as a subproblem. One example of these problems is the aircraft rotation problem, where the constraining resource is the flying time between required maintenance [2]. Other similar problems include the airline crew pairing problem [12], and the flight crew scheduling problem [8].

One example that is especially relevant to the GSP is the vehicle routing problem with time window (VRPTW) [5, 6, 10, 13]. In the VRPTW, the goal is to minimize the vehicle travel and idle time of a route where route nodes have specific delivery time interval requirements. Early arrival at a node incurs idle time and late arrival makes a route infeasible. As in all the previous examples, the resource is not replenishable. If there is no gas station, the GSP is a VRPTW with node intervals specified in fuel units $[0, C_{\max}]$.

Once gas stations are introduced, the GSP does not fit the VRPTW model and forms a new class of constrained shortest-path problems with renewable resource.

In this section, we generalize the SPP by allowing the nodes to add either 0 or $C_{\max} - C$ fuel units to a visiting vehicle. We propose a method that consists of three stages, each utilizing standard graph algorithms. These algorithms have many well-studied modifications that can be used to customize the method to accomodate additional constraints.

With the flexibility derived from its components, the proposed method can be easily combined with other optimization methods to solve various Vehicle Routing and Scheduling Problems (VRSP) that require similar fuel and refueling constraints. Without assuming planarity or Euclidean geometry, in the worst case, the method proposed in this section can provide a deterministic answer in a time complexity of at most $O(V^3)$.

4.1.2 Formulation

Let V be the set of nodes in a transportation network, and E be the set of edges linking pairs of nodes therein. For each $e \in E$, the length of e is denoted by $d(e)$. Assuming the edges allow traffic to move in both directions, we can encode this information as a labeled undirected graph $G = (V, E, d)$.

Traveling along an edge $e = (u, v)$ that connects nodes u and v incurs a fuel consumption that is linearly proportional to the length $d(e) > 0$ of the edge, $k\,d(e)$. Without loss of generality, let us assume $k = 1$ and express the $d(e)$ in the equivalent fuel units required to drive from u to v.

Let $C(u)$ denote the amount of fuel carried by the vehicle at node u. A vehicle can only travel from u to v along e if it carries with it an amount of fuel $C(u) \geq d(e)$ at u. We call this condition the *feasibility condition* at u (or along e). At v, the vehicle will have $C(u) - d(e)$ fuel units left in the tank.

Suppose a vehicle with a tank capacity of $C_{\max}$ fuel units wishes to travel from node $s \in V$ to another node $t \in V$, as "efficiently" as possible, where efficiency is measured in terms of distance. To achieve this goal, we have to find the shortest-path (SP) $f^*(s, t)$

from s to t that obeys the feasibility condition.

A path f connecting v_1 to v_n through $v_2, \ldots, v_{n-1}$ is a *feasible path* if the feasibility condition is satisfied along all its $n-1$ edges $e_i = (v_i, v_{i+1})$, $i = 1 \ldots n-1$ — i.e., $C(v_i) \geq d(e_i)$. The set of all paths in G will be denoted by P, and the set of all feasible paths by F. We denote the sum of edge weights of $p(u,v) \in P$ by $d(u,v)$. If $p = e = (u,v)$, then $d(p) = d(u,v) = d(e)$.

Of course, $F = F(G)$ depends on the particular network G under consideration. However, G only provides us with information on fuel consumption. To determine $F(G)$, we need to specify the resources available to the vehicle.

At s, the vehicle starts with $C(s) \leq C_{\max}$ units of fuel. A subset of nodes, i.e., the "gas stations", $U \subseteq V$ allows a visiting vehicle to replenish its onboard fuel to the maximum capacity $C_{\max}$. For example, suppose a vehicle arrives at u with $C(u)$ fuel units, traveling along $e = (u,v)$ toward a gas station $v \in U$. Departing from v, the vehicle will have $C_{\max}$ units of fuel, not $C(u) - d(e)$.

A GSP is fully specified by its U, $C_{\max}$, $C(s)$, and $G = (V, E, d)$. We can then pose two questions. First, is it possible to travel from s to t? In other words, is F an empty set? Second, if F is not an empty set, then which path $f^* \in F$ minimizes the travel distance from s to t, and what is its length $d(f^*)$?

Theorem 1. A path f^* solves the GSP specified by U, $C_{\max}$, $C(s)$, and the graph $G = (V, E, d)$ *if and only if* it solves the SPP specified by

$$
\begin{aligned}
&G' = (V', E', d'), \quad \text{where} \\
&V' = \{s, t\} \cup U \\
&E' = \{\, \operatorname{argmin}_{f'} d(u,v) \mid f' \in F' \,\} \quad \text{where } u \neq v \text{ and } u, v \in V', \\
&F' = \{\, f(u,v) \mid f \in F, \text{ i.e. } d(u,v) \leq C_{\max}, \text{ and } d(s,v) \leq C(s) \,\} \\
&d' = \{\, d(e') \mid e' \in E' \,\}.
\end{aligned} \tag{4.1}
$$

In above set of equations, we assume that $u \neq v$ and that $u, v \in V'$. The set V' contains the source node s, the destination node t, and the gas station nodes. These nodes are all linked by "virtual" edges $e' \in E'$. In turn, each of these virtual edges is the shortest-path among all feasible paths $f' \in F' \subseteq F$ connecting the end nodes u and v in $V' \subseteq V$.

Proof. First, let us note that the optimal path f^* consists of $n+1$ edges that connect all the path nodes: $f^* = s \rightsquigarrow u_1 \rightsquigarrow \cdots \rightsquigarrow u_n \rightsquigarrow t$. This optimal path is the solution to the GSP and is a concatenation of $n+1$ segments f_i^*. The nodes u_i are all gas stations, i.e., $u_i \in U$, $i = 1 \ldots n$, and $0 \leq n \leq |U|$. If $n = 0$, then f^* is a direct path $s \rightsquigarrow t$ that does not contain any gas station.

Because f^* solves the GSP, the segments f_i^* must also be the shortest-paths between the nodes in V' that they are connecting. Similarly, the elements in the set $\{u_i\}$ also minimize the sum $d(s,u_1) + \sum d(u_i, u_{i+1}) + d(u_1, t)$. Changing the segments f_i^* or modifying the membership of $\{u_i\}$ results in a longer path g with $d(g) \geq d(f^*)$. Since the nodes s, t, u_i, the segments f_i^*, and the weights $d(f_i^*)$ are all parts of the specification of a SPP $(V', E', d') \subseteq (V, E, d)$, then f^* is a solution of the SPP specified by G'.

For the reverse proof, suppose that g^* solves the SPP, and that g^* is different from f^* that solves the GSP. We proved that f^* solves the SPP, and thus $d(f^*) \leq d(g^*)$. If $d(f^*) < d(g^*)$, then g^* does not solve the SPP, which is a contradiction. Therefore, $d(f^*) = d(g^*)$, which means f^* solves the GSP. □

4.1.3 Algorithm

This section presents an algorithm that implements the method suggested in Theorem (10). It derives G' from G, and obtain the path f^* that solves the SPP specified by G'. By Theorem (10), f^* then solves the GSP specified by G.

GAS-STATION($G, U, C(s), C_{\max}$)

1: Remove all edges $\{ e \in E \mid d(e) > C_{\max} \}$ from E

2: Remove all nodes $\{ v \in V \mid \deg(v) = 0 \}$ from V

3: **for** $v_1 \in V'$ **do**

4: Obtain the SP tree from v_1 using DIJKSTRA(G, v_1)

5: **for** $v_2 \in V' \neq v_1$ **do**

6: Add an edge $e' = (v_1, v_2)$ into E and $d(e') = d(v_1, v_2)$ into d'

7: **end for**

8: **end for**

9: Remove all edges $\{\ e' = (v_1, v_2) \in E' \mid d(e) > C(v_1)\ \}$ from E'

10: Remove all nodes $\{\ v' \in V' \mid \deg(v') = 0\ \}$ from V'

11: Obtain the SP from s to t using DIJKSTRA(G', s)

Lines 1 to 8 form the first stage of the proposed method. First, on line 1 and 2 it prunes all infeasible edges $e \in E$ where $d(e) > C_{\max}$. Then, on line 4, it runs the Dijkstra algorithm (DA) [4] on all $v_1 \in V'$, every time producing an shortest-path tree rooted at v_1. From the tree, edges e' connecting v_1 and $v_2 \in V'$ with weights $d(v_1, v_2)$ are added into E' on line 6. It is possible to add only feasible edges where $d(v_1, v_2) \leq C(v_1)$, but for clarity, we separate this step into a separate stage. This first stage basically calculates all-pairs shortest-path of the nodes $v \in V' \subseteq V$. At the end of the first stage, the set V' and tentatively, E' and d' for the virtual graph G' are obtained.

In the pruning stage on lines 9 and 10, the edges $e' \in E'$ where $d(e') > C_{\max}$ are removed from E', and the nodes $v' \in V'$ with degree 0 are removed from V'. Finally in the third stage on line 11, we obtain the solution $f^*(s,t)$ to the GSP using DA with $s \in V'$ as the source node.

For brevity, let us denote $|V'|$ by n. In this notation, the time complexity of the first stage is $O(nV^2)$. The factor n takes into account the fact that the DA is performed n times, each time rooted on $v \in V'$. The result is a complete graph with n nodes and $n(n-1)$ directed edges. The pruning stage searches linearly over these edges and nodes at a time complexity cost of $O(n^2)$. The final DA has a time complexity cost of $O(n^2)$. Therefore, overall time complexity is bounded by $O(nV^2 + n^2)$, or conservatively, $O(V^3)$.

If the DA subroutine is implemented using Fibonacci heap, then its complexity could be as low as $O(V \log V + E)$. Depending on G, the shortest-path $d(v_1, v_2)$ can also be obtained using other shortest-path algorithms or all-pairs shortest-path algorithms [4].

4.1.4 Numerical Example

In this section, we provide a numerical example of the GSP and its solution using the GAS-STATION algorithm. Consider a transportation network G' in Figure 4.2(a). The nodes s, t, and the gas stations a, b, $c \in V'$ are in italics.

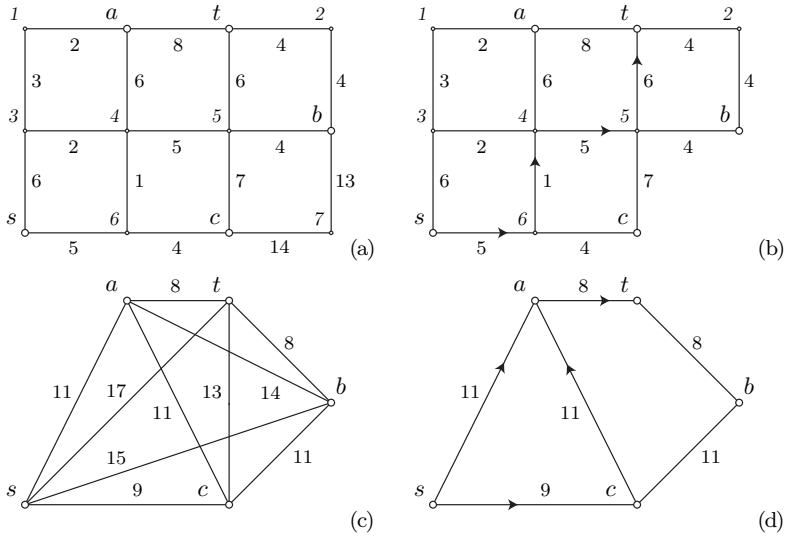


FIGURE 4.2 An example of the Gas Station Problem

First, lines 1 and 2 eliminate the infeasible edges $(c, 7)$ and $(7, b)$ and empty node 7 from G, as shown in Figure 4.2(b). Next, lines 3 to 8 calculate the shortest-paths between the nodes in $V' = \{s, t, a, b, c\}$. For example, from DIJKSTRA(G, s) the SP from s to t is $f^*(s,t) = s \to 6 \to 4 \to 5 \to t$, with $d(f^*) = 5+1+5+6 = 17$. In Figure 4.2(b), $f^*(s,t)$ is the path along the marked edges from s to t.

The shortest-paths between pairs of nodes in V' form the edges connecting $v \in V'$ of G', as shown in Figure 4.2(c). For example, $f^*(s,t)$ in G is now the edge (s,t) in G'. Obviously, any edge $e \in E'$ with $d(e) > C_{\max} = 12$ is infeasible. Figure 4.2(d) shows the result of lines 9 and 10 to Figure 4.2(c), assuming $C(s) = C_{\max} = 12$ fuel units.

From DIJKSTRA(G', s) on line 11, the SP from s to t on G' is $f^*(s,t) = s \rightarrow a \rightarrow t$, with $d(f^*) = 11+8 = 19$. If we assume $C(s) = 10$, then edge (s,a) in G' is infeasible and thus $f^* = s \rightarrow c \rightarrow a \rightarrow t$, with $d(f^*) = 9+11+8 = 28$. In both cases, f^* is a longer path than the solution of a standard SPP on G.

4.1.5 Conclusion

We have presented a worst-case $O(V^3)$ algorithm that solves the problem of finding the shortest travel path for a vehicle with a limited fuel capacity operating in a network with refueling nodes. Future research topics include the stochastic case where edge weights are random variables, and the dynamic case, where the edge weights change during travel.

4.2 APPLICATIONS OF THE GAS STATION PROBLEM

The Gas Station Problem is very closely related to the problems commonly encountered in designing and operating communication networks. For example, one of the biggest problems in successfully integrating sensor networks into mission-critical applications is to ensure that these networks can reliably and reasonably perform under the worst-case scenarios that often produce extreme and unusual events which these networks are designed to detect. One of the key performance measures is the sensor network's ability to route, i.e., choose the appropriate route for transporting, measurement data from the source node to the intended destination. In this section, we show how this problem can be reformulated as the Gas Station Problem.

The strong emphasis on sensor network reliability is understandable. Originally designed for military applications, sensor networks perform functions that have very low error tolerance. Recently, however, sensor networks have steadily entered many areas of non-critical civilian applications where remote environmental monitoring capability is

needed. Examples of such systems include traffic sensors, fire alarms, and motion detectors that are now connected to form large networks in many buildings, factories, and facilities to maintain security, environmental control, and track valuable assets.

State of the art sensor networks are just beginning to be integrated into many mission-critical civilian data monitoring systems. For example, the Advanced National Seismic System (ANSS) [30] and the Pacific Tsunami Warning System operated by the National Weather Service [31] are two large networks of sensors designed to reliably detect, measure, report, and track large scale, catastrophic geological events moving at the speed of several kilometers per second. While these systems are currently implemented as wired sensor networks, many future mission-critical systems would exist in the form of wireless sensor networks designed to detect the presence of dangerous levels of biohazards, radiation, lethal mineshaft gases, etc.

Obviously, these types of applications have very high requirements for performance and reliability. Losing even one symbol, packet, or file can have very disastrous consequences. Mission-critical communication networks are designed to prevent such situations and maximize the probability of successfully transmitting the information to the destination node.

Solving the problem of improving the Quality of Service (QoS) for such networks involves the task of evaluating and comparing network paths (and edges) based on a criteria of optimality that minimizes the worst possible number of packet loss (Worst-Case Erasure or Error, or WCE) received at the target node. This is not a trivial task, especially in wireless sensor networks where the Bit Erasure (or Error) Ratio (BER) could be very high. To make it more challenging, retransmission might be impossible, as the source nodes could have been destroyed or incapacitated.

Fault tolerance can be incorporated into these types of sensor networks by using specialized algorithms, such as the ones reviewed in [32–39]. However, the degree of worst-case fault tolerance ultimately depends on how well the networks are designed. Another

relevant problem to solve for a network designer is: given two or more competing sensor network designs, which one would best meet the worst-case routing performance requirements?

Motivated by these problems, we introduce a new QoS metric called the WCE metric. This metric can be used to measure, compare, and select the network path whose "length" (the WCE length) is minimum. In any metric, a path's length depends on its edges' lengths, and the WCE metric is no exception. To use our metric, we use a network model where each edge represents a set of parameters of a particular channel model.

The edge's parameters (such as the BER) in turn determines its WCE length. The parameters themselves can be used to measure edge and path lengths, and thus qualify as a metric. The WCE length is a non-decreasing function of the length measured in the parameter metric, and thus, the path with the best parameter in a network is also the path with the minimum WCE.

We show that if *some* network nodes are capable of correcting edge errors and erasures, then it is possible to find a path (or paths) of zero WCE length. We do not assume that all nodes can correct errors and erasures. In wireless and ad-hoc networks where compute power and energy are constrained, available resource for complex mathematical operations used by high-performance FEC (such as finite fields arithmetic and iterative algorithms) is limited. It is therefore desirable to have only as many such "overhead" nodes as needed for reliability. In addition, compared to repeater nodes, error-correcting nodes incur delay and consume bandwidth resources. Here we present a $O(V^3)$ algorithm that can compute the best path to transport information with zero WCE through the appropriate error-correcting nodes.

The inclusion of error-correcting nodes is quite realistic. Forward Error Correction (FEC) is gaining acceptance in modern networks. Historically, wired networks have predominantly used the Automatic Repeat reQuest (ARQ) methods over the FEC methods. However, in wireless multimedia applications, FEC outshines ARQ in its ability to signif-

icantly improve network QoS [15–18] without a heavy premium on performance. Unlike TCP and ARQ, FEC does not use return requests and thus consumes less bandwidth [19], especially in large (multicast) wireless networks [20, 21].

Even in peer-to-peer networks, FEC deployed at strategically positioned error-correcting nodes is superior compared to replication [22]. Multicast algorithms such as *Digital Fountain* [23] and *Bullet* [24] employ FEC-based codes. Adaptive [17, 18] and hybrid (ARQ-FEC) QoS-driven algorithms that dynamically adjust FEC level to network conditions have been proposed [25–28] to reduce the bandwidth and computation overhead of FEC methods.

When not limited to the WCE metric and the objective of finding the zero WCE path, the minimum WCE routing algorithm presented in this section can be applied to many other network QoS optimization problems that attempt to minimize the worst-case occurence probability of non-typical, but highly catastrophic events. Once the path with minimum WCE metric is computed, the path length can be used to evaluate the worst-case routing performance of that particular sensor networks.

4.2.1 Formulation and Notation

We model the network as a digraph $G = (V, E)$, where V, E, and Π are the node, edge and path sets of G. The nodes $s, d \in V$ are the source and destination nodes, and $\Pi \subset \Pi$ is the set of all paths from s to d.

A path $\pi \in \Pi$ whose nodes $V_\pi \subset V$ are connected by $E_\pi \subset E$ is denoted by $\langle v_0, \ldots, v_J \rangle$, $\langle e_1, \ldots, e_J \rangle$, or $\langle v_0, e_1, \ldots, e_J, v_J \rangle$. The number of nodes (or edges) in π is denoted by $|\pi|_v$ (or $|\pi|_e$). The symbol $\langle v_i, v_{i+1} \rangle$ denotes the edge connecting the two adjacent nodes v_i and v_{i+1}. If v_i and v_{i+1} are not adjacent to one another, then $\langle v_i, v_{i+1} \rangle$ denotes the path connecting the two nodes. We also define a *partial* path π_j of π to be the path $\langle v_0, \ldots, v_j \rangle$, with $0 < j \leq J$, and a *truncated* path $\bar{\pi}_j$ to be $\langle v_0, e_1, \ldots, v_{j-1}, e_j \rangle$ without the final node.

Denote the message produced at the source by $B \in \mathcal{B}$, where $\mathcal{B}$ is the space of all

allowable messages in the network. In particular, we focus on messages that are constructed from a finite q-ary alphabet Q that contains the letters $\{0,\ldots,q-1\}$. We can use the same notation $B \in Q^n$ for these messages, with $B_l \in Q$ denoting the symbols in the message block, and with $l = 1 \ldots n$.

Alternatively, we can also define the message B as a block that contains n packets of m symbols each. In this definition, instead of an alphabet, Q is then the set of possible packet states that encodes different QoS metrics such as delay, loss, jitter, etc., which then allows our symbol- and alphabet-based results to be applied to other packet- and state-based network QoS problems.

Let B_i and $b_i = B_{li}$ denote the actual content of B and its components B_l as they depart from v_i. In a similar manner, let $\bar{B}_i$ and $\bar{b}_i = \bar{B}_{li}$ denote the values of B and B_l as they leave e_i. Both v_i and e_i are parts of the path $\langle v_0, e_1, \ldots, e_J, v_J \rangle$. Along this path, the message B evolves as follows:

$$B_0 \xrightarrow{e_1} \bar{B}_1 \xrightarrow{v_1} B_1 \xrightarrow{e_2} \bar{B}_2 \xrightarrow{v_2} \cdots \xrightarrow{e_J} \bar{B}_J \xrightarrow{v_J} B_J \quad .$$

We can think of the nodes and edges v_i and e_i as operators $\boldsymbol{v}_i, \boldsymbol{e}_i \in \mathcal{E} : \mathcal{B} \to \mathcal{B}$ that are transforming the content of B. More precisely, these operators can be defined by the following equations:

$$B_i = \boldsymbol{v}_i(\bar{B}_i) \qquad \text{and} \qquad \bar{B}_{i+1} = \boldsymbol{e}_i(B_i).$$

For π, the path evolution operator $\boldsymbol{\pi}$ is defined as a concatenation of the above operators: $\boldsymbol{\pi} = \boldsymbol{v}_J \circ \boldsymbol{e}_J \circ \cdots \circ \boldsymbol{e}_1 \circ \boldsymbol{v}_0$. For π_j, it is $\boldsymbol{\pi}_j = \boldsymbol{v}_j \circ \boldsymbol{e}_j \circ \cdots \circ \boldsymbol{e}_1 \circ \boldsymbol{v}_0$, and for $\bar{\pi}_j$, it is $\bar{\boldsymbol{\pi}}_j = \boldsymbol{e}_j \circ \boldsymbol{v}_{j-1} \cdots \circ \boldsymbol{e}_1 \circ \boldsymbol{v}_0$. Thus, $\boldsymbol{\pi}_j(B_0) = B_j$ and $\bar{\boldsymbol{\pi}}_j(B_0) = \bar{B}_j$.

Define $X : \mathcal{B} \times \mathcal{B} \to \mathcal{M}$ as a function measuring the Hamming distance $x \in \mathcal{M}$ between a message B_0 at v_0 which evolves into B_J at v_J; where v_0 and v_J are connected by $\langle v_0, v_J \rangle$, and $\mathcal{M}$ is the metric space. The distance x from B_i to $B_{i'}$ is denoted by $x = X_{i,i'} = X(B_i, B_{i'})$. Let $X_{i,\bar{i}'}$ denote $X(B_i, \bar{B}_{i'})$. If $e = \langle v_i, v_{i'} \rangle$ then we define the short-

hand $X(e) = X(B_i, \bar{B}_{i'})$.

Except for B_0, the messages B_i at v_i are random variables. Therefore, $X_{0,i}$ must also be random variables. Define $P(X = x, \boldsymbol{\lambda})$ as the probability density of x parameterized by a vector $\boldsymbol{\lambda} \in \Lambda$ — for example, if P is Gaussian and $x \in \mathbb{R}$, then $\boldsymbol{\lambda}$ is the vector of P's mean and variance (μ, σ^2) — where $x = X_{i,i'}$ is the random variable that measures the Hamming distance between the message at v_i and its image at $v_{i'}$. Each edge e_i in the network is characterized by the parameter $\boldsymbol{\lambda}_i$ of the associated error probability density. In all cases, $\boldsymbol{\lambda} = \infty$ denotes the lack of connection between two nodes.

We now provide two possible definitions of the *worst case function* $\bar{x}(\boldsymbol{\lambda}_i, \epsilon)$. The function is really a functional that accepts the $\boldsymbol{\lambda}$-parametrized probability density $P(x, \boldsymbol{\lambda})$ as its argument. In the first definition, the function computes, given $P(x, \boldsymbol{\lambda}_i)$, the worst case "possible" value of x, where "possible" values y are defined as those y values with probability $P(y, \boldsymbol{\lambda}_i) > \epsilon$. We also observe from (4.7) that $\bar{x}(1, \epsilon) = \bar{x}(\infty, \epsilon) = n$ and $\bar{x}(0, \epsilon) = 0$. Note that from (4.7), $\bar{x}(1, \epsilon) = \bar{x}(\infty, \epsilon) = n$, $\bar{x}(0, \epsilon) = 0$, and $\bar{x}(p, \epsilon)$ is not continuous.

$$\bar{x}(\boldsymbol{\lambda}_i, \epsilon) = \max_x \{\, x \mid P(x, \boldsymbol{\lambda}_i) \geq \epsilon\,,\; x \in \mathcal{M}\} \tag{4.2}$$

$$\bar{x}(\boldsymbol{\lambda}_i, \epsilon) = \max_x \{\, x \mid P(X > x, \boldsymbol{\lambda}_i) \geq \epsilon\,,\; x \in \mathcal{M}\} \tag{4.3}$$

For convenience, let us define the function $\beta : \Pi \to \Lambda$ that maps a path π (or an edge e_i) into a density parameter $\boldsymbol{\lambda}_\pi$ (or $\boldsymbol{\lambda}_i$) and the function $\omega : \Pi \to \mathcal{M}$ that maps a path or an edge (given ϵ) into its worst-case value $x \in \mathcal{M}$. For e_i, the β and ω are related to $\bar{x}(\boldsymbol{\lambda}_i, \epsilon)$ through: $\omega(e_i) = \bar{x}(\beta(e_i), \epsilon)$.

Consider a path $\pi = \langle e_1, \ldots, e_J \rangle \in \Pi$ and its partial path $\pi_j = \langle e_1, \ldots, e_j \rangle$ with $1 \leq j \leq J$. For the path π, assuming $\boldsymbol{\lambda}_\pi$ is defined, the worst case function is $\omega(\pi) = \bar{x}(\beta(\pi), \epsilon)$. In this section we will show that $\boldsymbol{\lambda}_\pi$ can be defined in terms of $\boldsymbol{\lambda}_i$'s, and hence, $\bar{x}(\boldsymbol{\lambda}_\pi, \epsilon)$ can be defined in terms of $\bar{x}_i = \bar{x}(\boldsymbol{\lambda}_i, \epsilon)$.

The quantities x, $\boldsymbol{\lambda}$, or $\bar{x}$ all have the potential to be used as the routing metric. How-

ever, in general, $x_\pi \neq \sum x_i$, $\boldsymbol{\lambda}_\pi \neq \sum \boldsymbol{\lambda}_i$, and $\bar{x}_\pi \neq \sum \bar{x}_i$, where $\sum$ is the standard scalar or vector summation. Let us assume that the addition operation is defined in Λ and $\mathcal{M}$ and is denoted by $\oplus$. If $x_1 = X(e_1)$, $x_2 = X(e_2)$, $\boldsymbol{\lambda}_1 = \beta(e_1)$, $\boldsymbol{\lambda}_2 = \beta(e_2)$, $\bar{x}_1 = \omega(e_1)$, $\bar{x}_2 = \omega(e_2)$, and $\pi = \langle e_1, e_2 \rangle$, then we say $x_\pi = x_1 \oplus x_2$, $\boldsymbol{\lambda}_\pi = \boldsymbol{\lambda}_1 \oplus \boldsymbol{\lambda}_2$, or $\bar{x}_\pi = \bar{x}_1 \oplus \bar{x}_2$.

To avoid producing a routing loop when used with the generalized Dijkstra's algorithm, $\oplus$, Λ and $\mathcal{M}$ have to obey the algebraic properties we discussed in the previous chapter. With $\oplus$, we can now define the path quantities x_π, $\boldsymbol{\lambda}_\pi$, and $\bar{x}_\pi$ in terms of the edge quantities x_i, $\boldsymbol{\lambda}_i$, and $\bar{x}_i$ using a generalized summation: $x_\pi = \bigoplus x_i$, $\boldsymbol{\lambda}_\pi = \bigoplus \boldsymbol{\lambda}_i$, and $\bar{x}_\pi = \bigoplus \bar{x}_i$.

The pairings of Λ and $\mathcal{M}$ with $\oplus$ form algebraic structures which we call the **X**, **B**, and **W** algebras, from the X, β, and ω functions, respectively. Between two nodes, the optimal path π^* is the path with the "shortest" path length from s to d when measured in the **X**, **B** or **W** algebra (or metric). However, having $\oplus$, X, β, and ω is not enough to calculate π^*. We need to compare path lengths. Therefore we need a total order $\preceq$ on Λ and $\mathcal{M}$ to evaluate expressions like $x_\pi \preceq x_{\pi'}$, $\boldsymbol{\lambda}_\pi \preceq \boldsymbol{\lambda}_{\pi'}$, or $\bar{x}_\pi \preceq \bar{x}_{\pi'}$.

Once $\preceq$ is defined, then we can define these optimal values for G:

$$\begin{aligned} x^* &= \min_\pi \{\, x_\pi \mid \pi \in \Pi \,\} \\ \boldsymbol{\lambda}^* &= \min_\pi \{\, \boldsymbol{\lambda}_\pi \mid \pi \in \Pi \,\} \\ \bar{x}^* &= \min_\pi \{\, \bar{x}_\pi = \bar{x}(\boldsymbol{\lambda}_\pi, \epsilon) \mid \pi \in \Pi \,\} \quad . \end{aligned} \tag{4.4}$$

Intuitively, we would like to see that the minima $\boldsymbol{\lambda}^*$ and $\bar{x}^*$ related by the expression: $\bar{x}^* = \bar{x}(\boldsymbol{\lambda}^*, \epsilon)$. Indeed, this is true for the q-ary symmetric channel and q-ary erasure channel networks presented in the previous chapter. In this section, we also introduce a new type of network that consists of constrained AWGN channels and prove that the above also holds.

Next, consider a subset of nodes $\{u_i \in U \subseteq V\}$ that implements (possibly different)

q-ary, erasure- or error-correcting codes of block length n that can correct up to $x_{\max}$ erasures in B_i relative to B_0. By default, we assume that $d \in U$. For packet-based systems, U are nodes that can restore up to $x_{\max}$ lost packets, or packets in any unfavorable states back to the original, favorable states. For a partial path π_j, we can state the following:

$$\begin{aligned} X_{0,j} &= 0 && , v_j \in U \text{ and } X_{0,\bar{j}} \leq x_{\max} \\ &\geq X_{0,\bar{j}} && , v_j \in U \text{ and } X_{0,\bar{j}} > x_{\max} \\ &= X_{0,\bar{j}} && , v_j \in V \backslash U \quad . \end{aligned} \tag{4.5}$$

If $v_j \in U$ and $\bar{B}_j$ does not have more than $x_{\max}$ errors, then v_j restores $\bar{B}_j$ back to B_0. Otherwise, if $x_{\max}$ is exceeded, v_j may or may not increase the number of errors in B. If $v_j \notin U$, it simply repeats the content of $\bar{B}_j$ into B_j.

If $V_\pi \cap U = \varnothing$, then the lengths x_{π_j}, $\boldsymbol{\lambda}_{\pi_j}$ and $\bar{x}_{\pi_j}$ are non-decreasing functions of j — a claim which we will prove later in this section. This means that on any path, the metric $X_{0,j}$ increases as j increases. In this case, reliable communication (when measured in the worst-case metric) becomes very difficult to achieve except when the values $\boldsymbol{\lambda}_i$ (which could represent a very small delay or error probability) are *simultaneously* favorable.

Without the existence of U, this stringent level of QoS is often prohibitively difficult to achieve, forcing the engineers to settle for an unacceptably high catastrophic event probability. However, with U, we later show that with our algorithm, even a *zero* ϵ-worst-case path is achievable.

So far, we have not specified the types of channels we are considering. In this section, we will analyze the two types of channels we have discussed extensively in the previous chapters: the q-ary symmetric channels (q-SC) and the q-ary erasure channels (q-EC). We provide the definitions of the q-SC and the q-EC in the following examples for review.

In addition, we also introduce and define a new type of channel: the AWGN channels with nonnegative mean. This channel is relevant to many network problems where the edge quantities are positive real numbers (the most important of which is delay). We prove that the algebras for these channels satisfy the same set of properties that guarantees the compatibility with the GDA, and the absence of any undesirable routing loop.

Example 1 (q-ary Symmetric Channels)**.** Let Q be the q-ary alphabet $\{0,\ldots,q-1\}$. Suppose that the source s produces n-symbol messages $B \in Q^n$, with $B_l \in Q$, and $l = 1 \ldots n$. Each network link is modeled as a q-ary symmetric channel (q-SC) with symbol error rate p. Let B_i (and $b_i = B_{li}$) denote B (and B_l) as it departs from v_i; and let $\bar{B}_i$ (and $\bar{b}_i = \bar{B}_{li}$) denote B (and B_l) as it leaves e_i. The transition probability defining a q-SC is:

$$P(b_{i+1} \,|\, b_i) = \begin{cases} 1-p & , \; b_i = b_{i+1} \\ p/(q-1) & , \; b_i \neq b_{i+1} \quad . \end{cases} \tag{4.6}$$

The operators $\boldsymbol{v}_i, \boldsymbol{e}_i \in \mathcal{E} : Q^n \to Q^n$ are given by $B_i = \boldsymbol{v}_i(\bar{B}_i)$ and $\bar{B}_{i+1} = \boldsymbol{e}_i(B_i)$. The function X measures the number of errors (Hamming distance) in B_i compared to $B_{i'}$, denoted by $x = X_{i,i'} = X(B_i, B_{i'}) = |\{l \,|\, b_i \neq b_{i'}\}|$. In a q-SC, the scalar parameter that plays the role of $\boldsymbol{\lambda}$ is p, the link symbol error probability. The probability density function $P(x, \boldsymbol{\lambda}) = P(x, p)$ is:

$$P(x,p) = \begin{cases} \binom{n}{x} p^x (1-p)^{n-x} & , \; p \in (0,1) \\ \delta(x) & , \; p = 0 \\ \delta(x-n) & , \; p = 1, \infty \quad . \end{cases} \tag{4.7}$$

The **B** and **W** algebras are such that two adjacent edges e_1 and e_2 with parameters $\boldsymbol{\lambda}_1$ and $\boldsymbol{\lambda}_2$ can be viewed as a single edge with parameter $\boldsymbol{\lambda} = \boldsymbol{\lambda}_1 \oplus \boldsymbol{\lambda}_2 = p_1 \oplus p_2$ defined by

the following equations:

$$p_1 \oplus p_2 = 1 - (1-p_1)(1-p_2) - (p_1 p_2) \,/\, (q-1)$$
$$\bar{x}_1 \oplus \bar{x}_2 = \max\{\, x \mid P(x, p_1 \oplus p_2) \geq \epsilon \,\} \quad . \tag{4.8}$$

If the message B is defined in event- and packet-based scenarios, the letters in the alphabets then correspond to event or packet states. Consequently, the probability densities and the transition probabilities are also defined in terms of these states (instead of symbols).

Example 2 (q-ary Erasure Channels)**.** The symbol $q-1$ in Q is designated as a special *erasure* symbol. Each network link is modeled as a q-ary erasure channel (q-EC) with symbol erasure rate p. The transition probability is given by Equation (4.9) below. Recall that B can be also defined as a data block with n packets of m symbols each, and erasures represent lost packets.

$$P(\, b_{i+1} \,|\, b_i \,) = \begin{cases} 1-p & ,\ b_i = b_{i+1} \\ p & ,\ b_i \neq b_{i+1} = q-1. \end{cases} \tag{4.9}$$

The function X measures the number of erasures in B_i relative to $B_{i'}$, denoted by $x = X_{i,\,i'} = X(B_i, B_{i'}) = |\,\{\, l \,|\, b_i \neq b_{i'} = q-1 \}\,|$. The density function $P(x, \boldsymbol{\lambda})$ is the same as in Example 1, except p is the link symbol erasure probability and the **B** algebra is $\boldsymbol{\lambda} = \boldsymbol{\lambda}_1 \oplus \boldsymbol{\lambda}_2 = p_1 \oplus p_2$ where:

$$p_1 \oplus p_2 = 1 - (1-p_1)(1-p_2) \tag{4.10}$$
$$\bar{x}_1 \oplus \bar{x}_2 = \max\{\, x \mid P(x, p_1 \oplus p_2) \geq \epsilon \,\} \quad . \tag{4.11}$$

Again, as in the q-SC, if the message B is defined as event- and packet-based messages, then the alphabet letters represent specific event or packet states. Accordingly, the prob-

ability densities and the transition probabilities are also defined not in terms of symbols, but rather in terms of these states.

Example 3 (Nonnegative-mean AWGN)**.** In this example, the message is a scalar $B \in \mathbb{R}^+$. On each link, a nonnegative-mean AWGN is added into the message. This noise could represent the amount of nonnegative degradation a packet has experienced so far, such as delay or signal loss. For convention, we assume the source node always transmits the $B_0 = 0$ message.

At each node $v_i \in V_\pi$, two decisions are made. If $x_i = B_i - B_0$ exceeds $x_{\max}$, then the message (or the packet corresponding to the message) is discarded, otherwise it is re-transmitted with the same level of degradation. In other words, nodes do not introduce additional degradation: only edges do. Furthermore, we also assume the existence of the error- (or erasure-) correcting nodes $u_i \in U \subseteq V$ can reset B (and x) back to 0.

Each AWGN channel is characterized by a two-dimensional parameter $\boldsymbol{\lambda} = (\mu \geq 0, \sigma^2)$. The component μ is the mean, and σ^2 the variance of the Gaussian density of the channel. The transition probability and $\bar{x}$ are given by:

$$P(B_{i+1} \,|\, B_i) = P(x; \mu, \sigma^2)[\sqrt{2\pi\sigma^2}]^{-1} \exp\left(\frac{-(x-\mu)^2}{2\sigma^2}\right) \tag{4.12}$$

$$\bar{x} = P^{-1}(\epsilon; \mu, \sigma^2) = \mu + \sqrt{-\sigma^2 \ln(2\pi\epsilon^2\sigma^2)} \tag{4.13}$$

where $x = B_{i+1} - B_i$. The definition of the $\mathbf{B}$ algebra follows from the algebraic property of independent AWGN random variables. Two *independent*, adjacent edges e_1 and e_2 with AWGN parameters $\boldsymbol{\lambda}_1$ and $\boldsymbol{\lambda}_2$ can be combined into a single edge with parameter $\boldsymbol{\lambda} = \boldsymbol{\lambda}_1 \oplus \boldsymbol{\lambda}_2 = \boldsymbol{\lambda}_1 + \boldsymbol{\lambda}_2$. The $+$ operator in the last equation is just the standard vector summation operator.

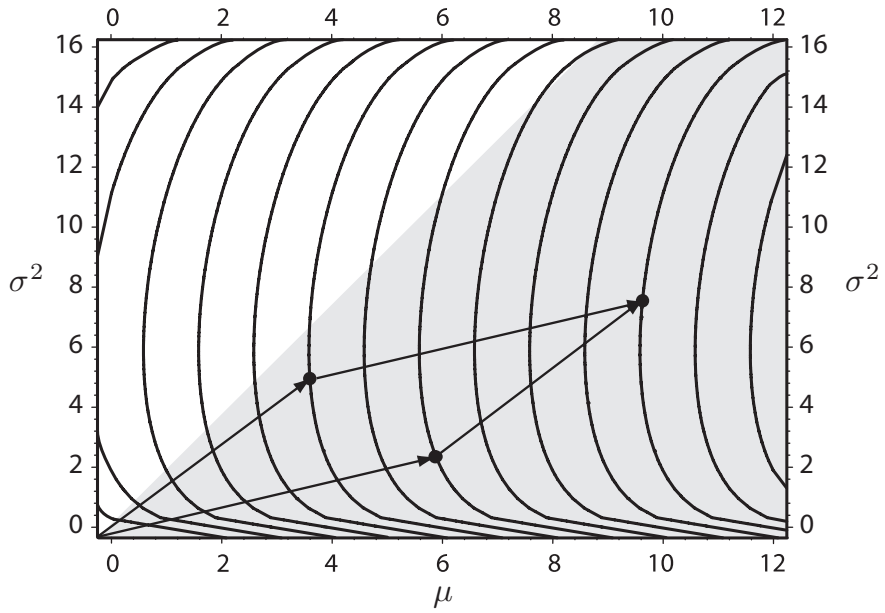


FIGURE 4.3 The metric space Λ (shaded).

Theorem 2. The algebras $\mathbf{B} = (\Lambda, \oplus)$ defined in examples 1–3 and their total orders $\preceq$ satisfy the set of compatibility properties **P** of the GDA.

PROOF: We have proved this theorem for the q-SC and the q-EC. Therefore, we will not repeat the proof here and only discuss the constrained AWGN channel. Before we begin, let us define and use the relational operator $\preceq$ that we can use to compare any $\boldsymbol{\lambda}, \boldsymbol{\lambda}' \in \Lambda$. First, note that $\Lambda \subset \mathbb{R}^{2+}$. Since the set of two-dimensional real numbers $\mathbb{R}^{2+}$ is not a totally ordered set, we must define $\preceq$ in terms of the worst case function $\bar{x}(\boldsymbol{\lambda}, \epsilon)$ as follows:

$$\begin{aligned} &\boldsymbol{\lambda} \prec \boldsymbol{\lambda}' \Leftrightarrow \bar{x}(\boldsymbol{\lambda}, \epsilon) < \bar{x}(\boldsymbol{\lambda}', \epsilon) \text{ or} \\ &\boldsymbol{\lambda} = \boldsymbol{\lambda}' \Leftrightarrow \bar{x}(\boldsymbol{\lambda}, \epsilon) = \bar{x}(\boldsymbol{\lambda}', \epsilon) \quad . \end{aligned} \tag{4.14}$$

Referring to Figure 4.3, we can see that the space Λ is ordered by the isocontour lines defined by the values of $\bar{x}$. Two points that lie on the same contour line are considered equal to each other. Further, later we show that to guarantee closure, the scope of Λ also

has to be restricted to the following region:

$$\Lambda = \left(\mathbb{R}^+ \times [0, \sigma^2_{max}] \cap \left\{\boldsymbol{\lambda} : \frac{\mu}{\sigma^2} \geq \frac{\partial \mu}{\partial \sigma^2}\Big|_{\sigma^2_{max}}\right\}\right) \cup \{\infty\} \quad .$$

In Figure 4.3, the set Λ is depicted by the shaded region. Notice that for a fixed value of σ^2, the slope $\mu' = \partial\mu/\partial\sigma^2$ is equal for all μ along any $\bar{x}$ contour. Further, this slope is maximized at the upper bound of σ^2 defined by σ^2_{max}. If the lines connecting $\boldsymbol{\lambda}$ and $\boldsymbol{\lambda}'$ to the origin have slopes that are larger than the maximum μ', then the sum is guaranteed to lie in a higher contour.

In the following, we will prove that the constrained AWGN satisfies all the compatibility properties for the GDA. We also prove that addition of a vector $\boldsymbol{\lambda}''$ to any pair of $\boldsymbol{\lambda}$ and $\boldsymbol{\lambda}'$ will not change the ordering between them.

P1 Except for closure, the other monoid properties are obvious because $\oplus$ is the standard vector addition. The main reason for Λ not occupying the full non-negative octant, but rather being bounded by $\sigma^2 \leq \sigma^2_{max} = \epsilon^{-2}_{max}/(2\pi)$, is to satisfy the closure property. Inside the shaded area, every vector has a slope of at least μ'. The sum of two such vectors must also have a slope of at least μ'. The value σ^2_{max} is given by:

$$\sigma^2_{max} = \max_{\pi \in \Pi} \left\{ \sum_{i: v_i \in V_\pi} \sigma^2_i \right\} \tag{4.15}$$

This value of σ^2_{max} can be obtained by running a regular DA once on G, with a time complexity cost of $O(V^2)$. If one (or both) of the vectors is ∞, then by the definition of ∞, the $\oplus$ sum must be ∞.

P2 The proof is derived from closure on ∞.

P3 The proof follows from the definition of Λ and $\preceq$.

P4 Both terms in equation (4.13) are minimized when they are zero, i.e., $\mu = 0$ and

either $\sigma^2 = 0$ or $\sigma^2 = \epsilon^{-2}/(2\pi)$. However, since $\boldsymbol{\lambda} = (0,0) \in \Lambda$, then $(0,0)$ is indeed the 0 element in Λ.

P5 From Figure 4.3 and the definition of Λ, the addition of $(\mu, 0)$ followed by $(0, \sigma^2)$ to any two points in Λ preserves their ordering. □

The following proposition proves that the presence of error- and erasure-correcting nodes in the network can have practical benefit for routing information, allowing a very high level of reliability to be achieved by relatively unreliable edges. Mathematically, we say that if the path includes $u \in U$, then for some $j \in [0, J]$, we can have $\beta(\pi_j) = 0$ even with $p_j \neq 0$ for all j.

Proposition 3. If $V_{\pi_J} \cap U = \varnothing$, then $\beta(\pi_j)$ is an non-decreasing function of j. The minimum $\beta(\pi_j) = 0$ is only possible if $p_j = 0$ for all $j = 0 \ldots J$.

PROOF: If we apply the isotonicity property **P5** with $0 = a \prec b = p_j$, and $c = p_{\pi_{j-1}}$, then we will have $\beta(\pi_{j-1}) = p_{\pi_{j-1}} \prec p_{\pi_{j-1}} \oplus p_j = \beta(\pi_j)$. This proves that that $\beta(\pi_j)$ is a non-decreasing function of j. The second part of the proof can be derived directly from **P1**. □

Having proven the GDA compatibility, we will now prove that the $\bar{x}$ values in examples 1–3 can be defined as non-decreasing functions of $\boldsymbol{\lambda}$s. If this is true, then the path with minimum $\boldsymbol{\lambda}$ is also the path with minimum $\bar{x}$.

For AWGN, this monotonicity property automatically follows from its definition of $\boldsymbol{\lambda} \preceq \boldsymbol{\lambda}'$. In contrast, the proof of monotonicity for q-SC and q-EC is quite involved. Luckily, the proof for both channels is identical. First, we claim — and prove — that within an admissible range of $\epsilon \in [0, \bar{\epsilon}]$, the WCE function $\bar{x}(p, \epsilon)$ is a non-decreasing (albeit discontinuous) function of p.

Lemma 4. For a given n and a fixed x, the probability function $P(x, p)$ is maximized at $p = \frac{x}{n}$. Furthermore, $P_* = P(\lfloor \frac{n}{2} \rfloor, \frac{1}{2})$ minimizes $P(x, \frac{x}{n})$ over all possible values of $x \in [0, n]$.

PROOF: Let us start from the definition of $P(x,p)$ in equation (4.7). For a given n, the lemma is true at $x=0$ and $x=n$ because the value of $P(x,p)$ reaches the maximum of $\delta(0)=1$. To prove the lemma for other values of p, we first compute the derivative of $P(x,p)$ with respect to p to find the extrema:

$$\frac{\partial P(x,p)}{\partial p} = P(x,p)\,(\tfrac{x}{p} - \tfrac{n-x}{1-p}) = 0 \quad . \tag{4.16}$$

On the right-hand side of Equation (4.16), the definition of $P(x,p)$ shows that it does not have any root with respect to p. Therefore the root, if there is any, must be contained in the factor $(\frac{x}{p}-\frac{n-x}{1-p})=0$. Indeed, solving the factor for p gives us $p=\frac{x}{n}$, which maximizes the function $P(x,p)$ for any given $x\in(0,1)$.

Having identified the value of p that minimizes $P(x,p)$, the next question is, for $0<x<n$, which x minimizes $P(x,\frac{x}{n})$? Unlike with p, we cannot differentiate $P(x,p)$ with respect to x because it is a discrete variable. We choose not to take the easiest shortcut of approximating $P(x,p)$ with a Gaussian distribution and taking the derivative of this approximation because this approach is only valid for certain values of n and p. Instead, we find the location of the minimum by using the upper and lower bounds of $\binom{n}{x}$ given in [29] :

$$\lambda_{nx} < \binom{n}{x} < \mu_{nx} \tag{4.17}$$

$$\lambda_{nx} = \lambda(n,x) = \xi\big(\tfrac{1}{12n} - \tfrac{1}{12x} - \tfrac{1}{12(n-x)}\big)$$

$$\mu_{nx} = \mu(n,x) = \xi\big(\tfrac{1}{12n} - \tfrac{1}{12x+1} - \tfrac{1}{12(n-x)+1}\big)$$

$$\xi(A) = \frac{e^A\, n^{n+\frac{1}{2}}}{(2\pi)^{\frac{1}{2}}\,(n-x)^{n-x+\frac{1}{2}}\,x^{x+\frac{1}{2}}} \quad .$$

From Equation (4.17), we can then conclude that $P(x,p)$ is bounded from below and above by two continuous and differentiable functions of x.

$$\lambda_{nx}\,p^x\,(1-p)^{n-x} < P(x,p) < \mu_{nx}\,p^x\,(1-p)^{n-x}$$

Since $p = \frac{x}{n}$, we substitute $x = np$ into the equation above and solve the p roots of the p derivatives of both the lower and upper bounds of $P(x, p)$ to find the minima with respect to p. The lower and upper bounds are minimized at $p = \frac{1}{2}$. Since $p = \frac{x}{n}$, then $x = \frac{n}{2}$, and the inequality becomes:

$$\sqrt{\frac{2}{n\pi}}\, e^{-\frac{18n-1}{12n(6n+1)}} < P(\lfloor \tfrac{n}{2} \rfloor, \tfrac{1}{2}) < \sqrt{\frac{2}{n\pi}}\, e^{-\frac{1}{4n}} \quad . \tag{4.18}$$

For large values of n the lower and upper bounds converge. In fact, in (4.17), λ_{nx} converge to μ_{nx} for all x, including at the discrete points $0 < x < n \in \mathbb{N}$. Thus the minima for λ_{nx} and μ_{nx} over the continuous x must also be the minimum for $P(x, x/n)$ over the discrete x, denoted by $P_* = P(\lfloor \frac{n}{2} \rfloor, \frac{1}{2})$. □

Lemma 5. $P(x, p)$ is unimodal over x and p. A function $f(x)$ is *unimodal* over $x \in [a, b]$ if there exists a single value x_0 such that $f(x)$ is monotonically increasing for $x < x_0$ and monotonically decreasing for $x > x_0$.

PROOF: To prove unimodality over x, solve the inequality $P(x, p) < P(x+1, p)$ for x, which gives us $x < np - (1-p)$. Since $0 \leq (1-p) \leq 1$, then $x < \lfloor np \rfloor$. In other words, $x_0 = \lfloor np \rfloor$. From $P(x, p) > P(x+1, p)$ we also conclude $x > x_0$. To prove unimodality over p:

$$P'(x, p) = \frac{\partial}{\partial p} P(x, p) = P(x, p)(\frac{x}{p} - \frac{n-x}{1-p}) \quad .$$

We have calculated that $P'(x, p_0) = 0$ at $p_0 = \frac{x}{n}$. Therefore, p_0 is an extremum and a candidate for a global maximum. Indeed, for $p < p_0$, we can easily show that $P'(x, p) > 0$ and vice versa for $p > p_0$, we have $P'(x, p) < 0$. □

Corrolary 6. The set $\boldsymbol{\epsilon}$ of admissible ϵ is defined by the interval $[0, \bar{\epsilon} = P_*]$.

PROOF: First, recall the definition $\bar{x}(p, \epsilon) = \max\{\, x \mid P(x, p) \geq \epsilon \}$. To ensure that $\bar{x}(p, \epsilon)$ is properly defined for all $p \in \mathcal{P}$, for each value of p we must have $P(x, p) \geq \epsilon$ for some x. This condition is trivially met if $\epsilon \leq 0$ because $P(x, p) \geq 0$. However, we are only

interested in $\epsilon \geq 0$. If $\epsilon > P_*$, then $P(x,p)$ lies completely beneath ϵ and the set $\{\, x \mid P(x,\frac{1}{2}) \geq \epsilon \} = \varnothing$. Consequently, $\bar{x}(p,\frac{1}{2})$ is invalid. Thus, the set of admissible ϵ is given by $[0,P_*]$. □

Consider the equation $P(x,p) = \epsilon$ for some n and ϵ. Define the set of all the roots of this equation by $P = \{\, p \in \mathcal{P} \mid P(x,p) = \epsilon \,\}$. The i-th root of this equation is denoted by $p_i(x,\epsilon) = p_i(x) = P_i$. From Lemma 5, if we can guarantee $\epsilon \in \boldsymbol{\epsilon}$, then there is at least one root, i.e., $P \neq \varnothing$.

In three special cases, P only contains one root. At $x = 0$, it is denoted by $p_0(0) = 0$, and at $x = n$, by $p_1(n) = 1$. Finally, when $\epsilon = P_*$, at the midpoint $x_m = \lfloor \frac{n}{2} \rfloor$ the root is $p_0(x_m) = p_1(x_m) = \frac{1}{2}$. In general, however, there are two distinct roots $p_0(x), p_1(x) \in [0,1]$, with $p_0(x) < \frac{x}{n} < p_1(x)$. If ϵ goes toward 0, then $p_0(x) \to 0$ and $p_1(x) \to 1$, except $p_1(0) = 0$ and $p_0(1) = 1$. If ϵ goes toward P_*, then $p_0(x)$ increases, while $p_1(x)$ decreases.

Define the sets P_0 and P_1, each having the $n+1$ values of the first and second roots $p_0(x)$ and $p_1(x)$ of $\boldsymbol{P}$. Each of them corresponds to the $n+1$ different values of $0 \leq x \leq n$. Using the equation $P(x,p) = \epsilon$, these roots correspond to the sets $x_0(p)$ and $x_1(p)$, for $0 \leq p = \frac{x}{n} \leq n$.

Lemma 7. The roots $p_0(x)$ and $p_1(x)$ are non-decreasing functions of x with $p_0(x) = p_1(x)$ only at $x = 0$ and $x = n$ (or $x = x_m$ for the case where $\epsilon = P_*$).

PROOF: First, observe that $P(x,p) = P(x+1,p)$ only has one root at $p = p^\times = \frac{x+1}{n+1}$ between the maxima of $P(x,p)$ and $P(x+1,p)$, i.e., $\frac{x}{n} < p^\times < \frac{x+1}{n}$. From lemma 5, this implies that if $p < p^\times$ then $P(x,p) > P(x+1,p)$, and if $p > p^\times$, $P(x,p) < P(x+1,p)$. Hence, $p_0(x) \leq p_0(x+1)$ and $p_1(x) \leq p_1(x+1)$. Therefore, both $p_0(x)$ and $p_1(x)$ are non-decreasing functions of x. □

Theorem 8. For q-SCs and q-ECs, the worst case function values $\bar{x}(p,\epsilon) = \max_{x=np} \{x_0(p), x_1(p)\}$ are non-decreasing functions of p.

PROOF: First, the $\max_x$ function is used because it is possible to have identical values of

p that solves $P(x,p)=\epsilon$, that correspond to different values of x. In other words, it is possible to have $p_0(x)=p_0(x')\in P_0$ for $x\neq x'$, where $P(x,p_0(x))=P(x',p_0(x'))=0$. For example, if $\epsilon=0$, $p_0(0)=\cdots=p_0(n-1)=0$. The exact same argument applies to the roots in the set P_1.

It is interesting to note that in our definition of $\bar{x}(p,\epsilon)$, we also have used $\max_x$. The function $\max_x$ in $\bar{x}(p,\epsilon)$ isolates the largest x satisfying $P(x,p)\geq\epsilon$, while $\max_x x_0(p),x_1(p)$ isolates the largest x that satisfies $P(x,p)=\epsilon$. Finally, we claim that $P(x,p)=\epsilon$ implicitly defines a function $p(x)$ that links each value of p to a value of x. The proof for this theorem then follows directly from the monotonicity of $p(x)$ already proven in Lemma 7. □

Theorem 8, proves that the worst case functions $\bar{x}(p,\epsilon)$ of the q-SC and q-EC are non-decreasing functions of p. Therefore, as we stated before, a network path that minimizes p_π also minimizes $\bar{x}$. The constrained AWGN channel also satisfies this due to its definition of $\preceq$ that directly utilizes $\bar{x}$.

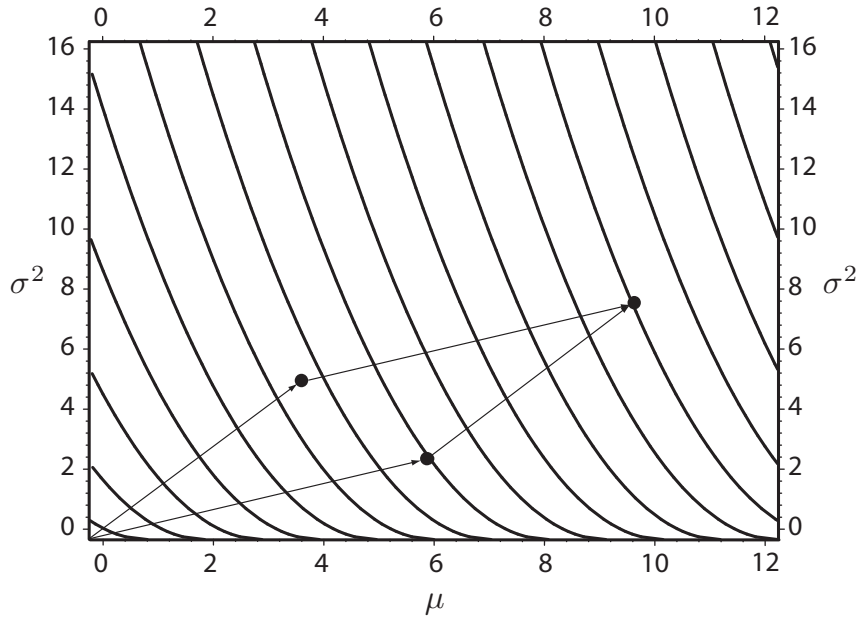


FIGURE 4.4 Isocontour profile of $\bar{x}$ on σ^2 versus μ

The previous results are based on the definition of $\bar{x}(\boldsymbol{\lambda}_i,\epsilon)$ provided in Equation (4.2). In Equation (4.3), we provide an alternative definition of $\bar{x}(\boldsymbol{\lambda}_i,\epsilon)=\max_x\{\,x\mid P(X>$

$x, \boldsymbol{\lambda}_i) \geq \epsilon\,,\ x \in \mathcal{M}\}$. In the remainder of this subsection, we shall prove that even with this alternative definition, the worse case function $\bar{x}$ is also a non-decreasing function of its probability density parameters.

Theorem 9. In examples 1–3, the $\bar{x}$ values defined according to Equation (4.3) are non-decreasing functions of their respective $\boldsymbol{\lambda}$s, which means that the path with minimum $\boldsymbol{\lambda}$ obtained from the GDA is the path with minimum $\bar{x}$.

PROOF: Before we prove the theorem, let us find the definition for the **W** algebra based on the alternative definition of $\bar{x}$, which states $\bar{x}(\boldsymbol{\lambda}_i, \epsilon) = \max_x\{\, x \mid P(X > x, \boldsymbol{\lambda}_i) \geq \epsilon\,,\ x \in \mathcal{M}\}$. Let us define a function F as follows:

$$F(\bar{x}; \mu, \sigma^2) = P(X > x; \mu, \sigma^2) = \frac{1}{2}\mathrm{erfc}\left(\frac{\bar{x} - \mu}{\sqrt{2\sigma^2}}\right) = \epsilon \quad . \tag{4.19}$$

Since in the above equation F is continuous in x, we can invert the function, solve for x, and obtain the following expression for $\bar{x}$ in terms of F:

$$\bar{x} = F^{-1}(\epsilon; \mu, \sigma^2 > 0). \tag{4.20}$$

From this result, the **W** algebra is defined as follows:

$$\bar{x}_1 \oplus \bar{x}_2 = F^{-1}(\epsilon; \mu_1 + \mu_2, \sigma_1^2 + \sigma_2^2). \tag{4.21}$$

Once the $\oplus$ operator is defined for the **W** algebra, we can interpret Figure 4.4. The contour lines are defined by Equation (4.20), and just as before :

$$\boldsymbol{\lambda} \preceq \boldsymbol{\lambda}' \quad \text{iff} \quad \bar{x}(\boldsymbol{\lambda}, \epsilon) \preceq \bar{x}(\boldsymbol{\lambda}', \epsilon). \tag{4.22}$$

This definition of $\preceq$ means that for the constrained AWGN, the theorem is automatically proven. For q-SC (and q-EC, which is identical) we prove the theorem for by analyzing $\bar{x}$

corresponding to the binomial distribution parametrized with $\boldsymbol{\lambda} = (n, p)$. From [40] we have:

$$\begin{aligned} P(X > x, p) &= \sum_{X=x+1}^{n} \binom{n}{X} p^X (1-p)^{n-X} \\ &= I_p(x+1, n-x) \end{aligned} \tag{4.23}$$

where $I_p(x+1, n-x)$ is the *regularized incomplete Beta function*, taking two non-negative integer arguments. This curious function in turn is defined as the ratio of an *incomplete Beta function* $B_p(x+1, n-x)$ and a regular *Beta function* $B(x+1, n-x)$, from which equation the term "regularized" sounds quite similar to "normalized":

$$\begin{aligned} I_p(x+1, n-x) &= \frac{B_p(x+1, n-x)}{B(x+1, n-x)} \\ &= B_p(x+1, n-x) \times \frac{n!}{x!(n-x-1)!} \\ &= (n-x)\binom{n}{x} \int_0^p t^x (1-t)^{n-x-1}\, dt. \end{aligned} \tag{4.24}$$

Since n is constant over the network, for each $0 \leq x \leq n$, the function $I_p(x+1, n-x)$ is really just a function of p. Intuitively, it is obvious that I_p is a non-decreasing function of p. We can prove this by taking the first derivative with respect to p for fixed n and x:

$$\begin{aligned} \frac{\partial I_p}{\partial p} &= (n-x)\binom{n}{x} \frac{\partial}{\partial p} \int_0^p t^x (1-t)^{n-x-1}\, dt \\ &= (n-x)\binom{n}{x} p^x (1-p)^{n-x-1} \geq 0. \end{aligned} \tag{4.25}$$

The last inequality has to be true because $x \leq n$ and $0 \leq p \leq 1$. This means for a given n and x, as we increase p, the value of $P(X > x, p) = I_p(x+1, n-x)$ is non-decreasing. It is important to note that for $0 \leq x < n$, if $p = 0$, then the function $P(X > x, p) = 0$. At the same time, if $p = 1$, then $P(X > x, p) = 1$. For intermediate values of $0 < p < 1$, the

set of values $\boldsymbol{P} = \{P_0, P_1, \ldots, P_n\}$ of $P(X > x, p)$ at $x = 0, 1, \ldots, n$ are *non-increasing* as x increases.

Recall the equation that defines $\bar{x}$ as a function of ϵ and p:

$$\bar{x}(p,\epsilon) = \max_x \{\, x \mid P(X \geq x, p) \geq \epsilon\,,\, 0 \leq x \leq n\}$$

This equation partitions the set $\boldsymbol{P}$ into $\boldsymbol{P}^+ = \{P_i \mid P_i \geq \epsilon, P_i \in \boldsymbol{P}\}$, i.e., the values P_i greater than ϵ, and its complement $\boldsymbol{P}^- = \{P_i \mid P_i < \epsilon, P_i \in \boldsymbol{P}\}$. We define the *index sets* $\mathcal{M}^+ = \{i \mid P_i \in \boldsymbol{P}^+, i \in \mathcal{M}\}$ and $\mathcal{M}^- = \{i \mid P_i \in \boldsymbol{P}^-, i \in \mathcal{M}\}$. Then $\bar{x}(p,\epsilon)$ is the maximum value of $i \in \mathcal{M}^+$ bordering $\mathcal{M}^-$.

$$\bar{x}(p,\epsilon) = \max\{\, i \mid i \in \mathcal{M}^+\}$$

However, from equation (4.25), the values P_i are non-decreasing functions of p (while simultaneously non-increasing with respect to i for each value of p).

Consequently, as p increases, the cardinality $|\boldsymbol{P}^+|$ of $\boldsymbol{P}$ increases. Since $|\mathcal{M}^+| = |\boldsymbol{P}^+|$, so does the cardinality of $\mathcal{M}^+$. However, since both $\mathcal{M}^+$ and $\mathcal{M}^-$ are two contiguous partitions of $\mathcal{M}$, this implies $\bar{x}$ (the maximum element of $\mathcal{M}^+$) must increase as a function of p. □

By proving Theorems 8 and 9, i.e., by proving that the values of $\bar{x}$ are non-decreasing functions of their $\boldsymbol{\lambda}$s, we can compute the path with minimum worst case error without using the function $\omega(\cdot)$ and the $\oplus$ operator from the **W** algebra directly. Instead, we can perform the operations in the **B** algebra.

4.2.2 Algorithm

In this subsection, we will show how to compute the optimal network path in the presence of erasure- and error-correcting nodes U by using the GAS-STATION algorithm and the $\oplus$ and $\preceq$ operators of the **B** algebra.

Theorem 10. *A path π^* is the minimum WCE path iff it solves the shortest-path problem (SPP) given by $G' = (V', E')$, where $V' = \{s\} \cup U$. An edge connecting two nodes $v_1, v_2 \in V'$ represents the shortest-path in $\Phi(v_1, v_2)$, i.e., $E' = \{ \mathit{argmin}_{\phi} \{ \omega(\phi) \mid \phi \in \Phi(v_1, v_2) \} \mid v_1, v_2 \in V' \}$.*

PROOF: Suppose π^* contains $n+1$ segments ϕ_i connecting the nodes in $V'' = \{s, U_{\pi^*}, d\}$, where $U_{\pi^*} = U \cap V_{\pi^*}$. In segment notation, π^* is denoted by $s \rightsquigarrow u_1 \rightsquigarrow \cdots \rightsquigarrow u_j \rightsquigarrow d$, with $\{u_i\} = U_{\pi^*}$, and $0 \leq j \leq |U|$. Then ϕ_i must be the shortest feasible paths between adjacent nodes in V'', and U_{π^*} must be the set that minimizes $\sum \beta(\phi_i)$. Otherwise, a better path ξ^* can be obtained by modifying ϕ_i or U_{π^*}, contradicting the claim that π^* is optimal.

For the forward proof, note that $V'' = \{s, d, U_{\pi^*}\} \subseteq V'$. Further, since each ϕ_i is a shortest-path between nodes in V', then it has a representation in E', i.e., $\phi_i \in E'$. Therefore π^* is the solution to the SPP given by $G' = (V', E')$.

For the reverse proof, suppose ξ^* is the SPP solution but is not the minimum $\bar{x}$ path π^*. From the forward proof, if π^* minimizes $\bar{x}$, then it also solves the SPP, thus $\omega(\pi^*) \leq \omega(\xi^*)$. However, if $\omega(\pi^*) < \omega(\xi^*)$, then ξ^* is not the SPP solution. This is obviously a contradiction to our original assumptions. Hence, $\omega(\pi^*) = \omega(\xi^*)$, and if path lengths are unique, $\pi^* = \xi^*$. Thus, the SPP solution π^* is also the minimum $\bar{x}$ path. □

Theorem 10 essentially proves the correctness of the optimal worst-case routing algorithm listed below. This algorithm is exactly the GAS STATION algorithm we discussed before. Although in this section $\bar{x}$ is linked to the worst-case value of x for the constrained AWGN channel, other worst-case metrics can be used.

1: **procedure** GAS STATION (G, $\boldsymbol{m}$, s)
2: $\quad E \mathrel{\backslash}= \{ e \in E \mid \omega(e) > x_{\max} \}$
3: $\quad V \mathrel{\backslash}= \{ v \in V \mid \deg(v) = 0 \}$
4: $\quad$ **for** $v_1 \in V'$ **do**

5: $\quad\quad$ SP_1 = GDA (G, $\boldsymbol{m}$, v_1)

6: $\quad\quad$ **for** $v_2 \in V' \neq v_1$ **do**

7: $\quad\quad\quad$ $E' = E' \cup \langle v_1, v_2 \rangle$

8: $\quad\quad$ **end for**

9: $\quad$ **end for**

10: $\quad$ $E' \setminus= \{\, e' = (v_1, v_2) \in E' \mid e' \notin \Phi(v_1, v_2) \,\}$

11: $\quad$ $V' \setminus= \{\, v' \in V' \mid \deg(v') = 0 \,\}$

12: $\quad$ SP_2 = GDA (G', $\boldsymbol{m}$, s)

13: **end procedure**

Hence, to find the minimum WCE path we first compute the minimum WCE path for each pair of nodes in V'. These minimum paths are then converted into edges in E', connecting the nodes in V'. The overall minimum WCE path is then computed from these edges using the GDA.

On lines 2 and 3, the algorithm prunes all infeasible edges. Then, on line 5, it runs the GDA on all $v_1 \in V'$, every time producing a shortest-path tree SP_1 rooted at v_1. On line 7, the edges connecting v_1 and $v_2 \in V'$ are added into E' based on SP_1. This finishes the first stage and starts the second stage. On lines 8 and 9, the infeasible edges in E' are pruned, and any isolated nodes in V' are removed. On line 10, π^* is finally calculated using GDA. □

Let us denote $|V'|$ by α. The first stage produces a complete graph with α nodes and $\alpha(\alpha-1)$ directed edges by executing the GDA α times on line 5, and thus has a time complexity of $O(\alpha V^2)$ (if the GDA is implemented using Fibonacci heap, then its complexity could reach $O(V \log V + E)$ [4]). Lines 8 and 9 search linearly over them with $O(\alpha^2)$ time complexity. The GDA on line 10 has a time complexity $O(\alpha^2)$. Hence, overall time complexity can be bounded by $O(\alpha V^2 + \alpha^2)$, or conservatively, $O(V^3)$.

4.2.3 Conclusion

In this section, we consider the problem of finding the path with minimum (or zero) worst possible number of erasures in mission-critical communication networks. We make the assumptions that some network nodes are capable of correcting up to a maximum number of $x_{\max}$ erasures in a block of n symbols, and that the nodes are connected by edges that represent probability density parameters from important channel models.

We introduce the Worst-Case Erasure (WCE) metric and an accompanying algebra that allows us to compute the path WCE length from its edge lengths. The metric-algebra pair is then used to optimize the network QoS in the WCE metric using a Generalized Dijkstra's algorithm in the worst-case time complexity of $O(V^3)$, where V is the number of nodes in the network.

We provide three examples where the links are modeled as q-ary symmetric channels, q-ary erasure channels, and nonnegative mean AWGN channels, each with its own edge metric that is nothing more than the parameters of its transmission failure probability density.

By letting the metric be density parameters, we compute any path length by an appropriate combination of its edge lengths according to the laws of probability, thus fully preserving the stochastic nature of the routing problem (instead of simply reducing the stochastic edge weights to proxy deterministic values). For a given level of "possibility threshold" ϵ, each edge density corresponds to a unique worst case value $\bar{x}$. We showed that edge worst case value can be combined into path worst case values, and that the minimum $\bar{x}^*$ of such values over a network can be computed using the GDA.

The pair $\bar{x}^*$ and ϵ allow us to compare the worst-case network routing performance. Given a constant benchmark worst-case value $\bar{x}$ for all the networks $\{G_i\}$ under evaluation, we can solve for the ϵ_i value for each network G_i. The network G^* with the lowest value of ϵ^* is the network with the best worst-case routing performance.

Alternatively, the ϵ value can be fixed for all the networks $\{G_i\}$ under evaluation. The

worst-case values $\bar{x}_i$ can then be compared among these networks to select the network G^* with the best worst-case performance, i.e., the smallest failure rate $\bar{x}_i$ among all the networks G_i.

Future research can explore the issues of (1) whether the approach outlined in this paper can be generalized to other types of distributions and scenarios, (2) the incorporation of the algorithm presented here into existing networking protocols, as well as (3) simulation and experimental verification of the algorithm on standard network models. Another interesting application would be to use the worst-case performance evaluation method on actual or planned mission-critical sensor network projects.

BIBLIOGRAPHY

[1] Y.P. Aneja, V. Aggarwal, K.P.K. Nair, "Shortest chain subject to side constraints", *Networks*, vol. 13, pp. 295–302, 1983.

[2] C. Barnhart, N. Boland, L. Clarke, E.L. Johnson, G.L. Nemhauser, R.G. Shenoi, "Flight string models for aircraft fleeting and routing," *Transport. Sci.*, vol. 32, pp. 208–220, 1998.

[3] J.E. Beasley, N. Christofides, "An algorithm for the resource constrained shortest-path problem," *Networks*, vol. 19, pp. 379–394, 1989.

[4] T.H. Cormen, C.E. Leiserson, R.L. Rivest, *Introduction to Algorithms,* MIT Press, Cambridge MA and McGraw-Hill, New York, 1990.

[5] M. Desrochers, J. Desrosiers, M. Solomon, "A new optimization algorithm for the vehicle routing problem with time windows," *Oper. Res.*, vol. 40, 342–354, 1992.

[6] M. Dror, "Note on the complexity of the shortest-path models for column generation in VRPTW," *Oper. Res.*, vol. 42, no. 5, pp. 977–978, 1994.

[7] D. Feillet, P. Dejax, M. Gendreau, C. Gueguen, "An exact algorithm for the elementary shortest-path problem with resource constraints: application to some vehicle

routing problems," *Networks*, vol. 43, no. 3, 216–229, 2004.

[8] G.W. Graves, R.D. McBride, I. Gershkoff, D. Anderson, D. Mahidhara, "Flight crew scheduling," *Manage. Sci.*, vol. 39, no. 6, 657–682, 1993.

[9] N. Kohl, "Exact methods for time constrained routing and related scheduling problems," Ph.D. Thesis, Institute of Mathematical Modelling, Technical University of Denmark, Ph.D. Dissertation no. 16, 1995.

[10] N. Kohl, J. Desrosiers, O.B.G. Madsen, M. Solomon, F. Soumis, "2-path cuts for the vehicle routing problem with time windows," *Transport. Sci.*, vol. 33, pp. 101–116, 1999.

[11] D. Kohler, A.A. Elimam, "Engineering application of the resource constrained shortest-path problem," *Eur. J. Oper. Res.*, vol. 103, pp. 426–438, 1997.

[12] S. Lavoie, M. Minoux, E. Ordier, "A new approach for crew pairing problems by column generation with an application to air transportation," *Eur. J. Oper. Res.*, vol. 35, pp. 45–58, 1988.

[13] M.M. Solomon, "Vehicle routing and scheduling with time window constraints: models and algorithms," Ph.D. Thesis, Dept. of Decision Sciences, University of Pennsylvania, 1983.

[14] J.L. Sobrinho, "Algebra and algorithms for QoS path computation and hop-by-hop routing in the Internet," *IEEE/ACM Transaction on Networking.*, vol. 10, pp. 541–550, August 2002.

[15] B. Kurceren, et al., "A joint source-channel coding approach to network transport on digital video," *Proceeding of INFOCOM 2000*, vol. 2, pp. 717–726, 2000.

[16] A.M. Michelson, et al., "Reliable ATM transmission of multimedia in wireless networks," *Universal Personal Communications, 1998*, vol. 2, pp. 1021–1027, 1998.

[17] N. Nikaein, et al., "MA-FEC: a QoS-based adaptive FEC for multicast communication in wireless networks," *Proceeding of IEEE ICC, 2000,* vol. 2, pp. 954–958, 2000.

[18] G.J. Wang, et al., "Channel-adaptive error protection for scalable video over channels with bit errors and packet erasures," *Proceeding of ISCAS 2002*, vol. 2, pp. 712–715, 2002.

[19] P. Barsocchi, et al., "Experimental Results with Forward Erasure Correction and Real Video Streaming in Hybrid Wireless Networks," *2nd International Symposium on Wireless Communication Systems*, pp. 662–666, 2005.

[20] C. Huitema, "The case for packet level FEC," in *Proceedings of IFIP PfHSN 1996*.

[21] J. Nonnenmacher, et al., "Parity-based loss recovery for reliable multicast transmission," *IEEE/ACM Transactions on Networking*, vol. 6, no. 4, pp. 349–361, 1998.

[22] L. Lancerica, et al. "Evaluation of content-access QoS for various dissemination strategies in peer to peer networks," *Proceeding of IEEE ICON 2003*, pp. 337–342, 2003.

[23] J. Byers, et al., "Informed Content Delivery Across Adaptive Overlay Networks," *Proceedings of ACM SIGCOMM*, 2002.

[24] D. Kostic, et al., "Bullet: High Bandwidth Data Dissemination Using an Overlay Mesh," *Proc. of ACM SOSP, 2003*.

[25] M. Wang, et al., "A High-Throughput Overlay Multicast Infrastructure with Network Coding," *Proceedings of IFIP IWQOS 2005* pp. 37–53.

[26] Q. Du, et al., "Adaptive Low-Complexity Erasure-Correcting Code-Based Protocols for QoS-Driven Mobile Multicast Services," *Proceeding of QShine 2005*, pp. 29–29, 2005.

[27] Q. Li, et al., "Providing adaptive QoS to layered video over wireless local area networks through real-time retry limit adaptation," *IEEE Transactions on Multimedia*, vol. 6, no. 2, pp. 278–290, 2004.

[28] J. Li, et al., "A practical proactive hybrid FEC/ARQ technique for reliable multisession multicast communication," *Proceeding of IEEE ICECS 2003*, vol. 2, pp. 519–522, 2003.

[29] P. Stanica, "Good Lower and Upper Bounds on Binomial Coefficients," *Journal of Inequalities in Pure and Applied Mathematics*, vol. 2, no. 3, article 30, 2001.

[30] "ANSS — Advanced National Seismic System," web site address at *http://earthquake.usgs.gov/anss/*.

[31] "PTWC — Pacific Tsunami Warning Center," web site address at *http://www.prh.noaa.gov/pr/ptwc/*.

[32] E. Royer, C. Toh, "A review of current routing protocols for ad-hoc mobile wireless networks," *IEEE Personal Communications Magazine*, vol. 6, no. 2, pp. 46–55, April 1999.

[33] A. Ahmed, H. Shi, Y. Shang, "A survey on network protocols for wireless sensor networks," *Proceeding of the International Conference on Information Technology*, pp. 301–305, August 2003.

[34] C. Intanagonwiwat, R. Govindan, D. Estrin, "Direct diffusion: a scalable and robust communication paradigm for sensor networks," *Proceedings of MobiCom*, August, 2000.

[35] W. Heinzelman, A. Chandrakasan, H. Balakrishnan, "An application-specific protocol architecture for wireless microsensor networks," *IEEE Transactions on Wireless Communications*, vol. 1, no. 4, pp. 660–670, October 2002.

[36] B. Karp, H. T. Kung, "GPSR: greedy perimeter stateless routing for wireless networks," *Proceedings of the MobiCom*, August, 2000.

[37] Y. Xue, K. Nahrstedt, "Fault-Tolerant Routing in Mobile Ad Hoc Networks," *Proceedings of IEEE Wireless Communication and Networking Conference*, pp. 1174–1179, March, 2003.

[38] V. Park, M. Corson, "A highly adaptive distributed routing algorithm for mobile wireless networks," *Proceedings of INFOCOM'97*, pp. 1405–1413, April 1997.

[39] X. Huang, J. Deng, J. Ma, Z. Wu, "Fault-Tolerant Routing For Wireless Sensor Grid Networks", *Proceedings of IEEE Sensors Applications Symposium 2006*, February 2006.

[40] M. Abramowitz, I. Stegun, *Handbook of mathematical functions,* National Bureau of Standards, Applied Mathematics Series – 55, December 1972.

5

Message-Passing Algorithms in Network Routing

THE most ubiquitous interior network routing protocols on the Internet are the Open Shortest Path First (OSPF) and the Optimized Link State Routing (OLSR). Both use Dijkstra's shortest-path algorithm (DA) to compute the best path for unicast communication between a pair of source and destination nodes within an autonomous system (AS), which is defined as a federation of computers (routers) managed under a common administrative authority.

Both protocols assume that all routers have identical copies of the complete routing table that describes the network topology and network link cost. Since the Internet is a dynamic network with links that change over time, the routers keep their routing tables synchronized by using message-passing algorithms that "flood" the network with identical copies of the routing table.

In the first section, we discuss the tasks of identifying the set of assumptions that guarantees message synchronization between a pair of nodes in a network where flooding is used and where connectivity changes are asynchronous. We call this problem the Highly Dynamic Network problem.

Next, we generalize the flooding algorithm by using a different model of uncertainty. Network nodes propagate information by sending "messengers" to their neighbors. Each

messenger randomly chooses which neighbor to visit, and in turn each node counts the number of visiting messengers. If the number is greater than a threshold, more messengers are sent, and so on. The problem is to identify the set of conditions under which an accurate message synchronization is met. We call this problem the Chinese Generals Problem.

In the final section, we discuss how threshold-based message passing algorithms can be used for decoding generalized LDPC codes that does not utilize linear finite field algebraic operators, but instead integer threshold functions for error detection and correction. As an example, we cast the Sudoku puzzle as a generalized LDPC code and present a recursive algorithm to decode it. The threshold function can be applied to other types of codes.

Message-passing algorithms are among the most important network communication algorithms. Over the years, a wide variety of message-passing algorithms have been invented for different types of communication networks [1–10]. Here, we focus on message-passing algorithms for "mobile" networks.

First, let us define what we mean by "mobile". Each node and link in a communication network is associated with (or assigned to) a specific combination of operating parameters that may include physical location, frequency, antenna direction, transceiver power, etc. Each unique combination in turn corresponds to a specific coordinate in a multidimensional parameter space.

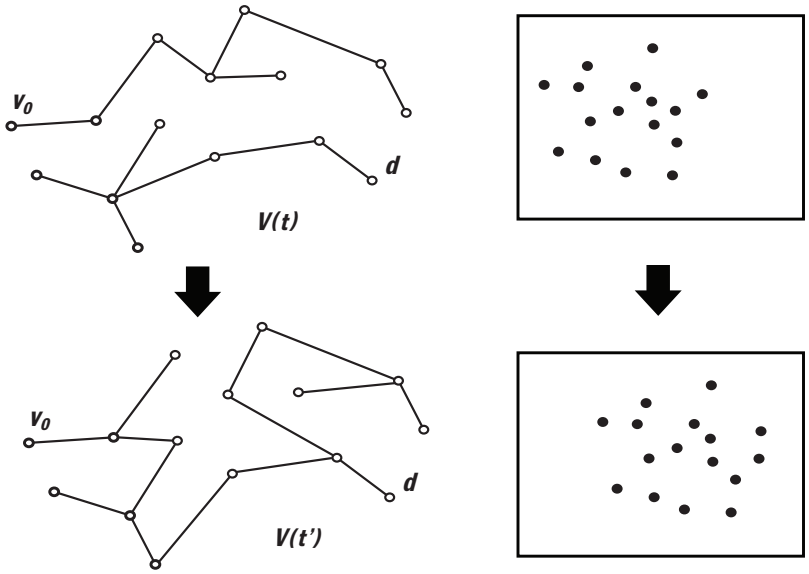


FIGURE 5.1 Change in network topology and parameter space coordinates

During its operation, the parameters of a particular node (or link) do not stay constant. For example, battery power fluctuates or decreases over time; different frequency bands are used to reduce interference; noise increases, etc. These changes move the node (or link) to a new coordinate in the parameter space. A "mobile" network is defined as a network containing nodes and links whose coordinates in the parameter space vary as

functions of time.

Conceptually, mobile networks are very closely related to dynamic networks. A dynamic network is defined as a network that has a time-varying topology. Changes in node and link operating parameters can lead to changes in the network topology, and any change in network topology must be due to a change in the operating parameter(s) of the related node(s) or link(s). For example, Figure 5.1 shows a change in the network topology and the corresponding change of the node coordinates in the two dimensional parameter space. In this sense, the network as a whole has also "moved".

In practice, mobile networks are also most commonly implemented as wireless networks. Therefore, in this section we use the terms mobile network, dynamic network, and wireless network interchangeably. In addition, since communication networks are abstractly represented by graphs, the terms edge and link are also used interchangeably.

Dynamic networks include a vast and diverse spectrum of networks with different node and edge mobility models. On one extreme of the spectrum, we have the static network mobility model. In this mobility model, communication nodes and links move so slowly in the parameter space that they are often considered to be constant. On the other extreme of the spectrum, we have various stochastic network mobility models. In these models, communication nodes and links move almost unpredictably. The only reasonable way to describe a network's motion is in terms of its stochastic parameters.

We define the *highly dynamic network* (HDN) as a mobility model that lies somewhere between these extremes, and patterned after the model presented in [1]. Just like other mobility models, the HDN is also defined by a set of assumptions, which we shall describe in detail later in this section. Of interest to us is what we call the *highly dynamic network problem*: Is there a reliable message-passing algorithm for a HDN with node and edge mobility?

The terms node mobility and edge mobility are defined as the range of ability over which nodes and links can enter and leave the network and modify the topology. We

call a node (or an edge) a *mobile* or a *dynamic* node if it has node mobility, otherwise, it is *fixed*. A reliable algorithm guarantees that a message from the source reaches the destination nodes in a polynomially bounded time period, i.e., it satisfies the correctness and termination criteria.

Previous mobility models defined in [2, 3, 5, 6, 8, 9] use different sets of assumptions that are more restricted than the set of assumptions we use here to define the HDN. In general, message-passing algorithms defined in those papers work only if the accompanying assumptions are also met. Many of these assumptions are too restrictive for us, but the COUNTERFLOODING ALGORITHM (CFA) found in [1] has many useful features for our HDN problem.

The CFA requires the network to have full connectivity at all time, and limits the rate of change in coordinate of any of its nodes and edges from exceeding a prespecified maximum speed of message transmission. In addition, it is guaranteed to work only in networks with dynamic edges and fixed nodes. In contrast, the HDN features dynamic nodes and dynamic edges. The good news is that we found a way to extend the CFA to solve the HDN Problem.

The approach to our solution is as follows. First, we provide an alternative proof to the CFA. In our proof, we relax the various assumptions used in the CFA. Most importantly we remove the stringent requirement that forces the network to stay connected at all times. Second, we prove the correctness and termination of our extended algorithm for the HDN.

The theoretical results from this section prove that the explicit set of assumptions used in defining the HDN is also required to guarantee correctness and termination of the modified flooding algorithm based on the CFA. This complements the results from [7, 10], which report empirical success of the flooding algorithms called probabilistic routing and epidemic routing in the *intermittently connected network* models similar to the model used in [1].

5.1.1 Formulation

Consider a set V that contains all the network nodes. In practice, the number of nodes $|V|$ might only be known up to an upper bound $\bar{V}$. For wireless networks, each node could represent a wireless device that is characterized by a generalized coordinate specifying the device's identity, physical location, antenna directivity, transceiver power, protocol, etc. These dynamic nodes are allowed to "move" in their generalized coordinate space. As a result of this movement, the network itself is changing.

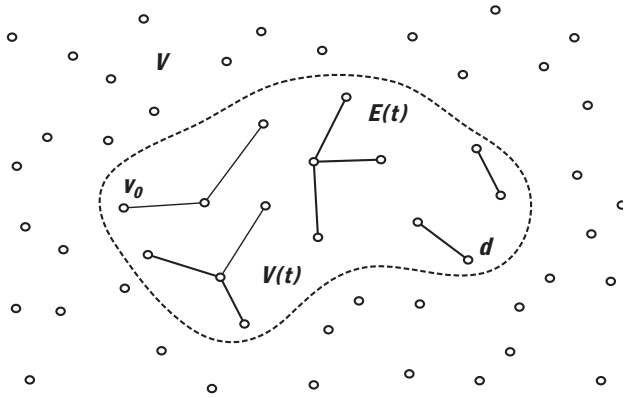


FIGURE 5.2 The sets V, $V(t)$, and $E(t)$

Denote by v_0 the "source" node from which information (or message) is assumed to be generated, and by V_d the set of all destination nodes d. Denote by $V(t)$ or V_t the subset of V containing all non-isolated nodes in V — those nodes with degree greater than zero — at a given time t. The nodes v_1 and $v_2 \in V(t)$ are *connected* at time t if a hypothetical message with infinite speed launched from either node at time t can reach the other. Lack of connection between two nodes may be due to incompatible protocol, path obstruction, insufficient power, internal filtering, or signal masking.

The connectivity between two nodes v_1 and v_2 at time t is represented by an edge $e(1,2,t)$, also denoted by $e_{12}(t)$, $e(t)$, or simply e. Denote by E, E_t, or $E(t)$ the set of all edges e at time t, and by $N(v,t)$ the set of all neighboring nodes to v at time t. At any given time t, the network is represented by a graph G that consists of nodes $V(t)$ and edges $E(t)$. We can also denote G by $G(t)$, and $G(V_t, E_t)$. Figure 5.2 shows the sets V,

$V(t)$, and $E(t)$ for $G(t)$.

We can also think of the network represented by a graph $G(t)$ as a dynamic system — not to be confused with *dynamical* system — with the nodes and edges playing the role of its "moving parts." From the two types of moving parts, $G(t)$ produces two types of discrete events:

(a) The *connectivity-driven* events $c = c(v,t)$ are generated (and received) by the node set v whenever they gain or lose edges. For now, let us assume that connectivity change related to v can be detected instantaneously. The realistic case with delay is discussed later. Denote by $C_v = \{\ t \mid N(v,t-\epsilon) \neq N(v,t+\epsilon)\ \}$ the set of all events $c(v,t)$ at a given v. Thus, C_v contains all local vertex events occuring at v. The set $C = \bigcup_v C_v$ then denotes the global set of network events. The "creation" event of $G(t)$ is formally denoted by the set $c(v,0)$.

(b) The *message-driven* events $m(v,t)$ are triggered at v when a new message m from $v' \neq v$ is received, where it is understood that $v' \in N(v,t' < t)$, which implies a finite message propagation time. In (a), nodes are described as network moving parts by virtue of their ability to move in and out of the network. Messages are also network moving parts in a sense that they have the ability to move between the nodes in G. Denote by M_v the set of all events $m(v,t)$ at v, and by $M = \bigcup_v M_v$ the global set of message-driven events.

The preceding discussion establishes that $G(t)$ consists of two major groups of dynamic components: $V(t)$ and $E(t)$. In turn, the set $V(t)$ itself consists of two complementary sets. The first set contains all nodes that are already visited by (and thus have stored) the message from the source node. Let us denote this set by $V_\bullet(t)$. The second set contains all nodes that have not yet been visited by, and thus are still waiting for the message from the source node. This set is denoted by $V_\circ(t)$. The two sets are shown in Figure 5.3. Obviously, at $t = 0$, $V_\bullet(0) = \{v_0\}$, and $V_\circ(0) = V \backslash \{v_0\}$.

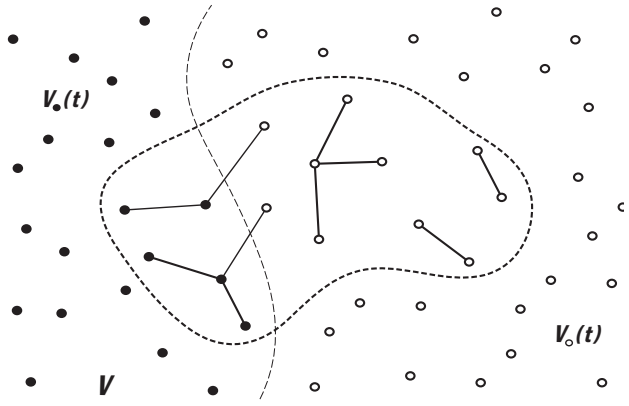


FIGURE 5.3 The sets $V_\bullet(t)$ and $V_\circ(t)$

Depending on whether a node v is in $V_\bullet(t)$ or $V_\circ(t)$, it processes the incoming events using either one of the following schemes. If the event is a connectivity-driven, i.e., the event is in $c(v,t)$, then the node $v \in V_\bullet(t)$ will retransmit the message m that is already in its memory to its neighbors $N(v,t)$. On the other hand, if the event is message-driven, i.e., the event is in $m(v,t)$, then v will retransmit m to $N(v,t)$, and stores m in its own memory. When $v \in V_\bullet(t)$ leaves the network at some time $t > t'$, that is, $v \notin V(t' > t)$, the message m stays in v's memory: $v \in V_\bullet(t')$.

With this scheme, a message m from the source node v_0 would "flood" the network and reach other nodes, including the destination nodes $d \in V_d$. The eventual arrival of m at V_d depends on: (a) the separation between $d \in V_d$ and v_0 in terms of time, (b) the rate at which both $E(t)$ and $V(t)$ are changing, and (c) the speed at which m travels from $v' \in V(t)$ to $N(v',t)$.

To ensure arrival of m at V_d within a finite amount of time, we need to impose certain restrictions on $G(t)$. These restrictions will also play the role as the correctness and termination criteria for a reliable message-passing algorithm on a dynamic network. Let us begin with the assumptions required by the CFA before discussing the assumptions used by the HDN:

1. Edge mobility is supported, but not node mobility, i.e., $V(t) = V$,
2. $E(t)$ is such that $G(t)$ stays *fully* connected at all times,

3. Any change $\Delta E(t)$ in $E(t)$ will instantaneously trigger connectivity-driven events at all the nodes affected,
4. Adjacent nodes can pass messages to each other in less than τ,
5. Two consecutive events in C_v are separated in time by at least τ,
6. Nodes always store the messages they receive, and
7. Nodes know the value of $\bar{V}$.

For the HDN, assumptions 4, 5, and 7 remain the same, but the other assumptions are relaxed, and assumption 6 is made explicit:

1. Both node and edge mobility are supported
2. $E(t)$ is such that each node d in V_d stays connected to at least one node in $V_\bullet(t)$ at all times,
3. Events in C_v may be delayed from their corresponding $\Delta E(t)$,
4. Adjacent nodes can pass messages t each other in less than τ,
5. Two consecutive events in C_v are separated by at least $\max(\tau, \tau_c)$,
6. Nodes always store the messages they receive for at least τ_m, and
7. Nodes know the value of $\bar{V}$

The HDN assumption 3 generalizes that of the CFAs by introducing a delay between the actual connectivity change $\Delta E(t)$ and its corresponding event $c(v,t)$. Without this delay, a node needs to continuously broadcast its presence, resulting in poor energy efficiency. We assume that the nodes beacon their presence periodically every τ_c. This signal can be used to detect disconnections. In subsection 5.1.2, we will also derive the minimum message retention time τ_m required for reliable message-passing.

To prevent uncontrolled and unterminated message replication, the receiving node masks the incoming beacon signal if it comes from a neighboring node that already sent the same message, which implies an additional memory requirement in the order of $\bar{V}$. Also note that, HDN assumption 5 is different from that of the CFAs due to the detection delay.

5.1.2 *The* COUNTER FLOODING *Algorithm*

We will first describe, and then prove that the CFA is a reliable algorithm that can also solve our HDN problem. On page 108, we have begun the discussion — in a narrative form — of the *flooding* behavior of the CFA that depends on the trigger event. We have not discussed the *counter* variable k that is used to control the extent of message propagation. This is included in the full listing of the CFA provided below:

1: **if** message $m(v,t)$ is received for the first time **then**
2: Broadcast message m
3: $k \leftarrow 0$
4: **end if**
5: **if** event $c(v,t)$ is received **then**
6: **if** $k < 2\bar{V}$ **then**
7: Broadcast message m
8: $k \leftarrow k+1$
9: **end if**
10: **end if**

We then have proven that when the CFA is run under the HDN assumptions, the number of nodes visited by the message at time $t+2\tau$ is always strictly larger (by at least one node) than the number of nodes with message at t, as long as there is at least one destination node without the message.

Lemma 11. $|V_\bullet(t+2\tau)| > |V_\bullet(t)|$ *when* $V_d \cap V_\circ(t) \neq \varnothing$.

PROOF: Consider one of the destination nodes $d \in V_d$. Denote by $V_\bullet(d,t)$ all nodes in $V_\bullet(t)$ connected (not necessarily directly) to d. By assumption 2, there must be at least one such node: $V_\bullet(d,t) \neq \varnothing$ at all time t. Refer to Figure 5.4. Assumption 2 also implies that as long as $d \in V_\circ(t)$, there must be at least one node $u \in V_\bullet(d,t)$ that is adjacent to some nodes $v \in V_\circ(t)$ (to see why this is true, just set $v = d$). Denote by $V_\circ(d,t)$ all

such adjacent nodes in $V_\circ(t)$. Members of $V_\bullet(d,t)$ and $V_\circ(d,t)$ are marked by $\uparrow$ and $\downarrow$, respectively.

Denote by $E(d,t)$ the set of all edges connecting the nodes $u \in V_\bullet(d,t)$ to the nodes $v \in V_\circ(d,t)$. In the Figure, these edges cross the dot-dashed line. Consider $e \in E(d,t)$ connecting u and v. Two types of events trigger a message transmission from u to v: (a) when u detects the presence of v at time t_c, the event $c(u,t_c) \in C_u$ is generated, and (b) when a new message m reaches u at time t_m and converts u into the set $V_\bullet(d,t_m)$, the event $m(u,t_m)$ is generated. If a message has been previously received, no event is generated.

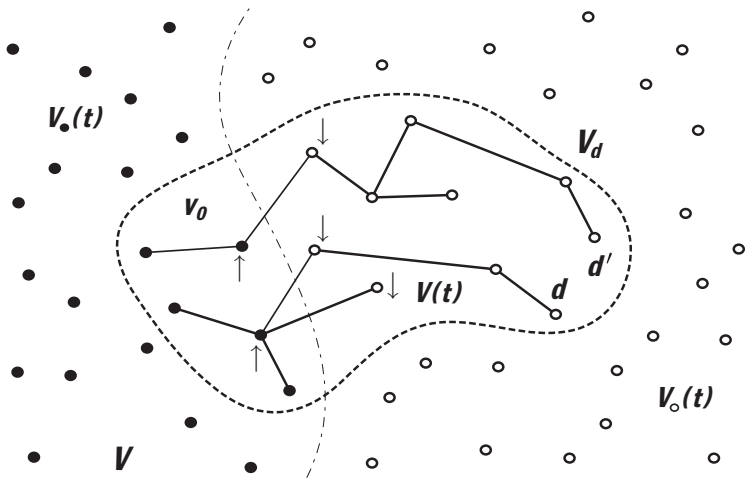


FIGURE 5.4 $V_\bullet(d,t)(\uparrow)$, $V_\circ(d,t)(\downarrow)$, $E(d,t)$, and V_d

Now, we will attempt to analyze the relationship between t, t_m, and t_c. We know that the two nodes u and v are already connected at time t, so $t_c < t$. Further, since $u \in V_\bullet(d,t)$ at time t, the message must have already arrived at u before t, which means $t_m < t$. However, clearly we can only have either $t_m < t_c$ or $t_c < t_m$, as shown in Figure 5.5.

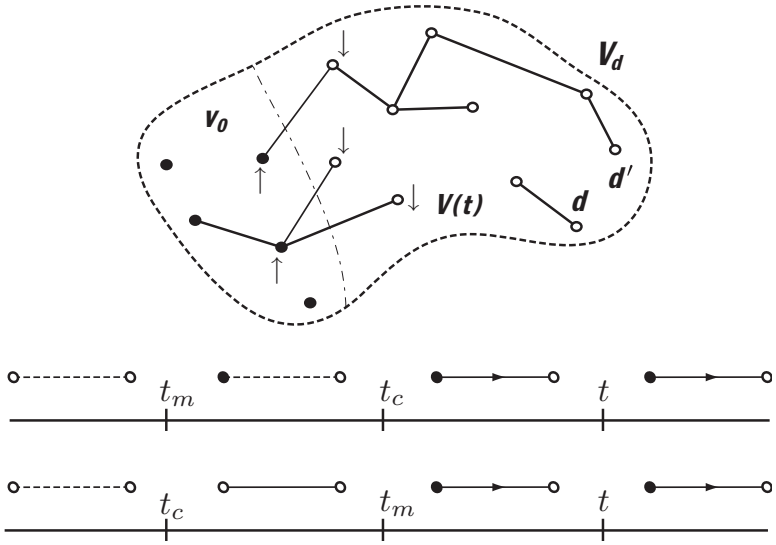


FIGURE 5.5 Two possible orders between t, t_c, and t_m

In the first case of $t_m < t_c$, the connection is established after u already receives and stores the message, while in the second case of $t_c < t_m$, the message arrives after connection between u and v is established. In both cases, the message m is transmitted from u to v at time $\max(t_m, t_c)$. Let us consider these two mutually exclusive cases:

(a) If the message is transmitted at t_c, then by assumption 4 of HDN, the message must reach v before $t_c + \tau$. Next, by assumption 5, we can guarantee that up to time $t_c + \tau$: (i) the connection e established at time t_c cannot be broken, and (ii) the node v cannot leave the network. Otherwise, either one of these will produce an event in C_u less than τ apart from the previous event. Therefore, at $t_c + \tau$, the node v can be guaranteed to be in $V_\bullet$.

Let us set τ_m in assumption 6 to a very large number for now, in effect retaining the message m indefinitely at the nodes that have received it. Consequently, at $t + 2\tau$, the node v is still in $V_\bullet$. If the case we just described — that the message is transmitted at t_c — is true for at least one edge e (and a terminal node v), then at $t + 2\tau$, the set $V_\bullet$ grows by one node, and $V_\circ$ shrinks by one node compared to their sizes at t.

(b) If the message is transmitted at t_m, then the situation is more complicated to prove. During the maximum transit time τ required by m to reach v from u, the edge e might disconnect itself; or, the target node v itself might leave the network before it receives the message. In fact, these events might take place sometime at t_d even before t_m, which means $t_c < t_d < t_m < t$!

We invoke assumption 2 to prove that such a failure cannot happen on *all* the edges in $E(d,t)$. If failure does not occur on the edge e, then again by assuming a large τ_m, a successful message transmission on e must increase the size of $V_\bullet$ by one, and decrease the size of $V_\circ$ by one at $t_m + 2\tau$.

To prove this claim, consider the separate timelines shown in Figure 5.6, one for each edge in $E(d,t)$, all aligned with marks placed at time t. A message traveling along an edge e is represented by an interval $[t_m, t_m + \tau]$ that is placed on the timeline such that $t_m \leq t \leq t_m + \tau$. Since several edges may originate from the same node and thus have the same t_m, some intervals may occupy identical horizontal locations on their timelines.

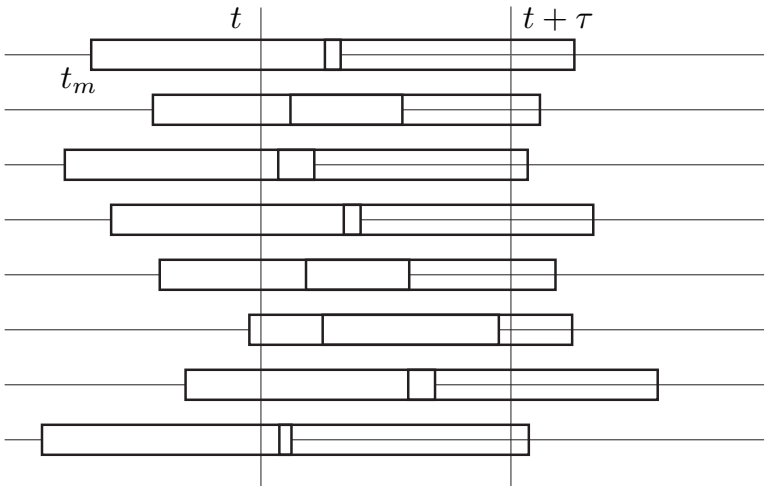


FIGURE 5.6 The timelines for $E(t)$ showing transit and dead regions

For a particular edge $e \in E(d,t)$, let t_d denote the time when either e disconnects, or v leaves the network. Since e exists at t, then $t_d > t$. If $t \leq t_d < t_m + \tau$, then the message fails to reach v. For $\forall e \in E(d,t)$, call the region $[t_d, t_d + \tau]$ the dead region $d(e, t_d)$. Again, since several edges may head toward the same node and as a resullt have

the same disconnection time t_d, some of these dead regions may occupy identical time regions.

Select two of the timelines with the maximum and minimum values of t_m, denoted by t_m by ${t_m}^*$ and t_{m*}, along with the associated intervals. The two intervals must overlap because they both contain t, otherwise it will contradict the original assumption that all intervals are aligned at time t. Next, if these two extreme intervals overlap, then the other non-extreme intervals with t_m obeying $t_{m*} < t_m < {t_m}^*$ must also overlap with at least one of them.

Suppose dead regions exist in *all* the intervals. Using the above argument, then the dead regions of the maximum and minimum intervals must also overlap because both of them must contain the time marker $t+\tau$. Just as before, we claim that the dead regions of all intervals must also then overlap at $t+\tau$. But this means that the nodes in $V_\bullet(d, t+\tau)$ are completely disconnected from those nodes in $V_\circ(d, t+\tau)$. Since at least one of the destination nodes must be in $V_\circ(d, t+\tau)$, this contradicts assumption 2.

Therefore, to preserve connectivity, there must be at least one interval without a dead region. A message m successfully transmitted along the edge associated with this interval will convert one node v from $V_\circ$ into $V_\bullet$ sometime between the edge's t_m and $t_m+\tau$. Assuming a large τ_m as before, at $t+2\tau$, v remains in $V_\bullet$. This implies that the set $V_\bullet$ grows by one node, and the set $V_\circ$ shrinks by one node from their original sizes at time t.

So far, we have been assuming that $d \in V_d$ is part of $V_\circ(t)$. These destination nodes are separated from the nodes $u \in V_\bullet(d, t)$ by zero or more nodes $v \in V_\circ(d, t)$. For now, assume that $d \notin V_\circ(d, t)$. At least one of the nodes in $V_\circ(d, t)$ must remain connected to $V_\bullet(d, t)$ to satisfy assumption 2. Consequently, one of the nodes in $V_\circ(d, t)$ then joins $V_\bullet(d, t+2\tau)$ at a later time. This proves that $|V_\bullet(d, t+2\tau)| > |V_\bullet(d, t)|$ when $d \in V_\circ(d, t)$.

This process is repeated until $d \in V_\circ(d, t)$ (or equivalently, until $N(d, t) \cap V_\bullet(d, t) \neq \varnothing$). When $d \in V_\circ(d, t)$, then it will have received a message from a node in $V_\bullet(d, t)$ at $t+2\tau$.

Once $d \in V_\bullet(t)$, then one of its neighbors $n \in N(d,t)$ are also in $V_\bullet(t)$ and assumption 2 is automatically met. Consequently, the remaining nodes in $V_\circ(t)$ may or may not be in contact with the other nodes in $V_\bullet(t)$. This proves that $|V_\bullet(d, t+2\tau)| \geq |V_\bullet(d,t)|$ when $d \in V_\bullet(t)$.

Imagine the same process happening in parallel to all the destination nodes in V_d as shown in Figure 5.7. Nodes in $V_\bullet(t)$ may leave $V(t)$ and become isolated, but when they re-enter the network at $t' > t$, they will be in $V_\bullet(t')$. The nodes in $V_\circ(t)$ is converted to $V_\bullet(t+2\tau)$ one by one.

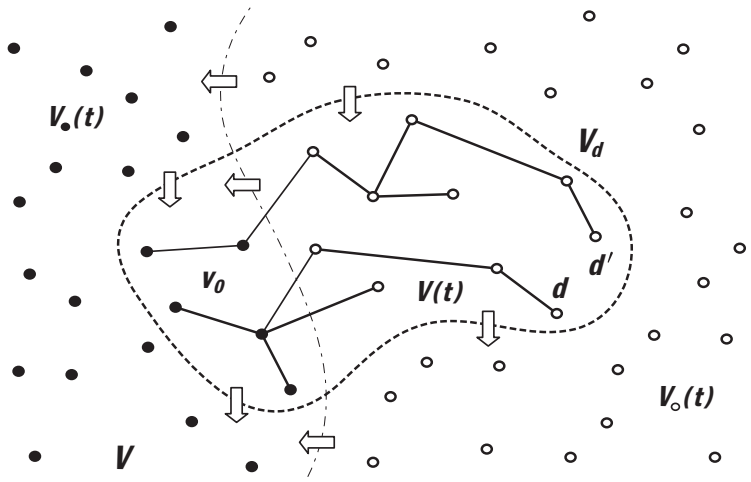


FIGURE 5.7 The arrows indicate allowable transitions of the network nodes in the sets $V_\circ(t)$, $V_\bullet(t)$, $V(t)$, and $V \backslash V(t)$ for HDN.

The arrows in Figure 5.7 indicate these transitions. Since $|V|$ is finite, the entire V_d is guaranteed to ultimately join $V_\bullet$ at some time t^*, at which point $V_\bullet(t > t^*)$ may or may not grow further. We have proved Lemma 11. □

Having proven the preceding Lemma 11, we now prove that the CFA is a solution to the HDN problem. The algorithm requires one input $\bar{V}$ and intercepts two types of events: the connectivity-driven events $c(v,t)$ and the message-driven events $m(v,t)$. When the message is first received at v, the counter k is reset to 0, and the message is retrans-

mitted to $N(v,t)$. If v is notified of a neighborhood change with the arrival of $c(v,t)$, it increments the counter k by one until it reaches the maximum $2\bar{V}$.

Theorem 12. *The CFA solves the HDN problem in less than* $2\tau\,\bar{V}$ *time unit.*

To prove that the CFA is a solution to the HDN problem, we have to prove two things. First, we have to prove that a message sent from the source using the CFA under the HDN assumption is guaranteed to reach the destination nodes. However, this was already proved in Lemma 11. The counter k is used to ensure that the CFA does not broadcast after termination.

Second, we have to prove that the CFA ends in finite time. Consider one destination node d. In the worst case scenario, $|V_\bullet(t)|$ is equal to 1 at $t=0$ (where $V_\bullet(0)$ contains only the source v_0), is equal to 2 at $t=1$, is equal to 3 at $t=2$, and so on, up to $t=2\tau\,(\bar{V}-1)$ where $|V_\bullet(t)| = |V| \leq \bar{V}$. Therefore, we can say that by $t=2\tau\,\bar{V}$, the message has reached the destination nodes. In other words, the CFA terminates in less than $2\,\tau\,\bar{V}$ time unit. □

The CFA can also be extended to support multiple messages simultaneously. From the listing, we can see that if the simple counter k is converted into an array k_m indexed by the message, then it terminates after all $k_m = \bar{V}$.

So far, we assume a large value of message retention time τ_m to ensure that when a node in $V_\bullet(t)$ leaves $V(t)$, it returns into $V_\bullet(t')$ some time $t' > t$ later. However, while these nodes need to pass many messages over its lifetime, realistically, most HDN nodes only have a limited amount of storage which also stores $N(v,t)$ used to detect $c(v,t)$.

Therefore, the nodes cannot keep their messages indefinitely — a large value of τ_m. From Theorem 12, if the nodes have access to a synchronized network clock, then they need to keep only messages $2\,\tau\,\bar{V}$ and younger (relative to the origination timestamps). Otherwise, messages $4\,\tau\,\bar{V}$ and younger (relative to the local *receipt* time) are needed.

There are other flooding and routing algorithms in [1] that use slightly modified assumptions. For example, the LIST FLOODING algorithm (LFA) does not assume that $\bar{V}$ is available. Instead, it assumes that the nodes have unique identifiers used to estimate $\bar{V}$.

Since these algorithms also use the CFA as their main ingredient, they too can be easily extended using Lemma 11 to solve the modified HDN problems. Thus, we omit this derivation.

Finally, we observe that the connectivity assumption of a HDN can be relaxed even further without affecting the reliability of the CFA! Assumption 2 can be restated as: $E(t)$ is such that at any time t, there is at least one connection between one node in $V_\circ(t)$ and another node in $V_\bullet(t)$.

The proof of this assertion is quite simple and sketched as follows. By requiring that at least one node in $V_\circ(t)$ is connected to a node in $V_\bullet(t)$ at all times, we can use the same argument as in Lemma 11 and state that $|V_\bullet(d, t+2\tau)| \geq |V_\bullet(d,t)|$. Since the number of nodes in V is finite, eventually d will be connected to a node in $V_\bullet$. In the worst case scenario, the nodes in V_d are the last nodes to receive the message, implying an identical upper bound.

5.2 The Chinese Generals Problem

A sensor network is built from a large number of sensor nodes distributed over an area where measurement of a specific phenomena or detection of a particular event is desired [12, 13]. Such a network is radically different from traditional networks in that its performance is largely determined by the density of its sensor nodes rather than their precise locations and interconnection topology. For this reason, sensor networks are often deployed in environmentally hostile areas where precise placement and configuration is prohibitively difficult. Operating in such areas, a sensor network is expected to experience a higher rate of node and link failures.

To successfully integrate any sensor network into a mission-critical detection and decision system, a designer needs to devise a mechanism that effectively addresses and mitigates these failures. For simplicity, in this section we restrict our discussion to bi-

nary detection and decision systems.

In a binary detection system, all sensor nodes collectively attempt to ascertain the existence or absence of a particular event of common interest. In a binary decision system, measurement results from all sensor nodes are used to decide between two possible course of actions. Naturally, since the sensors may have different positions, operating environment, and operating parameters, they may end up with different conclusions on whether or not an event is detected, and if it is, on the most appropriate decision to make.

In one extreme scenario, the entire network delegates the decision-making task to a single control node, which may or may not be a sensor node. Although appealing in its simplicity, this delegation effectively negates any desirable fault-tolerant properties that are inherent to sensor networks. Using this approach, the control node now becomes essentially a single point of failure of the entire network in several different ways.

First, either the node has to gather measurement results from all the sensor nodes, or all the sensor nodes have to reach the control node to transfer the measurement results. Second, even if we assume that all measurement results successfully reach the control node, they may not be all in agreement with one another. To reconcile the disagreement, the control node has to administer a (majority) voting procedure, which itself is also subject to failure, before finally committing to a decision.

Following the arguments in [14], we can say that if the control node is intended to be the one and only consumer of information produced by the sensor nodes, i.e., no other process or system relies on the control node's output, then from its point of view, the only relevant vulnerabilities are those related to routing and communication from the sensor nodes. However, if the control node affects other upstream processes, or if the network employs many control nodes for robustness, then processes at each control node become single-point network points of failure and sources for discrepancies.

The solution is to implement a *distributed voting* mechanism where every sensor node

in the system aggregates information from its neighboring nodes, administers a local voting procedure, and waits for the next round of information gathering. First introduced by Lamport in 1982, the *Byzantine Generals Problem* [11] (BGP) addresses this need of redistributing the voting process. A *Byzantine agreement* is reached when all operational sensor nodes agree on a common detection or an identical decision. At this stage, all operational sensor nodes carry the same information, and any upstream process can gather this information by sampling from any one of them.

Distributed voting cannot be used in systems that measure and choose their values from the set of real numbers. Sensor networks in such systems use a *distributed averaging* mechanism to combine local information in the presence of noise, delay, and unstable topology [15–19].

As an alternative to voting and averaging, we propose a new aggregation mechanism based on *thresholding*. Although in this section we only analyze binary threshold functions, the concepts and techniques introduced herein can be extended to threshold functions operating on real numbers.

This section is organized as follows. In this subsection the *Byzantine Generals Problem* (BGP) and one of its relevant variants are reviewed and compared to the proposed *Chinese General Problem* (CGP). In our attempt to clarify the notations and concepts of the CGP, we provide a detailed example in subsection 5.2.1. This example is then used as the foundation on which we present our main results in subsection 5.2.2.

In their seminal 1982 paper, Lamport et al. [11] introduced and analyzed the problem of reliable message synchronization in an otherwise unreliable distributed system, presented abstractly in terms of a group of Albanian Generals attempting to coordinate a synchronized action using messengers, and in the presence of treacherous generals amongst them.

To obtain the impossibility theorem and other results in their paper, the problem is reduced into the one involving three Byzantine Generals, each of which represents an en-

semble of Albanian generals and possibly some traitors. This reduced problem is called the *Byzantine Generals Problem*.

Since its introduction, the problem has been studied by many researchers in the area of fault-tolerant distributed computation [20–28]. However, this class of problem is also relevant for communication systems with messages from multiple sources that need to be synchronized over a dynamic network.

From a large number of BGP variants that have been developed since, two variants are relevant to our paper. Wang et al. [20] and Babaoglu et al. [21, 22] generalized the BGP by not requiring a full network connectivity and by modifying the definition of a faulty component, respectively. In this paper, we further extend the BGP in the same directions as the previous two generalizations by relaxing even more assumptions.

Before discussing our generalization to the BGP, perhaps it is beneficial to first discuss what the BGP really is. Using the summary from [20], we can state that the BGP assumes the following: (1) there are S generals of which at most F generals may either maliciously or inadvertently change the received message before passing the message to the other generals; (2) all generals have direct access to each other through the use of messengers, i.e., the communication network is fully connected; (3) the sender can be identified from either the message or the messenger; (4) there is one commanding general who broadcasts an initial message v_0 in the beginning of the campaign and finally, (5) messages are modified only by the generals, but not by their messengers. The BGP is solved by developing optimal algorithms that can achieve the so-called *Byzantine agreement* (BA): (BA1) All loyal generals agree on a common message v, and (BA2) if the commanding general is loyal, then v should be identical to the initial message v_0. In its original form, the BGP considers asynchronous communication between the generals.

The algorithm proven in [20] does not assume a fully connected network. This is a significant result, as otherwise the network has to suffer a quadratic growth in the number of communication resources (I/O equipments, frequency, etc.) to guarantee that it

can achieve a BA. Babaoglu extended the BGP by providing specific failure modes that include crash and omission.

Crash is just another word for a permanent failure: once a general enters the crash mode, it can no longer send out its messengers to the other generals. In contrast, omission is just a temporary failure. A general ignores all the message it receives during the omission period, but continues to behave normally after the omission period is over.

Having described the BGP, we are ready to discuss our extensions. Using the next letter in the alphabet (after Albanian and Byzantine), we choose to call our extended problem the *Chinese Generals Problem* (CGP). First, some comments are due with regard to the similarities between the CGP and its superclass, the BGP. Both problems can be abstractly described as the problem faced by a team of generals who plan to commit a synchronized action by communicating via messengers. As such, solving the BGP and the CGP both involve developing an algorithm satisfying (BA1) and (BA2). Now we discuss the main differences between the two.

Here, we introduce a preliminary version of the CGP that requires a synchronous communication (messenger) service among generals whose output is determined only by the current input. Consequently, in terms of synchronicity, the CGP is more stringent than the BGP because each message has to leave the sender and arrive at the destination within one clock cycle. The nodes can be "synchronized" by flooding an initialization message. If one clock cycle is much longer than the propagation time, the generals can achieve synchronous operation by delaying the message by one cycle.

Apart from this real difference, an apparent difference comes from our conscious effort to restrict our analysis of the CGP to systems that exclusively use binary messages. While at first this might hint at a severe limitation of the CGP, Turpin and Coan [29] showed that BA can be reached in systems operating on multivalue messages that can be represented by k bits by running the appropriate binary BA algorithm k times.

The Chinese General Problem

Consider a group of Chinese Generals $G_i \in \mathcal{G}$, $i = 0 \ldots S$. At any given time t, the two possible actions they can take are labeled A_0 and A_1. Denote by $\mathcal{G}_1(t)$ the group of generals who choose A_1 at time $t \in \mathbb{N}$, and by $K(t)$ the number of such generals. The generals reach a consensus on A_1 if and only if there exists a certain time τ after which the number of generals in $\mathcal{G}_1(t)$ is very large, or $K(t) \approx S$. Likewise, consensus on A_0 is reached if and only if after τ the number of generals in $\mathcal{G}_1(t)$ is negligible, or $K(t) \approx 0$.

The CGP asks whether consensus among the generals is possible with the given algorithm described below, and if so, under what condition.

At $t = 0$, each general G_i produces a binary message bit $v_i(t)$ to indicate whether they choose A_0 or A_1. If $v_i(t) = 1$, then G_i dispatches M indistinguishable messengers to γ other generals $G_j \in \Gamma(G_i)$. The function Γ allows us to construct an γ-regular graph G where $\mathcal{G}$ is the vertex set of G, and $\{E_{ij}\}$ the set of edges of G connecting G_i to G_j whenever $G_j \in \Gamma(G_i)$. Let $L_i(t)$ be the number of messengers arriving at G_i at t. The update rule for $v_i(t)$ is:

$$v_i(t+1) = V(L_i(t)) = 1 \quad \text{iff} \quad L_i(t) \geq T. \tag{5.1}$$

Let us call T and V the threshold value and the threshold function, respectively. In this problem, the messenger is also allowed to not reach any of the γ generals and instead return to the originating general. Each messenger chooses the $\gamma + 1$ destinations with equal probability. Returning messengers are also counted as arrivals by the function $L_i(t)$. □

Compared to the majority voting function used in the BGP, V is more flexible because it allows a particular general G_i to still choose a decision A_1 even if the value of $L_i(t)$ indicates that it is not the majority opinion among the generals from whom information flows to G_i via the messengers.

In addition, the CGP does not assume that failures are contributed solely by the nodes (the generals) as assumed in [20]. In the CGP, communication links (the messengers) behave probabilistically, reflecting the relatively lower reliability that is often experienced in wireless and sensor networks. We will also show that the effect of node unreliability can be included in this model. An unreliable node decides on A_0 when deterministically, under the normal behavior, the node should have decided on A_1 instead, based on the value of $L_i(t)$. Interestingly, threshold functions are also found in many different types of natural systems, the most famous being the ones found in neurons.

5.2.1 A Simple Example

Before proceeding to the next subsection for a full analysis of the CGP, recall that the definition includes the number of generals S, their valency γ, the initial condition $K(0)$ of the number of generals deciding on A_1, and the two tunable parameters M and T denoting the number of messengers dispatched and the corresponding threshold value, respectively.

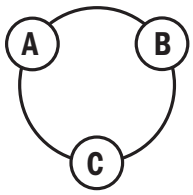


FIGURE 5.8 Three generals

In this section, we analyze a simple example of a CGP where we use fixed values of $S = 3$, $\gamma = 1$, $M = 2$, and $T = 2$. The problem involves three generals labeled A, B, and C, linked in a simple triangular network shown in Figure 5.8 above. First, at $t = 0$ each general can be either in $\mathcal{G}_1(0)$ or not, depending on the value of its message $v_i(0)$. Therefore, we have a total of eight possible message configurations as shown in Figure 5.9. For each of these configurations, the general with $v_i(0) = 1$ can dispatch two messengers each. With equal probability each messenger can then either go to the neighboring general in the clockwise direction, or stay with G_i.

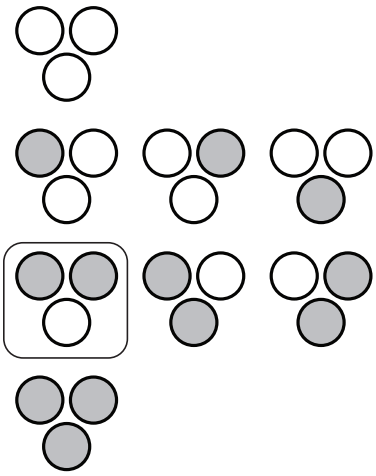


FIGURE 5.9 Eight possible message configurations

For $\boldsymbol{v}(0) = \{v_A(0)\, v_B(0)\, v_C(0)\}$, we have eight equally likely message configurations that can be written as binary strings: {000}, {001}, {010}, {100}, {011}, {101}, {110}, and {111}. Notice that we sort the eight configurations according to the number of 1s in their strings. The reason is quite simple. The triangle in Figure 5.8 is symmetrical, which means configurations having the same number of 1s in their strings should be mathematically identical.

Let us consider the highlighted configuration where two generals A and B, initially decide on A_1 and dispatch two messengers each. Although the messengers are identical, for illustration purpose, we label the messengers at A's and B's positions with a_1, a_2, b_1, and b_2. Let us devise a notation with two dots in which the messengers currently at A's, B's, and C's positions are written before the first dot, between the dots, and after the second dot, respectively. An o at any position indicates that there is no messenger. For the highlighted configuration, the messenger positions at $t = 0$ are denoted by:

$$a_1 a_2 . b_1 b_2 . o$$

For the highlighted message configuration shown on Figure 5.9, there are 16 possible ways with which the messengers can arrive at their destinations as shown in Figure

5.10 below. We call each one of them a messenger configuration. Each dot represents a messenger at that particular position.

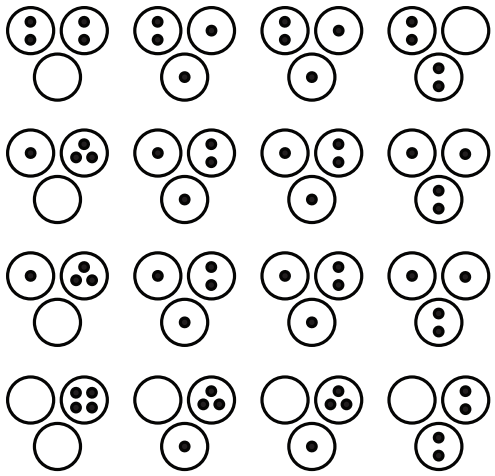


FIGURE 5.10 Sixteen possible messenger configurations

With the exception of their labels, these messenger configurations are practically identical to the ones produced in the cases where the message configurations have generals B and C in $\mathcal{G}_1(0)$, or generals C and A in $\mathcal{G}_1(0)$. Using the same notation, all sixteen messenger configurations are listed below.

$$
\begin{array}{cccc}
a_1a_2.b_1b_2.o & a_1a_2.b_1.b_2 & a_1a_2.b_2.b_1 & a_1a_2.o.b_1b_2 \\
a_1.a_2b_1b_2.o & a_1.a_2b_1.b_2 & a_1.a_2b_2.b_1 & a_1.a_2.b_1b_2 \\
a_2.a_1b_1b_2.o & a_2.a_1b_1.b_2 & a_2.a_1b_2.b_1 & a_2.a_1.b_1b_2 \\
o.a_1a_2b_1b_2.o & o.a_1a_2b_1.b_2 & o.a_1a_2b_2.b_1 & o.a_1a_2.b_1b_2
\end{array}
$$

From the above information, we can compute the message configuration at time $t = 1$. To determine $\boldsymbol{v}(t)$ at $t = 1$ if $\boldsymbol{v}(0) = \{110\}$), first we have to compute the arrival functions $\boldsymbol{L}(0) = \{L_A(0), L_B(0), L_C(0)\}$ that compute the number of messengers at each general at time $t = 0$. Then we apply the threshold function V on these messenger arrival values to obtain the value of $\boldsymbol{v}(1)$.

First, we will remove the distinction between all messengers arriving at the same general. This is compatible with Figure 5.10, where the messengers are unlabeled by their

source general. To do this, we denote the messengers currently at A's, B's, and C's by a, b, and c. As a shorthand, we denote aa by a^2 (and likewise with b). In this notation, the dots and o's are no longer needed and we have a more concise notation:

$$\begin{array}{cccc} a^2b^2 & a^2bc & a^2bc & a^2c^2 \\ ab^3 & ab^2c & ab^2c & abc^2 \\ ab^3 & ab^2c & ab^2c & abc^2 \\ b^4 & b^3c & b^3c & b^2c^2 \end{array}$$

or, algebraically, as a multivariate polynomial where each term and its coefficient corresponds to a messenger configuration and its multiplicities,

$$\begin{aligned} F_{2a} = {} & b^4 + a^2b^2 + b^2c^2 + a^2c^2 \\ & + 2a^2bc + 4ab^2c + 2abc^2 + 2b^3c + 2ab^3 \quad . \end{aligned} \tag{5.2}$$

At $t = 0$, the configuration is always a^2b^2 because $\mathcal{G}_1(0) = \{A, B\}$. Suppose after the first round, the configuration becomes ab^2c. F_{2a} contains abundant information: there are four ways to achieve this configuration, and the value of $\boldsymbol{L}(0)$ is $\{\deg a \deg b \deg c\}$, which is $\{121\}$.

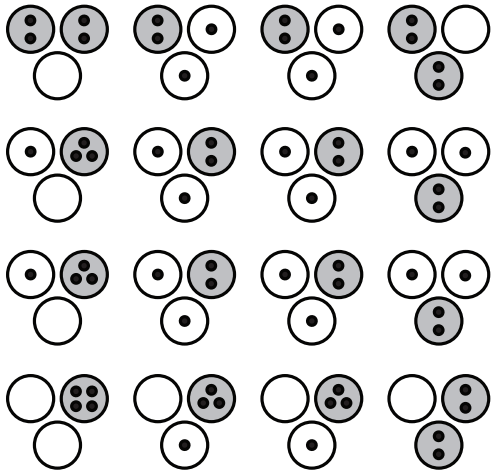


FIGURE 5.11 Obtaining the new message configurations from the previous messenger

configurations by applying the threshold

With a little more work, we can get even more information. By applying the threshold function V with $T = 2$ on $\boldsymbol{L}(0)$, we obtain $\boldsymbol{\nu}(1) = \{010\}$. An element of this string is a zero if the degree of the corresponding variable is less than T, and a one otherwise. In Figure 5.11 above, the generals G_i that satisfy the condition $\boldsymbol{\nu}_i(1) = 1$ are marked by the grey circles.

Finally, we can also compute $K(0)$ and $K(1)$ by counting the numbers of ones in $\boldsymbol{\nu}(0)$ and $\boldsymbol{\nu}(1)$, which are 2 and 1, respectively. Calculating $K(t)$ is of interest to us because it shows the evolution of $\mathcal{G}_1(t)$. Of course, in reality we can only compute an average value of $K(1)$ because the four messengers could just as well choose configurations other than ab^2c. The average is:

$$K(1) = \frac{\sum_i K_i(1)}{\kappa} \tag{5.3}$$

where the index i runs over all sixteen possible configurations at $t = 1$. In the numerator summand, the function $K_i(t)$ counts the number of ones in $\boldsymbol{\nu}_i(t)$ from a configuration i at time t. The denumerator κ is simply the total number of configurations, which in our present case is sixteen.

Having discussed $\mathcal{G}_1(0) = \{A, B\}$, we can now consider the other possible memberships of $\mathcal{G}_1(0)$: $\{\varnothing\}$, $\{A\}$, $\{B\}$, $\{C\}$, $\{A, B\}$, $\{A, C\}$, $\{B, C\}$, $\{A, B, C\}$. To do this, we need a polynomial that is more general than F_{2a}.

Consider the following polynomial $F(a, b, c; z)$ (or $F(z)$, for short). The coefficient F_k of z^k captures all the possible configurations given that $|\mathcal{G}_1(0)| = k$, i.e., k generals initially deciding on A_1. Obviously, $F(z)$ is more general than F_{2a} as all the configurations in F_{2a}

can also be found in F_2.

$$
\begin{aligned}
F(z) &= (1+(a+b)^2 z)\,(1+(b+c)^2 z)\,(1+(c+a)^2 z) \\
&= 1 + F_1 z + F_2 z^2 + F_3 z^3 \\
F_k &= [z^k]F(z) = \tfrac{1}{k!}\tfrac{\partial^k}{\partial z^k} F(z)\big|_{z=0} \\
F_1 &= 2ab+2bc+2ca+2a^2+2b^2+2c^2 \\
F_2 &= a^4+b^4+c^4 \\
&\quad + 2ab^3+2bc^3+2ca^3+2a^3b+2b^3c+2c^3a \\
&\quad + 3a^2b^2+3b^2c^2+3c^2a^2+8a^2bc+8ab^2c+8abc^2 \\
F_3 &= 10a^2b^2c^2+2a^3b^3+2b^3c^3+2a^3c^3 \\
&\quad + 6a^3b^2c+6a^3bc^2+6a^2b^3c+6a^2bc^3 \\
&\quad + 6ab^3c^2+6ab^2c^3+2a^4bc+2ab^4c+2abc^4 \\
&\quad + a^4b^2+a^2b^4+b^4c^2+b^2c^4+c^4a^2+c^2a^4 \qquad \text{(X1)}
\end{aligned}
$$

Let us first define F_{kl} as the number of l-th power of a, b, and c found in the monomials (i.e., terms, or configurations) of F_k. The index k restricts our count only to F_k, which is described above, while l restricts the count to only those generals having exactly l messengers: $\boldsymbol{L}_i(0) = l$. For example, in F_2, the forms a^2, b^2, or c^2 can be found in these terms:

$$3a^2b^2+3b^2c^2+3c^2a^2+8a^2bc+8ab^2c+8abc^2 \quad .$$

We can expand the above expression to count the number of monomials of degree 2, which is given by $3\times 3\times 2+3\times 8\times 1 = 42$. Therefore, $F_{22} = 42$.

Having defined F_{kl}, we can now compute $K_k(1)$, which is the average value of $|\mathcal{G}_1(1)|$ over all configurations in F_k.

$$\begin{aligned}
K_1(1) &= F_{12}/\kappa_1 = 6/(3\cdot 2^2) \\
&= 6/12 = 0.5000 \\
K_2(1) &= (F_{22}+F_{23}+F_{24})/\kappa_2 \\
&= (42+12+3)/(3\cdot 2^4) \\
&= 57/48 = 1.1875 \\
K_3(1) &= (F_{32}+F_{33}+F_{34})/\kappa_3 \\
&= (72+48+12)/(1\cdot 2^6) \\
&= 132/64 = 2.0625,
\end{aligned}$$

or more generally, we can use ($x' \in \mathcal{G} \setminus x$):

$$\begin{aligned}
K_k(1) &= \frac{1}{\kappa_k}\sum_{l\geq T}^{Mk} F_{kl} \\
F_{kl} &= \sum_{x\in\mathcal{G}} \tfrac{1}{k!}\tfrac{\partial^l}{\partial x^l} F(a,b,c;z)\big|_{x=0,x'=1} \\
\kappa_k &= \tbinom{S}{k}(\gamma+1)^{Mk}
\end{aligned} \tag{5.4}$$

Consider the case where all three generals are in $\mathcal{G}_1(0)$. Not knowing each other's decisions, they send their messengers out to notify their neighbors. From $K_k(1)$, we predict that $|\mathcal{G}_1(1)| = 2.0625 \approx 2$. Likewise, in the next time step $|\mathcal{G}_1(2)| = 1.1875 \approx 1$, and finally, at $t = 3$, the generals no longer agree on A_1 as $|\mathcal{G}_1(3)| = 0.50 < 1$. Therefore, with the given M, T, γ, and S, a proper consensus reflecting the generals' initial observations cannot be reached.

In Section 5.2.2, we present the general results for γ-regular network for all possible parameter values and show that with an appropriate choice of parameters, a proper con-

sensus can be reached. In preparation for these general results, let us first generalize our triangular network into a ring network G with S generals (shown below with eight generals).

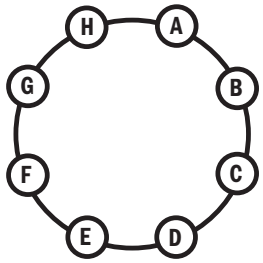


FIGURE 5.12 The network G with eight generals

In our analysis of G, the values of M, S, and T are no longer fixed. As a result, we can no longer count the results manually and have to resort to the mathematical formalism of multivariate generating function (mgf).

Let us denote the mgf for G by $F(\boldsymbol{x};z)$, with $\boldsymbol{x} = \{x_i\} = \{x_0, x_1, \ldots, x_{S-1}\}$ and $i \in \mathbb{N}$ mod S (i.e., $i + S \equiv i \mod S$) indexing the S generals in $\mathcal{G}$. Denote by $[z^k]$ the operator that extracts the coefficient of z^k and by $[x_i^l]$ the operator that extracts the coefficient of x_i^l. As before, let i' denote any member of the index set that is different from i which is $\{0, \ldots, S-1\} \setminus i$.

$$\begin{aligned} F(\boldsymbol{x};z) &= \textstyle\prod_{i=1}^{S} \left(1 + (x_i + x_{i+1})^M z\right) \\ F_k(\boldsymbol{x}) &= [z^k]F(\boldsymbol{x};z) = \tfrac{1}{k!}\tfrac{\partial^k}{\partial z^k} F(z)\big|_{z=0} \\ F_{kl} &= \textstyle\sum_{i=1}^{S} [x_i^l] F_k(\boldsymbol{x}) \\ &= S\tfrac{1}{k!}\tfrac{\partial^k}{\partial z^k}\tfrac{1}{l!}\tfrac{\partial^l}{\partial x_i^l} F(\boldsymbol{x};z)\big|_{x_i=0, x_{i'}=1} \end{aligned} \tag{5.5}$$

The formulas for $K_k(t)$ and κ_k are the same as the ones found in (5.4). The summands in (5.5) are identical due to the symmetry of $F(\boldsymbol{x};z)$ with respect to $\boldsymbol{x}$. To derive the explicit formula for F_{kl}, suppose $l \geq 1$. Although counterintuitive, it is easier to start

with $[x_i^l]$ instead of the standard operator $[z^k]$.

$$\begin{aligned}
F_l(z) &= S\,[x_i^l]F(\boldsymbol{x};z) \\
&= S\,\tfrac{1}{l!}\tfrac{\partial^l}{\partial x_i^l}F(\boldsymbol{x};z)\big|_{x_i=0,x_{i'}=1} \\
&= S\prod_{i'\in\{2,S\}}\left(1+(x_{i'}+x_{i'+1})^M z\right)\big|_{x_{i'}=1}\times \\
&\quad \tfrac{1}{l!}\tfrac{\partial^l}{\partial x_1^l}\left(1+(x_S+x_1)^M z\right)\left(1+(x_1+x_2)^M z\right)\big|_{x_1=0}
\end{aligned}$$

Note that in the last equation above, we have chosen to extract the l-th coefficient of x_1, effectively choosing $i=1$. Due to symmetry, the expressions for other i's are the same, which allows us to use the multiplicative factor of S.

$$\begin{aligned}
&= (1+2^M z)^{S-2}\,\tfrac{1}{l!}\tfrac{\partial^l}{\partial x_1^l}\left(1+(1+x_1)^M z\right)^2\big|_{x_1=0} \\
&= (1+2^M z)^{S-2}\,[x_1^l](1+2(1+x_1)^M z+(1+x_1)^{2M}z^2) \\
&= (1+2^M z)^{S-2}\left[\tbinom{2}{1}\tbinom{M}{l}(1)^{M-l}z+\tbinom{2}{2}\tbinom{2M}{l}(1)^{2M-l}z^2\right] \\
&= \textstyle\sum_{i=0}^{S-2}\tbinom{S-2}{i}2^{Mi}z^i\left[2\tbinom{M}{l}z+\tbinom{2M}{l}z^2\right]
\end{aligned}$$

The value of $x_{i'}=1$ is then substituted, before a tedious algebraic coefficient extraction procedure is run on the remaining polynomial that contains x_1.

$$\begin{aligned}
F_{kl} &= [z^k]F_l(z) = \tfrac{1}{k!}\tfrac{\partial^k}{\partial z^k}F_l(z)\big|_{z=0} \\
&= 2S\tbinom{M}{l}\tbinom{S-2}{k-1}2^{M(k-1)}+S\tbinom{2M}{l}\tbinom{S-2}{k-2}2^{M(k-2)}
\end{aligned} \tag{5.6}$$

At this point, recall that we haven't considered the case where $l=0$, which requires a slightly different derivation because the term corresponding to $l=0$ is the constant term in the polynomial.

The derivation for F_{k0}, is provided below. Again, we use the symmetry property of $F(\boldsymbol{x};z)$. Note that at the first glance, the result for F_{k0} below seems to be missing the $\binom{iM}{l}$

factor when compared to Equation (5.6). However, recall that $\binom{iM}{0} \equiv 1$ for all iM.

$$\begin{aligned}
F_{l=0}(z) &= \textstyle\sum_{i=1}^{S} F(\boldsymbol{x}; z)\big|_{x_i=0, x_{i'}=1} \\
&= S\, F(\boldsymbol{x}; z)\big|_{x_i=0, x_{i'}=1} \\
&= S \textstyle\prod_{i\in\{1,S\}} \left(1 + (x_i + x_{i+1})^M z\right)\big|_{x_i=0, x_{i'}=1} \\
&= S\,(1+2^M z)^{S-2}(1+z)^2 \\
F_{k0} &= S\,[z^k](1+2^M z)^{S-2}(1+z)^2 \\
&= S\,\tfrac{1}{k!}\tfrac{\partial^k}{\partial z^k} \textstyle\sum_{i=0}^{S-2} \binom{S-2}{i} 2^{Mi} z^i (1+2z+z^2)\big|_{z=0} \\
&= S \textstyle\sum_{i=0}^{2} \binom{2}{i}\binom{S-2}{k-i} 2^{M(k-i)}
\end{aligned} \tag{5.7}$$

This last observation suggests a formula for F_{kl} that is valid for $\gamma = 1$ and all values of k and l, that can be used in Equation (5.4):

$$F_{kl} = \sum_{i=0}^{2} S \binom{iM}{l} \binom{2}{i} \binom{S-2}{k-i} 2^{M(k-i)}$$

To conclude our analysis on this example, we consider the other extreme value for λ. Now, instead of setting $\lambda = 1$ and working with a ring network, we set $\lambda = S-1$ and work with a complete graph, which is somewhat simpler to analyze. The steps of deriving F_{kl} are identical:

$$
\begin{aligned}
F(\boldsymbol{x};z) =& \prod_{i=1}^{S}(1+(\sum_{j=1}^{S}x_j)^M z) \\
=& (1+(\sum_{j=1}^{S}x_j)^M z)^S \\
F_l(z) =& S[x^l](1+(S-1+x)^M z)^S \\
=& S[x^l]\sum_{i=0}^{S}\binom{S}{i}(S-1+x)^{Mi}z^i \\
=& S[x^l]\sum_{i=0}^{S}\binom{S}{i}\sum_{j=0}^{Mi}\binom{Mi}{j}(S-1)^{Mi-j}x^j z^i \\
=& S\sum_{i=0}^{S}\binom{S}{i}\binom{Mi}{l}(S-1)^{Mi-l}z^i \\
F_{kl} =& S[z^k]\sum_{i=0}^{S}\binom{S}{i}\binom{Mi}{l}(S-1)^{Mi-l}z^i \\
=& S\binom{S}{k}\binom{Mk}{l}(S-1)^{Mk-l} \quad .
\end{aligned}
\tag{5.8}
$$

As we mentioned previously, calculating $K_k(t)$ is important because the function allows us to learn about the evolution of $\mathcal{G}_1(t)$ and whether consensus is possible under the threshold function V. In our analysis of the triangular network, we manually calculated $K_k(t)$ for different values of k and learned that a proper consensus is not possible. Obviously, for the ring and complete network with variable parameters, a manual and exhaustive analysis of $K_k(t)$ is not an option. One feasible way would be a numerical evaluation of the values of $K_k(t)$ as a function of its parameters.

However, all is not lost. Let Mk be the number of messengers dispatched by k generals in $\mathcal{G}_1$, and $\lambda = \frac{Mk}{S}$. If we fix λ and $Mk \gg 1$, (5.8) becomes simple, and the qualitative behaviors of $K_k(t)$ and the consensus become clear.

$$
\begin{aligned}
P_{kl} = \frac{F_{kl}}{S\kappa_k} = \frac{S\binom{S}{k}\binom{Mk}{l}(S-1)^{Mk-l}}{S\binom{S}{k}S^{Mk}} \to \frac{\lambda^l}{l!}e^{\lambda} \\
K_k(1) = S\sum_{l\geq T}^{Mk}P_{kl} \to S\sum_{l\geq T}^{Mk}\frac{\lambda^l}{l!}e^{\lambda} = S(1-\frac{\Gamma(T,\lambda)}{(T-1)!})
\end{aligned}
$$

Perhaps not surprisingly, for $Mk \gg 1$ and $\gamma = S-1$, each term in the summation in Equation (5.4) is the mathematical expression for a Poisson density P_{kl} with parameter $\lambda = \frac{Mk}{S}$ and argument l, and therefore $K_k(1)$ can be expressed in terms of the incomplete

gamma function Γ.

Figure 5.13 is a plot of $K_k(t+1)$ against $K_k(t)$, which is actually the plot of $K_k(1)$ against $k = |\mathcal{G}_1(0)|$ for $M = 2$ (the lower curve) and $M = 4$ (the upper curve) compared to the diagonal line $K_k(t+1) = K_k(t)$. For both curves, we have set the parameters $\gamma = S - 1$, $T = 2$, and $S = 100$. These curves can be used to describe the time evolution of $K_k(t)$, and the existence of a consensus.

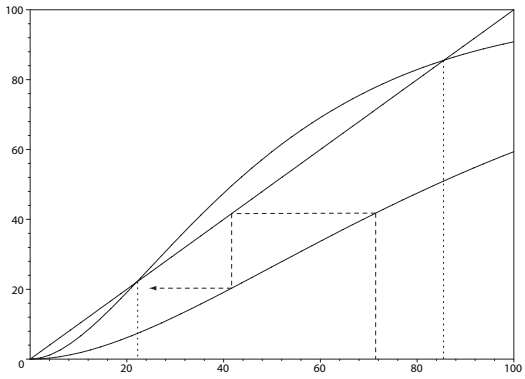


FIGURE 5.13 The plot of $K_k(t+1)$ versus $K_k(t)$

For example, suppose $K_k(0) = 72$. For the $M = 2$ curve, $K_k(1) \approx 42$, and $K_k(2) \approx 20$, before eventually reaching $K_k(t) = 0$ (the dashed path with an arrowhead). In other words, all generals would eventually settle on A_0 no matter how many of them initially agreed on A_1. In contrast, the $M = 4$ curve intersects the diagonal line at $k \approx 22$ and $k \approx 85$. These points define two possible steady-state behaviors of $K_k(t)$. If $K_k(0) < 22$, then as $t \to \infty$ $K_k(t)$ settles at the stable fixed point at $k = 0$. Otherwise, it settles at $k \approx 85$.

Denote by $F(k)$ the function that maps $K_k(t)$ to $K_k(t+1)$. The properties of Γ require that $F(k) \to 0$ as $k \to 0$ and $F(k) \to S$ as $t \to \infty$, and in addition, the slopes $\frac{\partial F(k)}{\partial k} \to \frac{1}{S}$ and $\frac{\partial F(k)}{\partial k} \to \frac{1}{S}$ as $t \to 0$ and $t \to \infty$, respectively. Due to space limitation, we do not analyze the effects of changing M and T on $F(k)$, or proofs to the above claims. We do, however, provide in the next section a sketch of the analysis of errors caused by the generals themselves.

5.2.2 General Result

Finally, we present the most general version of the *CGP* with no limits on any of the parameters. Shown in Figure 5.14(a) is one such network with $S = 9$ generals arranged in a 3×3 grid with $\gamma = 4$ such that each general G_i can send his messengers to the neighbors N_i to the east, west, north, and south (in addition to himself). The network wraps around on itself, so some neighbors are on the opposite side of the grid.

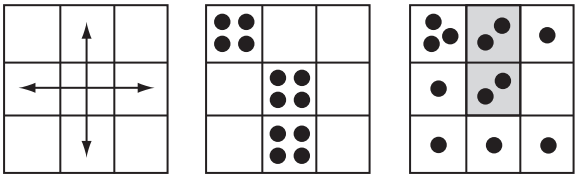


FIGURE 5.14 A general γ-regular network

Figure 5.14(b) shows the $k = 3$ generals from $\mathcal{G}_1(0)$, each equipped with $M = 4$ messengers. Figure 5.14(c) shows one possible messenger configuration after redistribution. The two shaded cells mark the generals G_i such that $l = L_i(1) = 2$. As before, F_{kl} then computes the number of such cells in all possible configurations. We generalize our previous results and prove that the following is true ($L = \gamma + 1$):

$$F_{kl} = \sum_{i=0}^{L} S\binom{Mi}{l}\binom{L}{i}\binom{S-L}{k-i}(L-1)^{Mi-l}L^{M(k-i)} \quad . \tag{5.9}$$

To prove the above equation, we use the same technique we used in the previous section. First, start with the mgf:

$$\begin{aligned}
F(\boldsymbol{x};z) =& \textstyle\prod_{i=1}^{S}(1+(\sum_{j\in N_i} x_j)^M z) \\
F_l(z) =& S\,[x_n^l]\,\textstyle\prod_{i=1}^{S}(1+(\sum_{j\in N_i} x_j)^M z) \\
=& S\,[x_n^l]\,(1+L^M z)^{S-L}(1+(L-1+x_n)^M z)^L \\
=& S\,\tfrac{1}{l!}\tfrac{\partial^l}{\partial x_n^l}(1+L^M z)^{S-L} f(x_n;z)\big|_{x_n=0} \\
f(x_n;z) =& (1+(L-1+x_n)^M z)^L \\
=& \textstyle\sum_{i=0}^{L}\binom{L}{i} z^i \sum_{h=0}^{Mi}\binom{Mi}{h}(L-1)^{Mi-h} x_n^h \quad .
\end{aligned}$$

Now we substitute the above expressions into F_{kl}:

$$\begin{aligned}
F_{kl} =& S\,[z^k]\,(1+L^M z)^{S-L}\tfrac{1}{l!}\tfrac{\partial^l}{\partial x_n^l} f(x_n;z)\big|_{x_n=0} \\
=& S\,[z^k]\,(1+L^M z)^{S-L}\textstyle\sum_{i=0}^{L}\binom{L}{i} z^i \binom{Mi}{l}(L-1)^{Mi-l} \\
=& S\textstyle\sum_{i=0}^{L}\binom{L}{i}\binom{Mi}{l}(L-1)^{Mi-l}[z^k](1+L^M z)^{S-L} z^i \quad .
\end{aligned}$$

Solving the last line gives us an expression identical to (5.9):

$$F_{kl} = \sum_{i=0}^{L} S\binom{L}{i}\binom{Mi}{l}\binom{S-L}{k-i}(L-1)^{Mi-l} L^{M(k-i)} \quad . \tag{5.10}$$

The expression for F_{kl} from (5.10) can then be substituted into (5.4) to obtain a plot similar to Figure 5.13. We also note that all γ-regular graphs have the same mgf, and hence the same F_{kl}, whether or not they are isomorphic to each other. Figure 5.15 shows two non-isomorphic 4-regular octagons with identical F_{kl}. Aside from node labels, their mgf's are exactly the same.

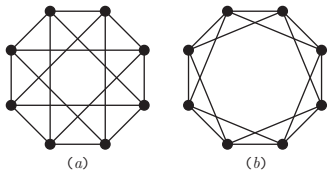


FIGURE 5.15 Two non-isomorphic 4-regular octagonal networks [30, 31]

Finally, we propose one way of modeling the case where the generals themselves commit decision errors. Suppose we have $k = |\mathcal{G}_1(t)|$. With probability p, each of the k generals may decide not to dispatch their messengers, and likewise, with probability p, each of the remaining $S-k$ generals may spontaneously decide to dispatch their messengers. Mathematically, this is equivalent to transforming the function $F(k)$ into $F'(k) = F(k) + p(S-k) - pk$.

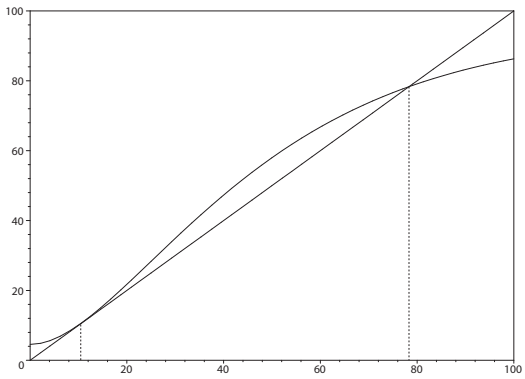


FIGURE 5.16 The plot of $K'_k(t+1)$ versus $K_k(t)$

An easy way to visualize this transformation is to imagine rotating $F(k)$ around a pivot located at $(\frac{S}{2}, F(k))$, as shown in Figure 5.16. Obviously, as a result, $F'(k)$ may or may not have any stable fixed point. However, the good news is that with a proper choice of parameters, $F'(k)$ can have two different fixed points with high and low values of k to represent two possible consensuses on A_0 and A_1, respectively. As a general rule, it is desirable to maximize the separation and the domain of attractions of the two fixed points to improve the network resistance against spurious noise.

In the previous section, we have seen how threshold-based message passing algorithms can be used to reliably achieve system-wide message synchronization by using consensus. The model we developed can take into account both node and edge unreliabilities. In this section, we shall show that if the node and edge unreliabilities are eliminated, we can use the same algorithm for the purpose of decoding fairly complex codes that are derived from the Latin and Sudoku squares. Instead of using the XOR operator, these codes use a version of the thresholding operator we used in the previous section.

A Latin square is a $q \times q$ array of numbers that meets two conditions: the numbers 1 to q appear only (1) once on each row and (2) once on each column. A Latin square is called a Sudoku square if $q = r^2$ for some integer r and it meets an additional condition: (3) if the square is divided into r^2 square blocks of $r \times r$ numbers, each block also contains the numbers 1 to q [32]. Figure 5.17a shows an example of the 4×4 Sudoku square.

1	2	3	4
3	4	1	2
2	1	4	3
4	3	2	1

1			
			2
		4	
	3		

FIGURE 5.17 A 4×4 Sudoku square (a) with and (b) without erasures.

One of the interesting properties of a Latin square (and by extension, a Sudoku square) is that the two (or three, for Sudoku) conditions in its definition introduce a large number of interlocking constraints that eliminate many degrees of freedom in assigning the q^2 numbers into the array.

In fact, these constraints are so restrictive that even when some numbers are removed from an otherwise valid Latin *or* Sudoku (LS, for short) square, often these numbers can be recovered. For example, in Figure 5.17b, the twelve numbers that are removed

from Figure 5.17a can be recovered uniquely by process of inference using the original interlocking constraints.

This property is the basis for the Sudoku puzzles. A Sudoku (Latin) puzzle is created from a valid Sudoku (Latin) square by removing numbers from the square such that these numbers can be *uniquely* restored (without this restriction, an empty square would qualify as a "puzzle" with many answers).

Each LS puzzle can also be viewed as a q-ary codeword of length q^2 with erasures. To decode such codewords, in this section we present a new algorithm similar to the iterative algorithm used to decode the Low Density Parity Check (LDPC) codes [33]. The algorithm first converts the q-ary symbols into binary symbols and the missing numbers into erasures. The erasures are then decoded and corrected using a generalized LDPC algorithm.

We prove that our algorithm returns the same answer(s) consistently, even under a randomized mode of operation, invariant to the decoding path used. The algorithm is a list decoder: depending on a "decoding radius", it can recursively return from one up to all codewords within the radius.

5.3.1 Codes, Combinatorial Designs, and Graphs

The $q \times q$ entries of a q-ary Latin square l can be mapped into the symbols of a q-ary codeword c of length q^2. Denote this mapping by C, which maps the set $\mathcal{L}$ of all $q \times q$ Latin squares is mapped into a "q-L" code, and the set $\mathcal{S}$ of all $q \times q$ Sudoku squares (where $q = r^2$) into a "q-S" code. For brevity, we refer to both codes as the q-LS code, or simply the LS code. Under this mapping, missing numbers in the puzzle map to codeword erasures. Solving a puzzle then amounts to correcting these erasures.

The entries of l can also be mapped into the labels of the q^2 vertices (nodes) of a graph g, whose adjacent vertices have different labels. Denote this mapping by G. Two vertices in g are adjacent iff their corresponding entries in l share a row or column. Since both C and G are 1-to-1 mappings, their inverse mappings exist. In fact, $\mathsf{G}\mathsf{C}^{-1}$ maps a q-LS

code into its *graph code*.

Graph codes are among the hottest current research topics in coding theory [33]. Some of them – including many linear ones, and the LDPC codes – perform very close to the Shannon Capacity, an impressive feat considering their conceptual simplicity. The sparsity of the LDPC matrices makes graphs the natural choice of representation for analysis and data structure implementation. Formulated as a graph code, the LS codes can be decoded with a variant of the iterative decoding algorithms used by (LDPC) graph codes.

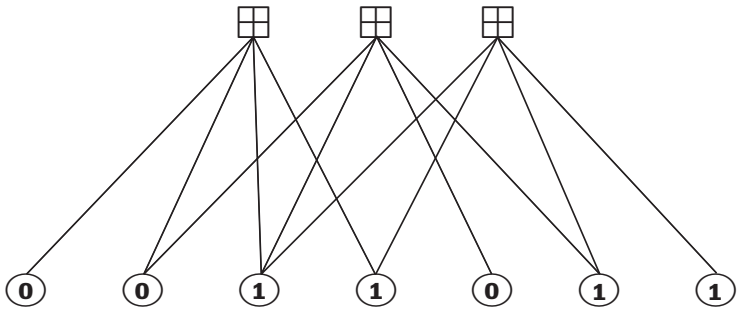


FIGURE 5.18 The Tanner graph for the (7,4) Hamming code.

We now give an example for the standard iterative erasure-correction algorithm using the simple (7,4) Hamming code – formulated as an LDPC graph code – whose graph is shown in Figure 5.18. The graph uses Tanner's [34] convention. The seven circular nodes are variable nodes, each containing one of the seven codeword symbols. The three square nodes are check nodes, one for each parity check equations. Each equation operates on the four outgoing edges, one for each symbol.

The iterative erasure correction algorithm is simple: (1) each check node can correct a single erasure, (2) each correction is propagated to other check nodes, which in turn correct their own single erasures, (3) the process continues until all erasures are corrected (unfortunately, the algorithm may reach a "stopping set" when there are no more *single* erasures for the check nodes).

Suppose symbols 4, 5, and 6 in Figure 5.18 are erased. The algorithm recovers these

symbols as follows. First, check node 1 determines that symbol 4 must be a one and makes a correction, which is propagated to check node 3, which then determines that symbol 6 must be a one, which is propagated to check node 2, which determines that symbol 5 must be a zero. The readers can verify that if symbols 2, 4, and 6 are erased, our algorithm encounters a stopping set. This algorithm can correct any double erasures and many triple erasures (exceeding simple decoders that only correct double erasures).

To use this algorithm for decoding LS codes, we have to make several changes. Why? First, LDPC codes are binary, whereas LS codes are q-ary. Second, the check nodes in LDPC codes require the variable nodes to contain an even (or odd) number of ones. In LS codes, only one variable node can contain a value of one. The next section provides an simple illustrative example.

5.3.2 Decoding Example

For brevity, the example is drawn from the smallest non-trivial 4×4 Sudoku code. Figure 5.19a shows the Sudoku square S with $q = 4$ (and $r = 2$) that we showed earlier in Figure 5.17. The entries S_{ij} of S contain the values from 1 to q that satisfy the Sudoku constraints. For example, $S_{14} = 4$ and $S_{44} = 1$. To the right of S is an $r \times r$ binary subarray S'_{14} corresponding to S_{14}. Each S_{ij} has a corresponding subarray S'_{ij} that contains q cells S'_{ijk}, with $k = 1 \ldots q$.

1	2	3	4	0	0
3	4	1	2	0	1
2	1	4	3		
4	3	2	1		

1	0	0	1	0	0	0	0
0	0	0	0	1	0	0	1
0	0	0	0	1	0	0	1
1	0	0	1	0	0	0	0
0	1	1	0	0	0	0	0
0	0	0	0	0	1	1	0
0	0	0	0	0	1	1	0
0	1	1	0	0	0	0	0

FIGURE 5.19 The (a) q-ary and (b) binary 4×4 Sudoku square

Cell $k = 1$ is on the upper-left corner of the subarray and cell $k = q$ is on the lower-right corner. The cells S'_{ijk} contain all zeros except where $k = S_{ij}$. Replacing the values S_{ij} in S with their $r \times r$ binary subarrays S'_{ij} produces a new $qr \times qr$ square S' with $(qr)^2 - q^2$ zeros and q^2 ones. Performing this operation on the S shown in Figure 5.19a produces S', shown in Figure 5.19b.

Figure 5.20a is derived from Figure 5.19a, showing only S_{11} and the other entries of S related through the Sudoku constraints. Each S_{ij} obeys the three constraints that prevent duplicates (1) across any row (2) down any column (3) in the same $r \times r$ block (Latin squares use the first two constraints). Since S_{11} in S is related to S'_{11} in S', instead of the q-ary values, Figure 5.20b shows S'_{111} and the other related binary values S'_{ijk}.

1	2	3	4
3	4		
2			
4			

1	0	0		0		0	
0	0						
0		0					
0							
0							

FIGURE 5.20 The entries related to S'_{111} through the constraints

Figure 5.20b gives away the algorithm: the entry S'_{111} is controlled by four constraints: h_h, h_v, h_b, and h_c. The subscripts $_{\{h,v,b,c\}}$ indicate that the constraint is operating horizontally, vertically, in a block, and in a cell. Each constraint $h_{\{h,v,b,c\}}$ operates on a set of cells $V(h_{\{h,v,b,c\}})$. For example, in Figure 5.20 we have:

$$V(h_h) = \{S'_{111}, S'_{121}, S'_{131}, S'_{141}\},$$
$$V(h_v) = \{S'_{111}, S'_{211}, S'_{311}, S'_{411}\},$$
$$V(h_b) = \{S'_{111}, S'_{121}, S'_{211}, S'_{221}\},$$
$$V(h_c) = \{S'_{111}, S'_{112}, S'_{113}, S'_{114}\}.$$

Each of the other S'_{ijk}'s also has its own constraints h_h, h_v, h_b, and h_c (possibly different from S'_{111}'s). Denote by $H_h(v)$, $H_v(v)$, $H_b(v)$, and $H_c(v)$ the horizontal, vertical, block, and cell constraints controlling v, and $H(v) = H_h(v) \cup H_v(v) \cup H_b(v) \cup H_c(v)$. How many such constraints are there? Let us count the horizontal constraints first. For a fixed i and k, $H_h(S'_{ijk}) = H_h(S'_{ij'k})$, therefore for all i and k, we have q^2 horizontal constraints. The same is true for the vertical constraints. There are q blocks, each with q constraints for $k = 1 \ldots q$, for a total of q^2 constraints. Finally, each of the q^2 cells has its own constraint. Thus, $4q^2$ constraints control the q^3 entries of S'.

Figure 5.21 shows the four check nodes h_h, h_v, h_b, and h_c for S'_{111}. The eleven variable nodes are shown underneath the circles labeled with their i, j, and k's. From Figure 5.20, $S'_{111} = 1$ and 0 for other values of i, j, and k. Each variable node S'_{ijk} is entangled in the same structure that enforces codeword integrity, i.e., erasing one symbol affects several check nodes.

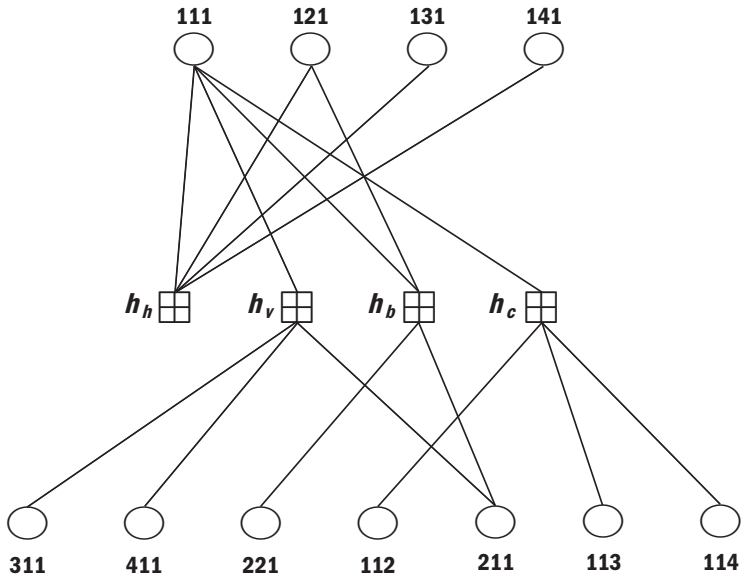


FIGURE 5.21 The check equations h_h, h_v, h_b, and h_c for S'_{111}

We now provide a our own definition for the check nodes, different from the standard LDPCs. Suppose that h operates on a set of q variable nodes $V(h)$, or V for short. Denote by $V_1(h)$ the set of variable nodes with ones, and by $V_0(h)$ those with zeros. Likewise,

denote by $V_e(h)$ the set of variable nodes with erasures. For any h, $|V_e \cup V_0 \cup V_1| = |V| = q$. Define H as a function of h that returns four possible values: *error*, *valid*, *pause*, or *solve* (or e, v, p, and s). The function $\mathsf{H}(h)$ reports the *state* of h. When $\mathsf{H}(h)$ returns an *error*, we say that h is in the *error* state, or h is an *error* node. Whenever $V(h)$ is updated, h is updated unless it is an *error* node.

$$\mathsf{H}(h) = \begin{cases} \textit{error} & \text{if } |V_1| > 1 \text{ or } |V_0| = q \\ \textit{valid} & \text{if } |V_1| = 1 \text{ and } |V_0| = q-1 \\ \textit{pause} & \text{if } |V_e| > 1 \text{ and } |V_1| = 0 \\ \textit{solve} & \text{if } |V_e| = 1 \text{ or } |V_1| = 1. \end{cases} \tag{5.11}$$

If S is a valid Sudoku square (and thus a valid codeword), then all its $4q^2$ check nodes are *valid* nodes. As entries are erased, some affected check nodes become *solve* nodes. As more entries are erased, the number of *pause* nodes increase. A decodeable codeword (or a solvable puzzle) leaves enough unerased entries to recover the numbers uniquely. Erasure correction starts from the *solve* nodes, iteratively attempting to convert all check nodes to *valid* nodes.

Figures 5.22 and 5.23 show a codeword S and its binary array S' with $qr \times qr$ variable nodes. Surrounding S' are four groups of $4q^2$ check nodes. Each block has q copies of $r \times r$ subarrays, each indexed by $k = 1 \ldots q$.

Directly above S' is the H_v group. The group has four subarrays $j = 1 \ldots 4$, from left to right. The kth cell in subarray j contains $H_v(S'_{1jk})$. To the right of S' is the H_h group with its four subarrays $i = 1 \ldots 4$, from top to bottom. The kth cell in subarray i contains $H_h(S'_{i1k})$. To the right is the square H_b group with its subarrays arranged from top left corner to the bottom-right corner. The kth node in subarray i controls all the kth cells in group i of S'. Finally, we have the H_c group, whose (i,j)th *cell* contains $H_c(S'_{ijk})$. Having established these definitions, the decoding process can now start.

1		3	4
2		4	
	3		1

FIGURE 5.22 One example of a Sudoku puzzle

Figure 5.23 shows the initial states of all the check nodes. In this Figure, there is no *valid* node, only *pause* nodes and *solve* nodes. For example, $H_h(S'_{411})$ and $H_h(S'_{413})$ are both *solve* nodes because S'_{423} and S'_{441} contain ones.

1	0			0	0	0	0
0	0			1	0	0	1
0	1			0	0		
0	0			0	1		
		0	0			1	0
		1	0			0	0

s	s
s	s
p	p
p	p
p	s
p	s
s	p
s	p

s	p	p	p
p	p	s	s
p	s	s	p
s	p	p	s

v	p	v	v
p	p	p	p
v	p	v	p
p	v	p	v

FIGURE 5.23 The variable and check nodes

In the first iteration, all *solve* nodes attempt to correct their single erasures, update their states to *valid*, and propagate the corrections to other check nodes. The result is shown in Figure 5.24. As expected, the set of *solve* and *valid* nodes grows, while the set of *pause* nodes shrinks.

This process is repeated in the next iteration. After the second iteration, the graph contains only five erasures, as shown in Figure 5.25. Finally, after the third iteration, all variable nodes are restored to their original values as shown in Figure 5.26. First seen as a hindrance to correcting the erasures, the four constraints on each variable node actually help in erasure correction.

Although this example only covers a 4×4 Sudoku square, the concepts can be applied to larger LS squares. Latin square decoders have no block check nodes — only the other $3q^2$ check nodes — and cannot be visualized as easily as the Sudoku decoders because q is not always a square power of an integer.

s	s	s	p	s	p	s	p
s	p	s	p	s	s	s	s

1	0	0		0	0	0	0
0	0			1	0	0	1
0		0					
				0	0	0	0
0	1		0	0	0	0	
0	0	0		0	1		0
	0	0	0	0		1	0
0		1	0		0	0	0

v	s
s	s
p	p
p	p
s	s
s	s
s	s
s	s

v	p	p	p
p	p	v	v
p	v	v	p
v	p	p	v

v	p	v	v
p	p	p	p
v	p	v	p
p	v	p	v

FIGURE 5.24 The puzzle after the first iteration

1	0	0	1	0	0	0	0
0	0	0	0	1	0	0	1
0	0	0		1		0	
1		0		0	0	0	0
0	1	1	0	0	0	0	0
0	0	0	0	0	1	1	0
0	0	0	0	0	1	1	0
0	1	1	0	0	0	0	0

v	v
v	v
v	p
v	p
v	v
v	v
v	v
v	v

v	s	v	p
v	p	v	v
v	v	v	v
v	v	v	v

v	v	v	v
s	p	s	s
v	v	v	v
v	v	v	v

v	v	v	s	v	s	v	s
v	s	v	s	v	v	v	v

FIGURE 5.25 The puzzle after the second iteration

1	0	0	1	0	0	0	0
0	0	0	0	1	0	0	1
0	0	0	0	1	0	0	1
1	0	0	1	0	0	0	0
0	1	1	0	0	0	0	0
0	0	0	0	0	1	1	0
0	0	0	0	0	1	1	0
0	1	1	0	0	0	0	0

v	v
v	v
v	v
v	v
v	v
v	v
v	v
v	v

v	v	v	v
v	v	v	v
v	v	v	v
v	v	v	v

v	v	v	v
v	v	v	v
v	v	v	v
v	v	v	v

v	v	v	v	v	v	v	v
v	v	v	v	v	v	v	v

FIGURE 5.26 The puzzle after the third and final iteration

The iterative algorithm we just described inherits the same limitations as the one used for decoding erased LDPC codewords: some puzzles contain stopping sets that halt the algorithm prematurely.

For example, the erasure pattern in Figure 5.27a poses no problem, but the one in Figure 5.27b has a stopping set. Once stopped, the algorithm can continue by selecting from the (q-ary) S the block, row, or column with the fewest number of erasures e and try all the $e!$ possible completion configurations. For each configuration, the algorithm runs until it encounters an error.

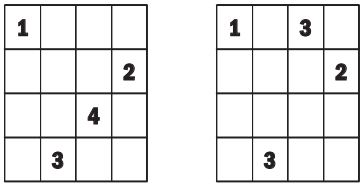


FIGURE 5.27 Squares without and with stopping set

In the best case, one configuration leads to a solution, while all others produce errors. In the worst case, more than one configuration leads to solution. In some case, it finds another smaller stopping set, from which the process is repeated recursively until a maximum number of solutions is found.

5.3.3 Algorithm

In this section we provide a detailed description of the two procedures that form our iterative decoding. The procedure in Listing 1 iteratively corrects as many erasures as possible given a starting *context* denoted by $X = (S, V, E, P, C)$, defined as a collection of four lists S, V, E and P which contain *pointers* to the *solve*, *valid*, *error*, and *pause* nodes, respectively, along with the list C that contains the values (zeros, ones, and erasures) of the q^3 variable nodes. In Listing 1, the notation $V \leftarrow h$ indicates that a check node h is moved from its current list into V. The notation $H \leftarrow S$ means the whole content of the list S is moved into the list H.

The procedure SOLVE simply calculates a modified context and a return value. It does not handle a stopping set. A very important question is: can we prove that SOLVE returns a "unique" answer regardless of the order in which the elements $h \in H$ are processed on line 4?

Before proving the uniqueness of the result of SOLVE, let us establish some notations and definitions. Let V_1 and V_2 denote two sets of variable nodes, whose binary contents are represented by the vectors $\vec{V}_1$ and $\vec{V}_2$, respectively. Let H_1 and H_2 denote two sets of check nodes whose four possible states are stored in the vectors $\vec{H}_1$ and $\vec{H}_2$, respectively.

Definition 1. The sets V_1 and V_2 (H_1 and H_2) are *equal* if and only if they contain the same variable (check) nodes. The two vectors $\vec{V}_1$ and $\vec{V}_2$ ($\vec{H}_1$ and $\vec{H}_2$) are *equal* if and only if their elements are identical. □

Listing 1: Procedure for iteratively removing erasures

```
1:  procedure SOLVE (S, V, E, P, C)
2:      while S ≠ ∅ do                                  ▷ Iterate over all solve nodes
3:          H ← S                                       ▷ Move solve nodes to a local list
4:          for all h ∈ H do                            ▷ For all solve nodes
5:              CORRECT( V_e(h) ⊂ C )                   ▷ Unerase V_e(h)
6:              V ← h                                   ▷ and move to the valid list
7:              H' = {h' | V(h') ∩ V_e(h) ≠ ∅}
8:              for all h' ∈ H' do                      ▷ update their states
9:                  {V | S | E | P} ← h'
10:             end for                                 ▷ While solve node exists
11:         end for
12:     end while
13:     if E ≠ ∅ then r v ← −1                          ▷ We have an error
14:     else if P = ∅ then r v ← 1                      ▷ Solution found
15:     else r v ← 0                                    ▷ We have run into a stopping set
16:     return ( S, V, E, P, C, r v )
17: end procedure
```

Suppose $h_1, h_2 \in H$ are two *solve* nodes waiting to be processed on line 4 of Listing 1. Denote by $V_1 = V_e(h_1)$ and $V_2 = V_e(h_2)$ the sets of erasures connected to (and eventually corrected by) h_1 and h_2. Denote by $H_1 = H(V_1)$ and $H_2 = H(V_2)$ the check nodes connected to V_1 and V_2; by $\vec{V}_{1^0}$ and $\vec{V}_{2^0}$ the contents of $\vec{V}_1$ and $\vec{V}_2$ *before* h_1 and h_2 are processed on line 4; by $\vec{V}_{1^{12}}$ and $\vec{V}_{2^{12}}$ the contents *after* h_1 and h_2 are processed (in that order), and by $\vec{V}_{1^{21}}$ and $\vec{V}_{2^{21}}$ the contents *after* h_2 and h_1 are processed. Denote by $\vec{H}_{1^0}$, $\vec{H}_{2^0}$, $\vec{H}_{1^{12}}$, $\vec{H}_{2^{12}}$, $\vec{H}_{1^{21}}$ and $\vec{H}_{2^{21}}$ the counterparts for $\vec{H}_1$ and $\vec{H}_2$. Finally, define $V_\cap = V_1 \cap V_2$ and $H_\cap = H_1 \cap H_2$, along with $\vec{V}_\cap$, $\vec{H}_\cap$, $\vec{V}_{\cap^0}$, $\vec{H}_{\cap^0}$, $\vec{V}_{\cap^{12}}$, $\vec{H}_{\cap^{12}}$, $\vec{V}_{\cap^{21}}$ and $\vec{H}_{\cap^{21}}$.

Listing 2: Recursive removal of stopping sets

```
1:  procedure DECODE (X, level)
2:      if level = 1 then
3:          Store the received codeword into Y
4:      else
5:          x ∈ X is the row / col / block with fewest erasures
6:          Store all e! configurations of x in Y
7:      end if
8:      for all y ∈ Y do
9:          ( X', rv ) = SOLVE( y )
10:         if rv = 1 then SOL ← SOL ∪ C
11:         if |SOL| > maxsol then exit
12:         if rv = 0 then DECODE( X', level + 1 )
13:     end for
14: end procedure
```

Definition 2. The procedure SOLVE returns a *unique* answer with respect to h_1 and $h_2 \in H$ on line 4 in SOLVE if and only if on line 11, these conditions are met: (a) both $\vec{H}_{\cap 12}$ and $\vec{H}_{\cap 21}$ contain at least one *error* node, or (b) $\vec{H}_{\cap 12} = \vec{H}_{\cap 21}$ and $\vec{V}_{\cap 12} = \vec{V}_{\cap 21}$. The procedure SOLVE returns a *unique* answer if it returns a unique answer with respect to any pair h_1 and $h_2 \in H$. □

Theorem 13. SOLVE returns a unique answer.

PROOF: We can prove the theorem for any pair h_1 and h_2 by considering every possible cardinality and interconnectivity of $V_\cap$ and $H_\cap$ and combinations of their corresponding vector values. Fortunately, we only need to consider the three canonical configurations shown in Figure 5.28. Our proof applies to other configurations that are unions of these canonical configurations. The middle nodes are in $V_\cap$, and the bottom nodes are in $H_\cap$.

Due to its topology, the third configuration automatically meets the conditions (a) and (b). Table 1 summarizes how the first and second configurations meet conditions (a) and (b). The first two columns in the table are the possible *solve* states for h_1 and h_2. The notation s_0 means that the check node infers that the erased variable node(s) should

be zero(s), and vice versa with s_1. Other states are abbreviated with their first initials in Equation 5.11.

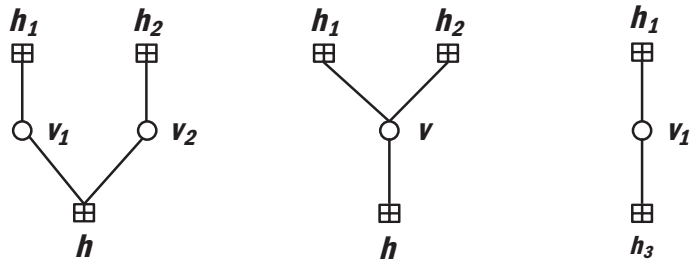


FIGURE 5.28 The three canonical configurations

In Figure 5.28a, h is connected to v_1 and v_2 and $q-2$ other variable nodes not shown on the diagram. In Figure 5.28b, h is connected to v and $q-1$ other variable nodes. Initially, v_1, v_2, and v all contain erasures (otherwise h_1 and h_2 are not $solve$ nodes). Denote by V_h^- the $q-2$ (or $q-1$) variable nodes connected to h. The vector $\vec{V}_h^-$ is the initial state of h (columns 4 and 11). Column 3 of Table 1 describes $\vec{V}_h^-$ in shorthand notation. The $+$ sign after a 0 means "one or more" and a $+$ after an e (for erasure) means "two or more". For example, 0_+1 means h is connected to $\vec{V}_h^-$ with one or more variable nodes with zeros, one node with a one, plus v_1 and v_2 in Figure 5.28a (or plus v in Figure 5.28b).

The column groups labeled 12 (or 21) show the final contents of $\vec{H}_\cap$ and $\vec{V}_\cap$ after h_1 and h_2 (or h_2 and h_1) are processed, in that order. For the second configuration, the intermediate state of h_2 (or h_1) right after h_1 (or h_2) is processed is also shown because in the second configuration, any change by h_1 (or h_2) on v affects h_2 (or h_1), raising the possibility of an error at this step.

Using definition 2, the procedure SOLVE returns a unique answer if for each configuration: (a) both the 12 and 21 groups contain at least one *error* state, or (b) the contents of the 12 and 21 groups are identical. Comparing columns 5–7 to 8–10 (or 12–14 to 15–17) in Table 1 confirms that configurations 1 and 2 meet conditions (a) and (b). Thus SOLVE returns a unique answer. □

			Configuration 1							*Configuration* 2						
			0	12			21			0	12			21		
h_1	h_2	V_h^-	h	v_1	v_2	h	v_1	v_2	h	h	v	h_2	h	v	h_1	h
s_0	s_0	0_+	p	0	0	e	0	0	e	s_1	0	v	e	0	v	e
s_0	s_0	0_+e	p	0	0	s_1	0	0	s_1	p	0	v	s_1	0	v	s_1
s_0	s_0	0_+e_+	p	0	0	p	0	0	p	p	0	v	p	0	v	p
s_0	s_0	0_+1	s_0	0	0	v	0	0	v	s_0	0	v	v	0	v	v
s_0	s_0	0_+1e	s_0	0	0	s_0	0	0	s_0	s_0	0	v	s_0	0	v	s_0
s_0	s_0	0_+1e_+	s_0	0	0	s_0	0	0	s_0	s_0	0	v	s_0	0	v	s_0
s_0	s_1	0_+	p	0	1	v	0	1	v	s_1	0	e	v	1	e	v
s_0	s_1	0_+e	p	0	1	s_0	0	1	s_0	p	0	e	s_0	1	e	s_0
s_0	s_1	0_+e_+	p	0	1	s_0	0	1	s_0	p	0	e	s_0	1	e	s_0
s_0	s_1	0_+1	s_0	0	1	e	0	1	e	s_0	0	e	e	1	e	e
s_0	s_1	0_+1e	s_0	0	1	e	0	1	e	s_0	0	e	e	1	e	e
s_0	s_1	0_+1e_+	s_0	0	1	e	0	1	e	s_0	0	e	e	1	e	e
s_1	s_0	0_+	p	1	0	v	1	0	v	s_1	1	e	e	0	e	e
s_1	s_0	0_+e	p	1	0	s_0	1	0	s_0	p	1	e	s_1	0	e	s_1
s_1	s_0	0_+e_+	p	1	0	s_0	1	0	s_0	p	1	e	p	0	e	p
s_1	s_0	0_+1	s_0	1	0	e	1	0	e	s_0	1	e	v	0	e	v
s_1	s_0	0_+1e	s_0	1	0	e	1	0	e	s_0	1	e	s_0	0	e	s_0
s_1	s_0	0_+1e_+	s_0	1	0	e	1	0	e	s_0	1	e	s_0	0	e	s_0
s_1	s_1	0_+	p	1	1	e	1	1	e	s_1	1	v	v	1	v	v
s_1	s_1	0_+e	p	1	1	e	1	1	e	p	1	v	s_0	1	v	s_0
s_1	s_1	0_+e_+	p	1	1	e	1	1	e	p	1	v	s_0	1	v	s_0
s_1	s_1	0_+1	s_0	1	1	e	1	1	e	s_0	1	v	e	1	v	e
s_1	s_1	0_+1e	s_0	1	1	e	1	1	e	s_0	1	v	e	1	v	e
s_1	s_1	0_+1e_+	s_0	1	1	e	1	1	e	s_0	1	v	e	1	v	e

Table 1: The first two configurations

Having proven that SOLVE returns a unique answer regardless of the order in which the *solve* nodes are processed, we address the issue of handling the stopping sets. SOLVE is called (recursively) by another procedure called DECODE, shown in Listing 2. The recursion is initiated by executing DECODE(X,1). The parameter *maxsol* controls the maximum number of results returned by DECODE, making it a variable *list decoder*.

Line 5 requires explanation: the row, column, block, and erasures mentioned there refer to the q-ary S, not the binary S'. For instance, if the first row has only three erasures, while the other rows, columns, and blocks have more erasures, then x refers to the first

row. If $q = 9$ and the missing numbers in x are 3, 4, and 7, then there are 3! possible ways to place these numbers into x, each potentially leading to a solution, which is stored in Y. Each $y \in Y$ is then passed as a context to SOLVE, which tries to find a solution within y.

In conclusion, in this paper we have presented an iterative algorithm that can be used to decode (and solve) the Latin and Sudoku codes (and puzzles). We proved that the algorithm returns consistent solutions regardless of the path taken to compute them. An interesting question for future research is whether the same decoding algorithm can be adapted to solve Sudoku puzzles with symbol errors (instead of erasures).

BIBLIOGRAPHY

[1] R. O'Dell, R. Wattenhofer, "Information dissemination in highly dynamic graphs," *DIALM-POMC 2005*, Cologne, Germany 2005.

[2] H. Dubois-Ferriere, et al., "Age matters: efficient route discovery in mobile ad-hoc networks using encounter ages," *Proc. of MobiHoc 2004*.

[3] E. Gafni, D. Bertsekas, "Distributed algorithms for generating loop-free routes in networks with frequently changing topology," *IEEE Trans. Comm.*, vol. 29, no. 1, 1981.

[4] Z.J. Haas, M.R. Pearlman, P. Samar, "ZRP: A hybrid framework for routing in ad-hoc networks," *Ad Hoc Networking*, Addison-Wesley, 2001.

[5] D.B. Johnson, et al., "DSR: The dynamic source routing protocol for multi-hop wireless ad hoc networks", *Ad Hoc Networking*. Addison-Wesley, 2001.

[6] F. Kuhn, R. Wattenhofer, Y. Zhang, A. Zollinger, "Geometric ad-hoc routing: Of theory and practice," *Proceedings of PODC, 2003*.

[7] A. Lindgren, A. Doria, O. Schelen, "Probabilistic Routing in Intermittently Connected Network," *Proceedings of SAPIR 2004*, Brazil, 2004.

[8] C.E. Perkins, E.M. Royer. "Ad-hoc on-demand distance vector routing," *Proceedings of WMCSA 1999*.

[9] P. Samar, et al., "Independent Zone Routing: An adaptive hybrid routing framework for ad hoc wireless networks," *IEEE/ACM TON*, vol. 12, no. 4, 2004.

[10] A. Vahdat, D. Becker. "Epidemic routing for partially connected ad-hoc networks," *Technical Report CS-200006*, Duke University, April 2000.

[11] L. Lamport, R. Shostak, M. Pease, "The Byzantine generals problem," *ACM Trans. Program. Lang. Syst. 4,* vol. 3, pp. 382–401.

[12] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci, "A Survey on Sensor Networks," *IEEE Communications Magazine*, August 2002.

[13] F. Zhao, L. Guibas, *Wireless Sensor Networks: An Information Processing Approach*, Morgan Kaufmann, 2004.

[14] M. Raynal, *Distributed Algorithms and Protocols*, John Wiley, 1988.

[15] R. Olfati-Saber, R.M. Murray, "Consensus problems in networks of agents with switching topology and time-delays." *The IEEE Transactions on Automatic Control*, vol. 49, no. 9, pp. 1520–1533, September 2004.

[16] R. Olfati-Saber, R.M. Murray, "Consensus protocols for networks of dynamic agents," *Proc. of the Amer. Cont. Conf.*, vol. 2, pp. 951–956, June 2003.

[17] R. Olfati-Saber, R.M. Murray, "Flocking with Obstacle Avoidance: Cooperation with Limited Communication in Mobile Networks." *Proc. of the 42nd IEEE Conf. on Dec. and Control*, vol. 2, pp. 2022–2028, Dec. 2003.

[18] W. Ren, R.W. Beard, "Consensus of Information Under Dynamically Changing Interaction Topologies." *Proc. of the Amer. Cont. Conf.*, 2004.

[19] A. Tahbaz-Salehi, A. Jadbabaie, "On Consensus Over Random Networks", *Proc. of the 44th Annual Allerton Conference*, Sept 2006.

[20] S.C. Wang, Y.H. Chin, K.Q. Yan, "Byzantine Agreement in a Generalized Connected Network," *IEEE Trans. Par. and Dist. Systems.*, vol. 6, pp. 420–427, April 1995.

[21] O. Babaoglu, R. Drummond, "Streets of Byzantium: Network architectures for fast reliable broadcasts," *IEEE Trans. Software Eng.*, vol. SE-11, pp. 546–554, June 1985.

[22] 0. Babaoglu, P. Stephenson, R. Drummond, "Reliable broadcasts and communication models: Tradeoffs and lower bounds," *Dist. Computing*, vol. 2, pp. 177–189, 1988.

[23] A. Bar-Noy, D. Dolev, C. Dwork, R. Strong, "Shifting gears: Changing algorithms on the fly to expedite Byzantine Agreement," *Proc. Symp. Principles Dist. Computing*, pp. 42–51, 1987.

[24] K.M. Chandy, J. Misra, *Parallel Program Design: A Foundation*. Addison-Wesley, Reading, MA, 1988.

[25] D. Dolev, R. Reischuk, "Bounds on information exchange for Byzantine Agreement," *JACM*, vol. 32, no. 1, pp. 191–204, Jan. 1985.

[26] M. Fischer, "The consensus problem in unreliable distributed systems (a brief survey)," *Lecture notes in computer science, in Proc. 1983 Int. FCT-Con.* Borgholm, Sweden, pp. 127–140, Aug. 1983.

[27] B.M. McMillin, L.M. Ni, "Byzantine fault-tolerance through application oriented specification," *Proc. COMPASAC87*, pp. 347–353, 1987.

[28] R. Reischuk, "A new solution for the Byzantine generals problem," *IBM Res. Rep.*, RJ–3673, Computer Science, 1982.

[29] R. Turpin, B. Coan, "Extending binary Byzantine agreement to multivalued Byzantine agreement", *Process, Lett.*, vol. 18, pp. 73–76, 1984.

[30] R.C. Read, R.J. Wilson, *An Atlas of Graphs*. Oxford University Press, Oxford, England, 1998.

[31] N.J.A. Sloane, *The On-Line Encyclopedia of Integer Sequences,* Sequences A006820/M1617, A033301, and A033483.

[32] Contributors, "Sudoku", *Wikipedia, The Free Encyclopedia*, `http://en.wikipedia.org/w/ index.php?title=Sudoku`, January 2006.

[33] D.J.C. MacKay, *Information Theory, Inference, and Learning Algorithms*, Cambridge University Press, 2003.

[34] R.M. Tanner, "A Recursive Approach to Low Complexity Codes", *IEEE Trans. on Information Theory*, vol. 27, no. 5, pp. 533–547, Sept 1981.

6

Additional Topics

THIS chapter contains two sections that discuss additional topics in the area of (1) error-correcting codes and their relationships to combinatorial structures, and (2) application of message-passing algorithms in games of perfect information.

In Chapter 5 we discussed the close relationship between binary LDPC codes and Latin (or Sudoku) Squares. In the first section, we will discuss the close relationship between MDC codes and ternary Latin hypersquares. We prove that every q-ary MDS code of length n and distance 2 can be uniquely represented by a Latin square of order q. By counting the total number of the latter, we count the total number of the former.

In our analysis, we define a Latin hypersquare as a generalization of a Latin square. Suppose we represent each letter in the ternary alphabet by three colored cubes: white, grey, and black. Hypercubes of dimensions 0, 1, 2, and 3 can be represented by a single cube, an array of three cubes, a 3×3 square of nine cubes, and a $3 \times 3 \times 3$ block of 27 cubes, as shown in Figure 6.1.

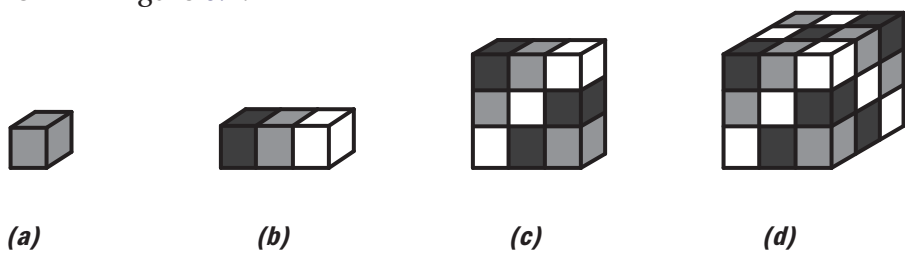


FIGURE 6.1 Hypercubes of different dimensions

One unifying characteristic of these ternary hypercubes is that with the exception of the 0-dimensional hypercube, every linear array of three adjacent cubes have three unique colors. Without going into too much detail, this aspect of the hypercubes can be very easily seen in Figure 6.2.

Figure 6.2(a) shows the same three-dimensional hypercube shown previously in 6.1. In Figure 6.2(b), we show the three slices of this hypercube that are obtained by cutting the hypercube along its height. In turn, each slice is a square. These squares are such that in every row and column, the three cubes all have different colors. Again in Figure 6.2(c), the hypercube is sliced along the width direction, and finally in Figure 6.2(d), the cut is along the depth direction.

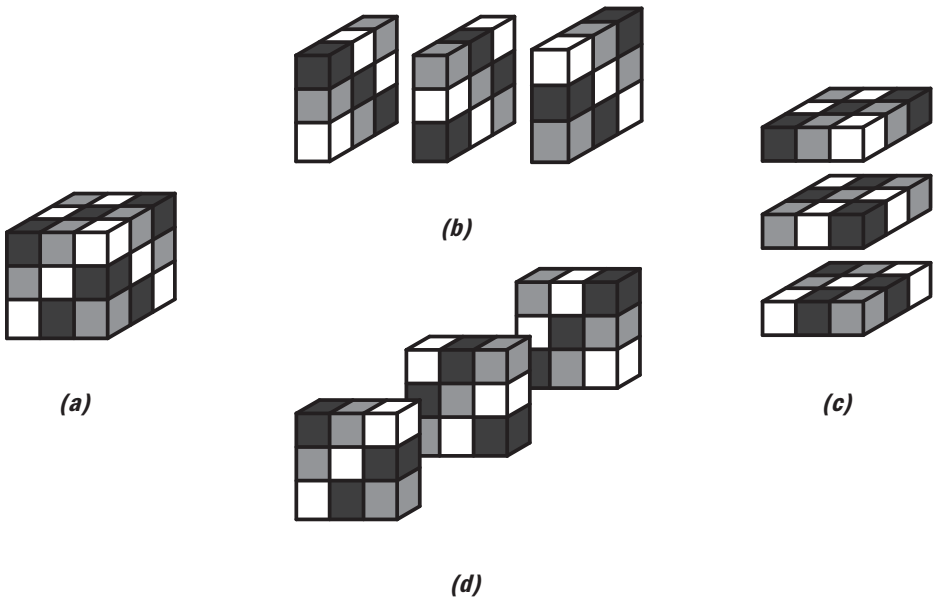


FIGURE 6.2 Three different ways to slice a hypercube

In the final section we discuss the application of message-passing algorithms in games of perfect information. In particular, we focus on the popular game Tic-Tac-Toe, which has enough complexity to illustrate the benefits of message-passing, and enough simplicity to keep the presentation within a manageable size. Our message-passing algorithm uses a hierarchical, and at the same time distributed decision-making structure

where decisions are made locally by using simple rules before being presented to a central arbiter. The arbiter, which itself has no access to the detailed rules, makes a global decision by sorting the local decisions using a simple ordering rule.

Compared to conventional implementations of an optimal algorithm using game tree search, recursive backtracking, and other types of heuristics, our approach unexpectedly has a much lower level of complexity in terms of the types of operations used and the number of steps in the algorithm. To achieve the game objective, the algorithm only uses a local function that counts the number of pieces of different types in neighboring cells and returns the appropriate decision value using `if` statements.

6.1 Latin Hypercubes and MDS Codes

Maximum distance separable (MDS) codes have special properties that give them excellent error-correcting capabilities. Counting the number of q-ary MDS codes of length n and distance d, denoted by $D_q(n,d)_{MDS}$, is a very hard problem. This section shows that for $d = 2$, it amounts to counting the number of $(n-1)$-dimensional Latin hypercubes of order q. Thus $D_q(3,2)_{MDS}$ is the number of Latin squares of order q, which is known only for a few values of q. This section proves constructively that $D_3(n,2)_{MDS} = 6 \cdot 2^{n-2}$.

6.1.1 Introduction

The binary parity check code is a popularly known error detection code: by adding one redundant parity bit that increases the minimum Hamming distance d from 1 to 2, a set of distinct 2^k binary strings of length k become binary parity check codewords of length $n = k+1$. This means that any two codewords in the set differ in at least two positions.

The binary parity check code is a primary example of a special class of codes called

the *Maximum Distance Separable (MDS)* codes, which have many special properties [3].

First, they obey the Singleton theorem [4], which requires the code's minimum distance not to exceed the number of its redundant symbols r by more than one. For the binary parity check code, $d = r+1 = 2$. Second, they map all the q^k elements of of q-ary strings of length k into codewords. The binary parity check code also has this property. As a result, MDS codewords populate the code space very uniformly with maximum distance possible — two very desirable properties for error-correcting codes.

What is the number $D_q(n,d)_{MDS}$ of q-ary MDS codes of length n and minimum distance d ? For $q = 2$, it is well known that if $d > 2$, $D_2(n,d)_{MDS} = 0$ for $n > d+1$, while $D_2(n,d)_{MDS} = 2$ for $n = d+1$ (the repetition code), and $D_2(n,2)_{MDS} = 2$ (the even and odd parity check codes). We prove that in general,$D_q(n > d+1, d > q)_{MDS} = 0$.

The number $D_q(n,d)_{MDS}$ is hard to calculate. This section shows that $D_q(n,2)_{MDS}$ is the number of $(n-1)$-dimensional Latin hypercubes of order q. Hence $D_q(3,2)_{MDS}$ is the number of Latin squares of order q, which is known only for a few q's [2]. No closed form formula is known and in fact, for $q = 6,\ldots,11$, $D_q(3,2)_{MDS}$ has $9,14,21,28,37$, and 48 *digits* ! Is there any hope at all for calculating $D_{q>2}(n,d)_{MDS}$?

This section shows that for ternary codes, $D_3(n,2)_{MDS}$ has an unexpectedly simple closed form formula of $6 \cdot 2^{n-2}$. The proof outlined here is constructive, providing a simple procedure to build ternary MDS codes of length n and Hamming distance 2.

6.1.2 Definitions

We begin by defining a hypercube. While we can extend the definition for Latin cubes in [1], we recursively define a hypercube of dimension n in terms of hypercubes of lower dimensions because the proofs of our main results use this recursively defined structure.

Definition 3. A dimension-n *hypercube* (or an n-cube) $a^n \in \mathcal{H}^n$ over the q-ary alphabet

$\mathcal{Q} = \{0, 1, \ldots, q-1\}$ is a multidimensional array defined recursively as follows:

$$\begin{array}{lcl} a^n & = & [\ b_0{}^{n-1} \quad b_1{}^{n-1} \quad \ldots \quad b_{q-1}{}^{n-1}\ ] \qquad b_0{}^{n-1}, \ldots, b_{q-1}{}^{n-1} \in \mathcal{H}^{n-1} \\ a^1 & = & [\ b_0 \quad b_1 \quad \ldots \quad b_{q-1}\ ] \qquad b_0, \ldots, b_{q-1} \in \mathcal{Q} \end{array} \tag{6.1}$$

where $\mathcal{H}^n$ denotes the space of all dimension-n hypercubes. The elements of $\mathcal{H}^n$ are arrays containing q hypercubes of dimension $n-1$. Hypercubes of dimension 0 are simply alphabet symbols in $\mathcal{Q}$. We denote $b_0{}^{n-1}$ by $a^n[0]$, i.e., the 0-th element of a^n, and the q^n elements of a^n by $a^n[i_1][i_2]\ldots[i_n]$, or simply $a^n[1,2,\ldots,i_n],\ i_j \in \mathcal{Q}$. □

Example 4. If $q = 3$, then $\mathcal{Q} = \{0,1,2\}$ and $\mathcal{H}^n$ is the space of ternary hypercubes. Examples of 1-cubes (strips) are $a^1 = [011]$, $b^1 = [201]$, and $c^1 = [222]$. From Definition 6.1, we can construct 2-cubes (squares) by combining the strips into an array. For example:

$$d^2 = [c^1\, a^1\, c^1] = [[222][011][222]],$$
$$e^2 = [a^1\, c^1\, b^1] = [[011][222][201]],$$
$$f^2 = [a^1\, b^1\, c^1] = [[011][201][222]].$$

To build any square, we can always select a group of 9 symbols from $\mathcal{Q}$. For example, we can build the square $g^2 = [[012][201][120]]$. □

Two hypercubes a^n and b^n are *equal* if their elements are identical in all positions, i.e., $a^n[1,2,\ldots,i_n] = b^n[1,2,\ldots,i_n]$. Two hypercubes are *different* if they differ in one or more positions, and *orthonormal* if they differ in every position. For the same reason we defined hypercubes recursively, we define orthonormality recursively as follows:

Definition 4. Two n-cubes a^n and b^n are *orthonormal* ($\perp$) if the hypercubes of dimen-

sion $n-1$ that form a^n and b^n are also pairwise orthonormal.

$$\begin{aligned} a^n \perp b^n \quad &\Leftrightarrow \quad a^n[i] \perp b^n[i] \quad i = 0 \dots q-1 \quad \text{and} \quad n > 1 \\ a^1 \perp b^1 \quad &\Leftrightarrow \quad a^1[i] \neq b^1[i] \quad i = 0 \dots q-1 \end{aligned} \tag{6.2}$$

The recursive definition is terminated by defining the orthonormality between a^1 and b^1 as simple pairwise inequalities between their elements.

Example 5. From example 4, $a^1 \perp c^1$. None of the 2-cube pairs are orthonormal to each other. For example, $d^2 \not\perp e^2$ because while $d^2[0] = c^1 \perp a^1 = e^2[0]$ and $d^2[1] = a^1 \perp c^1 = e^2[1]$, note that $d^2[2] = c^1 \not\perp b^1 = e^2[2]$. □

We are now ready to define Latin hypercubes. The recursive definitions for hypercubes and orthonormality lead to a simple definition for Latin hypercubes. To illustrate why a recursive definition works, consider a $3 \times 3 \times 3$ Latin cube. The three "slices" along the height, width, and depth, are orthonormal 3×3 squares. Along the height and width of each square, there are three 3×1 orthonormal strips, which themselves contain no identical elements. Precisely, a Latin hypercube is defined as follows:

Definition 5. A dimension-n *Latin hypercube* (or a Latin n-cube) $a^n \in \mathcal{L}^n$ over the q-ary alphabet $\mathcal{Q}$ is defined recursively as follows:

$$a^n \in \mathcal{L}^n \quad \Leftrightarrow \quad a^n[i] \perp a^n[j] \quad i \neq j, \quad 0 \leq i, j \leq q-1 \quad . \tag{6.3}$$

Here $\mathcal{L}^n$ is the space of all dimension-n Latin hypercubes. We call the elements of $\mathcal{L}^1$, $\mathcal{L}^2$, and $\mathcal{L}^3$ Latin strips, squares, and cubes, respectively. □

Definition 6. Two Latin n-cubes $a^n, b^n \in \mathcal{L}^n$ are *orthonormal Latin cubes* if and only if $a^n \perp b^n$. Note that for dimension $n = 2$, the definition should not be confused with the standard definition for *orthogonal* Latin squares of order q. In this section, we use the symbol n to denote the cube dimension, not the order, which we denote by q. □

Example 6. Using the same cubes defined in Example 4, we can see that b^1 and g^2 are Latin strips and squares, respectively, while the others are not. We display the strips a^1, b^1, and c^1; and the squares d^2, e^2, f^2, and g^2 visually below to show why this is so.

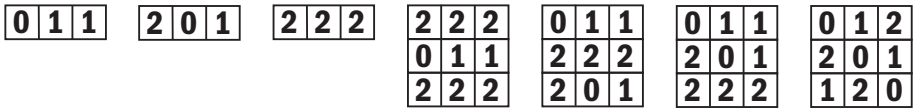


FIGURE 6.3 Different strips and squares

Clearly, a^1 and c^1 are not in $\mathcal{L}^1$ because they have identical elements 1 and 2, respectively. In contrast, the elements of b^1 are not identical, thus $b^1 \in \mathcal{L}^1$. Continuing with d^2, we see that all its rows have identical elements. Rows 1 and 3 also have identical values at all three positions, therefore, $d^2 \notin \mathcal{L}^2$. For e^2, row 3 is not orthonormal with the first two rows, and row 2 contains identical elements, therefore $e^2 \notin \mathcal{L}^2$.

We do not expect a different result with f^2, since its rows are simply the rows of e^2 permuted. This brings us to the last square g^2, which we claim to be a Latin square. The rows (and columns) do not contain identical elements, and the rows are orthonormal to each other. Therefore, $g^2 \in \mathcal{L}^2$. □

Thinking of a $3 \times 3 \times 3$ cube as a stack of three 3×3 squares, we can "shuffle" the stack in three different ways along its height, width, and depth. Obviously, we want to go beyond three different ways of shuffling. The following definition generalizes the concept of shuffling to dimension-n hypercubes.

Definition 7. The i-th coordinate *left-cyclic shift* operator (denoted by $\blacktriangleleft^i$) of an n-cube over $\mathcal{Q}$ are recursively defined as follows, with $j \in \{0 \ldots q-1\}$:

$$\begin{aligned} b^n = \blacktriangleleft^i a^n \quad &\Leftrightarrow \quad b^n[j] = \blacktriangleleft^i a^n[j] \qquad 1 \le i < n \\ b^n = \blacktriangleleft^n a^n \quad &\Leftrightarrow \quad b^n[j] = a^n[j+1 \bmod q] \end{aligned} \tag{6.4}$$

Similarly, the i-th coordinate *right-cyclic shift* operator $\triangleright^i$ is defined as:

$$\begin{array}{lll} b^n = \triangleright^i a^n & \Leftrightarrow & b^n[j] = \triangleright^i a^n[j] \qquad 1 \le i < n \\ b^n = \triangleright^n a^n & \Leftrightarrow & b^n[j] = a^n[j-1 \bmod q] \end{array} \tag{6.5}$$

Notice that operating on an n-cube, the left- and right-cyclic shift operation (or $\triangleleft$ and $\triangleright$ for short) does not change the cube's dimension. It merely shuffles the dimension $n-1$ elements if the shift is on the nth coordinate, or recursively applies the shift if the shift is on the lower coordinates $i < n$. □

Example 7. Let us show the effects of cyclic shift operations on the cubes from example 4. For example, $\triangleleft^1 a^1 = \triangleleft^1[011] = [110]$, while $\triangleright^1 b^1 = \triangleright^1[201] = [120]$, and $\triangleright^1\triangleright^1 b^1 = \triangleright^1\triangleright^1[201] = \triangleright^1[120] = [012] = \triangleleft^1[201]$. The last equation shows that $\triangleright\triangleright \equiv \triangleleft$, and conversely $\triangleleft\triangleleft \equiv \triangleright$.

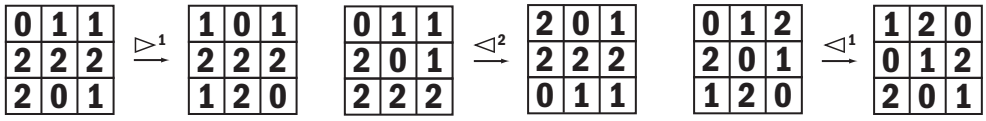


FIGURE 6.4 Three different cyclic shift operations

Referring to the above visualization of the cubes, $\triangleright^1 e^2 = \triangleright^1[a^1 c^1 b^1] = [\triangleright^1 a^1 \triangleright^1 c^1 \triangleright^1 b^1] = [\triangleright^1[011]\triangleright^1[222]\triangleright^1[201]] = [[101][222][120]]$, and $\triangleleft^2 f^2 = \triangleleft^2[a^1 b^1 c^1] = [b^1 c^1 a^1] = [[201][222][011]]$. Finally, $\triangleleft^1 g^2 = [\triangleleft^1[012]\triangleleft^1[201]\triangleleft^1[120]] = [[120][012][201]] = \triangleright^2 g^2$. □

6.1.3 Ternary Latin Hypercubes

In this section, we focus on Latin hypercubes over an alphabet $\mathcal{Q}$ with $q = 3$. These ternary n-cubes are of special interest to us because counting them is equivalent to counting the number of ternary MDS codes of length $n+1$ and minimum distance $d = 2$. First, an observation that along the nth axis, cyclically shifting the stack twice in a given direction is the same as shifting it once in the opposite direction.

Lemma 14. For $q = 3$ and $a^n \in \mathcal{H}^n$, $\triangleright^n \triangleright^n a^n = \triangleleft^n a^n$ and $\triangleleft^n \triangleleft^n a^n = \triangleright^n a^n$.

Proof. From Definition 7, we have the following identities:

$$\begin{aligned} b^n = \triangleleft^n a^n \quad &\Leftrightarrow \quad b^n[\,j\,] = a^n[\,j+1 \bmod q\,] \\ b^n = \triangleright^n a^n \quad &\Leftrightarrow \quad b^n[\,j\,] = a^n[\,j-1 \bmod q\,] \end{aligned}$$

Applying the identities involving $\triangleleft^n$ to produce another n-cube c^n, we get:

$$\begin{aligned} & c^n && = \triangleleft^n b^n = \triangleleft^n \triangleleft^n a^n \\ \Leftrightarrow\ & c^n[\,j\,] && = b^n[\,j+1 \bmod q\,] = \triangleleft^n a^n[\,j+1 \bmod q\,] \\ & && = a^n[\,j+2 \bmod q\,] = a^n[\,j-1 \bmod q\,] \\ \Leftrightarrow\ & c^n && = \triangleright^n a^n \end{aligned} \tag{6.6}$$

and the identities involving $\triangleright^n$, we also get:

$$\begin{aligned} & c^n && = \triangleright^n b^n = \triangleright^n \triangleright^n a^n \\ \Leftrightarrow\ & c^n[\,j\,] && = b^n[\,j-1 \bmod q\,] = \triangleright^n a^n[\,j-1 \bmod q\,] \\ & && = a^n[\,j-2 \bmod q\,] = a^n[\,j+1 \bmod q\,] \\ \Leftrightarrow\ & c^n && = \triangleleft^n a^n \end{aligned} \tag{6.7}$$

The last lines of equations 6.6 and 6.7 conclude the proof that for $q = 3$, the identities $\triangleright^n \triangleright^n a^n = \triangleleft^n a^n$ and $\triangleleft^n \triangleleft^n a^n = \triangleright^n a^n$ hold for all $a^n \in \mathcal{H}^n$. □

Next, we claim that two orthonormal ternary Latin n-cubes can be created from any ternary Latin n-cube using left- and right-cyclic shifts.

Lemma 15. If $a^n,\ b^n \in \mathcal{L}^n$ are two ternary Latin n-cubes. then $a^n \perp b^n$ iff either $b^n = \triangleright^n a^n$ or $b^n = \triangleleft^n a^n$ (or similarly, $a^n = \triangleright^n b^n$ or $a^n = \triangleleft^n b^n$).

Proof. First, let us prove this for $n = 1$. There are only six distinct ternary Latin 1-cubes, the cube $a^1 = [\ 0\ 1\ 2\ ]$ and its permutations $b^1 = [\ 0\ 2\ 1\ ]$, $c^1 = [\ 1\ 0\ 2\ ]$, $d^1 = [\ 1\ 2\ 0\ ]$, $e^1 = [\ 2\ 0\ 1\ ]$, $f^1 = [\ 2\ 1\ 0\ ]$. By inspection, $a^1 \perp d^1 \perp e^1$ and $b^1 \perp c^1 \perp f^1$. Examining these orthonormal Latin 1-cubes, we see that for example, $d^1 = \ \triangleleft^1 a^1$ and $a^1 = \ \triangleleft^1 e^1$, validating the lemma for those pairs. The lemma also holds for other orthonormal pairs $e^1 = \ \triangleleft^1 d^1$, $c^1 = \ \triangleleft^1 b^1$, $f^1 = \ \triangleleft^1 b^1$, and $c^1 = \ \triangleleft^1 f^1$. Obviously, we can reverse the above to obtain the following relationships: $a^1 = \triangleright^1 d^1$, $e^1 = \triangleright^1 a^1$, $d^1 = \triangleright^1 e^1$, $b^1 = \triangleright^1 c^1$, $b^1 = \triangleright^1 f^1$, and $c^1 = \triangleright^1 f^1$. Thus, the lemma holds for $n = 1$. Next, we extend the proof to a general n by induction. First, assume:

$$a^n \perp \ b^n \qquad \Leftrightarrow \qquad b^n = \triangleright^n a^n \quad \text{or} \quad \triangleleft^n a^n \tag{6.8}$$

Now, we have to prove that the lemma is also true for any ternary Latin $n+1$-cubes. For brevity, let us denote the cube c^n by c_0, while $\triangleright^n c^n$ by c_1, and $\triangleleft^n c^n$ by c_2. Similarly, we use the notations d_0, d_1, d_2, e_0, e_1, e_2, f_0, f_1, f_2, g_0, g_1, g_2, h_0, h_1, h_2 to denote the Latin n-cubes $c^n, d^n, e^n, f^n, g^n, h^n$ and their left- and right-cyclic shifts.

First, let $a^{n+1} \perp \ b^{n+1}$, and also assume $a^{n+1} = [\, c^n\ d^n\ e^n\,] = [\, c_0\ d_0\ e_0]$ and $b^{n+1} = [\, f^n\ g^n\ h^n\,] = [\, f_0\ g_0\ h_0]$. From the definition of a ternary Latin hypercube, we know that $c_0 \perp d_0 \perp e_0$. Since $c_0 \perp d_0$ and $c_0, d_0 \in \mathcal{L}^n$, from Equation 6.8 we know that either $d_0 = c_1$ or $d_0 = c_2$. Likewise, $c_0 \perp e_0$ implies that $e_0 = c_1$ or $e_0 = c_2$. Therefore, $a^{n+1} = [\, c_0\ \{c_1, c_2\}\ \{c_1, c_2\}\,]$, with the choices for d_0 and e_0 in braces, resulting in four different choices for a^{n+1}. With$d_0 \perp e_0$, the only two possible choices for a^{n+1} are $[\, c_0\ c_1\ c_2\,]$ or $[\, c_0\ c_2\ c_1\,]$. Similarly, b^{n+1} can only be either $[\, f_0\ f_1\ f_2\,]$ or $[\, f_0\ f_2\ f_1\,]$.

$$a^{n+1} = \{[\, c_0\ c_1\ c_2\,], [\, c_0\ c_2\ c_1\,]\} \perp \{[\, f_0\ f_1\ f_2\,], [\, f_0\ f_2\ f_1\,]\} = b^{n+1} \tag{6.9}$$

Note that the two choices for a^{n+1} have the same first element c_0. Likewise with b^{n+1}, its two choices have the same first element f_0. So let us analyze these first elements.

Since $a^{n+1} \perp b^{n+1}$, from the definition of orthonormality, $c_0 \perp f_0$. But by our assumption in Equation 6.8, this means $f_0 = \{ c_1 \ c_2 \}$. Substituting into equation 6.9, we have 8 choices:

$$\left.\begin{array}{l} a_0 = [\, c_0 \ c_1 \ c_2 \,] \\ a_1 = [\, c_0 \ c_2 \ c_1 \,] \end{array}\right\} = a^{n+1} \perp b^{n+1} = \left\{\begin{array}{l} [\, f_0 = c_1 \ f_1 \ f_2 \,] = [\, c_1 \ c_2 \ c_0 \,] = b_0 \\ [\, f_0 = c_1 \ f_2 \ f_1 \,] = [\, c_1 \ c_0 \ c_2 \,] = b_1 \\ [\, f_0 = c_2 \ f_1 \ f_2 \,] = [\, c_2 \ c_0 \ c_1 \,] = b_2 \\ [\, f_0 = c_2 \ f_2 \ f_1 \,] = [\, c_2 \ c_1 \ c_0 \,] = b_3 \end{array}\right.$$

Notice a_0, the first choice for a^{n+1} and b_1, the second choice for b^{n+1}. Obviously, they are not orthonormal because $[\, c_0 \ c_1 \ c_2 \,] \not\perp [\, c_1 \ c_0 \ c_2 \,]$ because their third elements c_2 are identical. With further observation, we see that not only $a_0 \not\perp b_3$, but also $a_1 \not\perp b_0$, and $a_1 \not\perp b_2$, thus reducing the above equation into the two choices of equations below:

$$\begin{array}{lll} [\, c_0 \ c_1 \ c_2 \,] = & a^{n+1} \perp b^{n+1} & = \{\, [\, c_1 \ c_2 \ c_0 \,], [\, c_2 \ c_0 \ c_1 \,] \} \quad \text{or} \\ [\, c_0 \ c_2 \ c_1 \,] = & a^{n+1} \perp b^{n+1} & = \{\, [\, c_1 \ c_0 \ c_2 \,], [\, c_2 \ c_1 \ c_0 \,] \} \end{array}$$

which can be restated as follows:

$$\begin{array}{llll} a^{n+1} & \perp & b^{n+1} = \{ & \triangleleft^{n+1} a^{n+1} \quad \triangleright^{n+1} a^{n+1} \ \} \quad \text{or} \\ a^{n+1} & \perp & b^{n+1} = \{ & \triangleright^{n+1} a^{n+1} \quad \triangleleft^{n+1} a^{n+1} \ \} \end{array}$$

The above equations are identical and equivalent to saying that if a^{n+1} is orthonormal to b^{n+1}, then b^{n+1} is either $\triangleleft^{n+1} a^{n+1}$, or $\triangleright^{n+1} a^{n+1}$. But this means the lemma also holds for $n+1$-cubes, which completes the induction. The proof for sufficiency follows from the definition of a Latin n-cube. □

The previous lemma leads to an observation that a ternary Latin n-cube always con-

sists of a ternary Latin $n-1$-cube and its left- and right- cyclic shifts along the nth axis, as stated in the following corrolary:

Corrolary 16. Given $a^n, b^n, c^n \in \mathcal{L}^n$ and $a^{n+1} = [\, a^n \;\; b^n \;\; c^n \,] \in \mathcal{L}^{n+1}$, then either $b^n = \triangleright^n a^n$ and $c^n = \triangleleft^n a^n$; or $b^n = \triangleleft^n a^n$ and $c^n = \triangleright^n a^n$. In other words, the elements of a Latin hypercube are cyclic shifts of each other.

Proof. From definition 5, we know that the Latin n-cubes a^n, b^n, and c^n are orthonormal to each other, that is, $a^n \perp b^n \perp c^n$. From Lemma 15, we know that $a^n \perp b^n$ implies $b^n = \triangleleft^n a^n$ or $b^n = \triangleright^n a^n$, and that $a^n \perp c^n$ implies $c^n = \triangleleft^n a^n$ or $c^n = \triangleright^n a^n$.

Substituting the above choices for b^n and c^n into the expression for a^{n+1} gives us four different choices for $[\, a^n \; b^n \; c^n \,]$, which are: $[\, a^n \;\; \triangleleft^n a^n \;\; \triangleleft^n a^n \,]$, $[\, a^n \;\; \triangleleft^n a^n \;\; \triangleright^n a^n \,]$, $[\, a^n \;\; \triangleright^n a^n \;\; \triangleleft^n a^n \,]$, or $[\, a^n \;\; \triangleright^n a^n \;\; \triangleright^n a^n \,]$.

Since a^{n+1} is a Latin n-cube, then $b^n \perp c^n$, and by lemma 2 the only valid choices for a^{n+1} are: $[\, a^n \;\; \triangleleft^n a^n \;\; \triangleright^n a^n \,]$ or $[\, a^n \;\; \triangleright^n a^n \;\; \triangleleft^n a^n \,]$. □

What about cyclic-shifts along an axis other than the nth axis? The next lemma extends the previous result not only to $\triangleleft^n$ and $\triangleright^n$ but also to all $\triangleleft^i$ and $\triangleright^i$ for $1 \leq i < n$. The lemma proves that surprisingly, they are equivalent to either $\triangleleft^n$ and $\triangleright^n$!

Lemma 17. Consider $\mathcal{S}^n$, the space containing the $2n$ left- and right-cyclic shift operators $\triangleright^i$ and $\triangleleft^i$ $(1 \leq i \leq n)$, that can operate on ternary Latin n-cubes $a^n \in \mathcal{L}^n$. The space $\mathcal{S}^n$ can be divided into *two* partitions $\mathcal{S}^n_+$ and $\mathcal{S}^n_-$, represented by $\triangleright$ and $\triangleleft$ (shorthands for $\triangleright^n \in \mathcal{S}^n_+$ and $\triangleleft^n \in \mathcal{S}^n_-$, respectively).

Proof. We will prove this lemma by induction. First, let us prove that this lemma is true for ternary Latin 1-cubes. The proof in this case is trivial because $\triangleright^1 a^1 = \triangleright a^1$ and similarly, $\triangleleft^1 a^1 = \triangleleft a^1$. Here, $\triangleright a^1 = \triangleright^n a^1 = \triangleright^1 a^1$ and $\triangleleft a^1 = \triangleleft^n a^1 = \triangleleft^1 a^1$

Next, assume that it is true for ternary Latin n-cubes, that is, $\triangleright^i a^n = \triangleright a^n$ or $\triangleleft a^n$; and $\triangleleft^i a^n = \triangleright a^n$ or $\triangleleft a^n$. Then it remains for us to prove that the lemma also holds for

$n+1$-Latin cubes to prove the lemma.

$$
\begin{aligned}
\triangleright^i a^{n+1} &= \triangleright^i [\, a0^n \;\; a1^n \;\; a2^n \,] && a1^n = \triangleright^n a0^n \text{ and } a2^n = \triangleleft^n a0^n \\
&= [\, \triangleright^i a0^n \;\; \triangleright^i a1^n \;\; \triangleright^i a2^n \,] && \\
&= [\, \triangleright^n a0^n \;\; a0^n \;\; \triangleleft^n a0^n \,] && \text{or} \quad [\, \triangleleft^n a0^n \;\; a0^n \;\; \triangleright^n a0^n \,] \\
&= \triangleright^{n+1} [\, a0^n \;\; \triangleright^n a0^n \;\; \triangleleft^n a0^n \,] && \text{or} \quad \triangleleft^{n+1} [\, a0^n \;\; \triangleright^n a0^n \;\; \triangleleft^n a0^n \,] \\
&= \triangleright^{n+1} [\, a0^n \;\; a1^n \;\; a2^n \,] && \text{or} \quad \triangleleft^{n+1} [\, a0^n \;\; a1^n \;\; a2^n \,] \\
&= \triangleright^{n+1} a^{n+1} && \text{or} \quad \triangleleft^{n+1} a^{n+1} \\
&= \triangleright a^{n+1} && \text{or} \quad \triangleleft a^{n+1}
\end{aligned}
$$

The first line is from Corollary 16, while the second line is from the definition of $\triangleright^i$ for $i < n$. The third line uses the assumption for ternary n-cubes where $\triangleright^i = \triangleright^n$ or $\triangleleft^n$, giving us 8 different combinations of $\triangleright^n$ and $\triangleleft^n$.

The first element of $\triangleright^i a^{n+1}$ can be $\triangleright^n a0^n$ or $\triangleleft^n a0^n$, then the next element can be $\triangleright^n a1^n = \triangleright^n \triangleright^n a0^n = \triangleleft^n a0^n$, or $\triangleleft^n a1^n = \triangleleft^n \triangleright^n a0^n = a0^n$, and similarly the third element can be $\triangleright^n a2^n = \triangleright^n \triangleleft^n a0^n = a0^n$, or $\triangleleft^n a2^n = \triangleleft^n \triangleleft^n a0^n = \triangleright^n a0^n$.

Still on the third line, the corrolary stipulates that only two of the combinations are valid: $[\, \triangleright^n a0^n \;\; a0^n \;\; \triangleleft^n a0^n \,]$, or $[\, \triangleleft^n a0^n \;\; a0^n \;\; \triangleright^n a0^n \,]$. In the remaining lines we again use the definitions of cyclic-shift operators before proving that indeed $\triangleright^i$ is equivalent to either $\triangleright^{n+1}$ or $\triangleleft^{n+1}$. The proof for $\triangleleft^i$ is identical to the proof we just described for $\triangleright^i$. □

Finally, the theorem proving that there are $6 \cdot 2^{n-1}$ ternary Latin n-cubes.

Theorem 18. There are $|\mathcal{L}^n| = 6 \cdot 2^{n-1}$ ternary Latin n-cubes. Factoring out symbol permutations, there are $6 \cdot 2^{n-1} / 3! = 2^{n-1}$ Latin n-cubes.

Proof. We will prove this by induction. For $n = 1$, 2, and 3, the proof amounts to counting the numbers of permutations, Latin squares, and Latin cubes of order 3. These num-

bers are well known [2] to be $3! = 6$, 12, and 24, respectively. The formula $6 \cdot 2^{n-1}$ obviously agrees with these well-known results.

Suppose the formula also holds for ternary Latin n-cubes, i.e., there are a total of $6 \cdot 2^{n-1}$ of *distinct* $a_i{}^n \in \mathcal{L}^n$, with $i = 1 \ldots 6 \cdot 2^{n-1}$. If we select one of these Latin n-cubes, from the corrolary we know that we can generate exactly two Latin $n+1$-cubes $[\, a^n \lhd a^n \rhd a^n \,]$ and $[\, a^n \rhd a^n \lhd a^n \,]$. Using the same procedure to all the distinct $6 \cdot 2^{n-1}$ Latin n-cubes to generate two distinct Latin $n+1$-cubes, we are creating $6 \cdot 2^n$ distinct Latin $n+1$-cubes. □

6.1.4 Ternary Latin Hypercubes and MDS Codes

Finally, we will establish the link between the ternary Latin hypercubes and the MDS codes. To do this, we will use the properties of the MDS codes.

Definition 8. The q-ary (n, q^k, d) *Maximum Distance Separable (MDS)* code is a set containing q^k distinct codewords (q-ary vectors) of length n, with the Hamming distance between any two codewords of $d = n - k + 1$.

If I is the set of q-ary vectors of length k (the *information* bits) and R is the set of q-ary vectors of length $r = n - k$ (the *redundant* bits) with $|I| = q^k$ and $|R| = q^r$, then we can also define an (n, q^k, d) MDS code as a bijection f from I to R such that for distinct $i_1, i_2 \in I$, the pair $(\, i_1 \; f(i_1)\,)$ and $(\, i_2 \; f(i_2)\,)$ have a Hamming distance of d. □

Theorem 19. For any (n, q^k, d) MDS codes, $d \leq q$.

Proof. Consider $I_0 = \{0^{k-1} q \mid q \in \mathcal{Q}\}$, the set of q-ary vectors of length k, all of which are prefixed with $0 \in \mathcal{Q}$ in the first $k-1$ coordinates. Obviously, $|I_0| = q$. Also consider $R_0 = \{q^r \mid q \in \mathcal{Q}\}$, the set of q-ary vectors of identical symbols $q \in \mathcal{Q}$ of length r.

Next, consider a bijection f from I_0 to R_0. The code represented by f has a Hamming distance of $r+1$. This is still true for all the $(q!)^r$ different ways we can permute the symbols in each of the r coordinates of R_0. Thus, the particular f we chose represents

all variants of f that map q points in I_0 into q points in R_0 such that the minimum pairwise distance of $(\, i \;\; f(i) \,)$ is $r+1$.

From Definition 8, f does not represent an MDS code, because it only maps q vectors of length k, out of the q^k possible. To make f similar to an MDS code, we need to enlarge I_0. Consider a vector i outside I_0. Next, let i be of a distance one from one member $i_0 \in I_0$ and a distance two with the rest.

Since $|\, i - i_0| = 1$, then if $d = r+1$, we must have $|f(i) - f(i_0)| \geq r$, i.e., $f(i)$ must not match any symbol in $f(i_0)$. However, there are only q symbols, thus $r \leq q-1$, or $n-k+1 \leq q$, but, $d = n-k+1$, thus $d \leq q$. □

Example 8. For example, {00000000, 00000001,00000002} are the members of a ternary I_0 with $k = 8$, and {0000, 1111,2222} is a ternary R_0 with $r = 4$. If f is a mapping between I_0 and R_0, then $(\, i \;\; f(i) \,)$ has a distance of 5. The distance is still the same even if we permute the symbols in the four coordinates of R_0. However, as we pointed out in Theorem 19, f does not represent an MDS code. We must enlarge I_0 to cover all 3^8 vectors {00000000, ... ,22222222}, while maintaining the minimum distance d.

Suppose we introduce 00000010 as i. This vector has a distance one with $i_0 = 00000000$, and two with the other vectors in I_0. If we want to have a distance of 5 between i and i_0, we must map i to a vector of length $r = 4$ that is different in *all* four positions from $f(i_0)$ = 0000. We cannot use the symbol 0, which leaves us only with 1 and 2. This means the largest possible r is 2, therefore the maximum d is $r+1 = 3 \leq q = 3$. □

Theorem 20. An $(n+1, q^k, 2)$ MDS code is isomorphic to a Latin n-cube of order q, therefore there are $6 \cdot 2^{n-1}$ distinct $(n+1, 3^n, 2)$ MDS codes.

Proof. From earlier results, we know that for an $(n+1, q^k, 2)$ MDS code, $k = n$ and $r = 1$. The bijection f for this code maps vectors of length n in $\mathcal{Q}^n$ to $\mathcal{Q}$. Consider an isomorphism g that maps the q^n points in the domain of f in $\mathcal{Q}^n$ into the q^n positions of a Latin n-cube a^n. In addition, g also maps the range of f in $\mathcal{Q}$ into the values $a^n[1, 2, \ldots, i_n]$ of

a^n. The bijection relationship f still holds between the cube coordinate positions and the values contained therein. With the result from Theorem 18, we prove that there are $6 \cdot 2^{n-1}$ distinct $(n+1, 3^n, 2)$ MDS codes. □

6.1.5 Experimental Results

To conclude, we present the experimental result that verifies the formula for the first few values of n. We wrote a program to verify that indeed there are $6 \cdot 2^{n-1}$ Latin hypercubes of dimension n.

The program iterates through all $\binom{6 \cdot 2^{n-2}}{3}$ triplets $[\, a_i{}^{n-1} \quad a_j{}^{n-1} \quad a_k{}^{n-1} \,]$ and tests whether the elements meet the orthonormality requirement, i.e., whether $a_i{}^{n-1} \perp a_j{}^{n-1} \perp a_k{}^{n-1}$. If this requirement is met, then the triplet is a Latin n-cube. If not, the program moves on to the next triplet until all triplets are tested. An output file is used to store all the Latin n-cubes found through the iteration.

n	$N = n-1$	$D_3(n,2)_{MDS} = 6 \cdot 2^{n-2} = 3 \cdot 2^N$	$A_3(n,2)_{MDS} = 3^{n-1}$
2	1	6	3
3	2	12	9
4	3	24	27
5	4	48	81
6	5	96	243
7	6	192	729
8	7	384	2187
9	8	768	6561
10	9	1536	19683

The column headers require a little explanation. The symbol n in the first column follows the convention for codeword string length n in coding theory. The Latin hy-

percube dimension is denoted by N instead. The symbol $D_3(n,2)_{MDS}$ denotes the maximum number of distinct ternary MDS codes with length n and minimum distance $d = 2$, while the symbol $A_3(n,2)_{MDS}$ is the maximum number of MDS codewords in a ternary code with length n and minimum distance $d = 2$.

6.2 Decentralized Decision Making in the Game of Tic-Tac-Toe

Traditionally, the game of Tic-Tac-Toe is a pencil and paper game played by two people who take turn to place their pieces on a 3×3 grid with the objective of being the first player to fill a horizontal, vertical, or diagonal row with their pieces. What if instead of having one person playing against another, one person plays against a team of nine players, each of whom is responsible for one cell in the 3×3 grid? In this new way of playing the game, the team has to coordinate its players, who are acting independently based on their limited information. In this section, we present a solution that can be extended to the case where two such teams play against each other, and also to other board games. Essentially, the solution uses a decentralized decision-making, which at first seems to complicate the solution. However, surprisingly, we show that in this mode, an equivalent level of decision-making ability comes from simple components that reduce system complexity.

6.2.1 Introduction

Perhaps it is not an exaggeration to claim that the game of Tic-Tac-Toe is among the most popular childhood games in the world [5, 6]. The game is played by two players who place their different colored or shaped game pieces on a 3×3 grid. Unlike checkers, chess, weiqi (go), and many other board games, the relatively simple grid has enabled people since antiquity to play Tic-Tac-Toe on beach sands, napkins, dusty windshields, or wherever the grid can be drawn. The rule is very simple: players take turn, each time

placing one of their pieces in an unoccupied position on the grid until the grid is filled, or until someone wins. The objective is also simple: the first player to fill a horizontal, vertical, or diagonal row with his/her pieces wins the game.

The fact that Tic-Tac-Toe is so simple and widely known makes it the ideal game of choice for classroom introduction to programming, game theory, data structure, and combinatorial enumeration of all possible outcomes. By one account, there are 765 essentially different configurations of the game pieces, or 26,830 possible games, taking into account different symmetries. If symmetry is not considered, there are 255,168 possible games.

While 255,168 sounds like a large number for a human player to memorize, it is certainly not a large number for most modern computers. One can imagine "training" a computer to be a competent player in Tic-Tac-Toe by memorizing all 255,168 games and use this knowledge to calculate the best move based on the existing board configuration.

This approach is of course a far cry from how people play. No one memorizes all 255,168 games to calculate the best move. Instead, we humans use simple rules that provide several possible moves, from which the best move was chosen. Sooner or later, most human players discover that Tic-Tac-Toe always ends in a draw when both players use the optimal rule of the game.

Just as we humans develop our decision-making ability through playing games — starting from the simple and concrete games to the more complex and abstract games — in its evolution, computers spent some of their "childhood" years playing Tic-Tac-Toe (ironically, after being forced into the horror of calculating ballistic trajectories, code breaking, and simulating atomic explosions in their "infancy" years!)

One of the first computers in the 1950s to play Tic-Tac-Toe, EDSAC1 was capable of playing a perfect game with a program less than 4,000 bytes long [7]. A human played against a single player, the machine. This tradition continued well into the modern era of the Internet. A casual search on the Internet would return hundreds, if not thousands,

of interactive web pages capable of playing perfect games against human players.

Taking a peek into the source codes behind these games and stripping away the user interface codes — leaving only the equations, logic, and database used by the programs — one cannot avoid the impression of relative complexity for such a simple game. Can we do better? Here we address the interesting question: is there a simpler way to program a competent Tic-Tac-Toe player? Can complexity be further reduced by a different, i.e., decentralized mode of decision-making? Later, we elaborate the definition of a competent player. For now, we simply mean a player who makes no mistake and can consistently force a draw, regardless of which player starts the game.

What if instead of trying to create one monolithic competent player, we create nine agents and one manager, together acting as a single competent player? At first, it sounds like we have increased the complexity of our solution. After all, the team still has to respect the original rules of the game, meet the same priorities, and on top of that, coordinate its action. However, here we show that surprisingly, the end result is a simpler set of rules for each agent. Consequently, the team has a much lower complexity compared to an equivalent centralized implementation, as evidenced by the types and numbers of instructions used by the team.

Further, it is not hard to imagine that in certain computing platforms, decentralized decision-making is the only possible avenue of computation. Next, we begin the presentation by analyzing a competent Tic-Tac-Toe player.

6.2.2 Competent Player

A competent player is defined as a player who has in its arsenal a complete collection of strategies that are necessary to consistently force a draw when faced with another competent player or win when the other player makes a mistake. In contrast, a less-than-competent player only has a subset of these strategies. For convention, the grid is numbered as in Figure 6.5 below. In this section, we assume that the opponent plays the O (for “opponent”) while the Tic-Tac-Toe team plays the X.

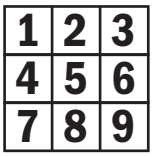


FIGURE 6.5 Grid cell numbering convention

A. Defensive Strategic Categories

A novice defensive player can block an immediate threat, i.e., it can prevent an opponent from completing a row. Such a player detects the presence of two opponent pieces on a row and reacts by placing its own piece in the remaining space on the targeted row.

An intermediate defensive player can detect and preempt any attempt by the opponent to introduce two possible completions. For example, in Figure 6.6, suppose the opponent, player O, just moved. An intermediate defensive player knows how to prevent the opponent from occupying cell 8, which prevents a two-completion attack on cell 5 and 9 in the next move.

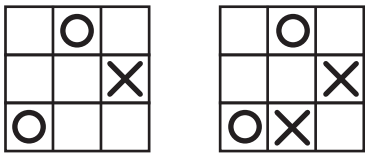


FIGURE 6.6 Intermediate defensive player

However, note that the opponent can also launch a two-completion attack on cell 3 and 4 by occupying cell 1. To force a draw, the player needs to utilize more than defensive moves. As the famous dictum says, "The best defense is offense." If we occupy cell 5, the opponent is forced to follow a series of defensive moves that lead to a draw.

An advanced defensive player can react appropriately to an opening move. In Tic-Tac-Toe, this translates to placing a piece in the center cell if the opponent starts anywhere but the center. If the opponent starts from the center cell, such a player reacts by occupying one of the corner cells. Similarly, we can categorize players based on their offensive capabilities.

B. Offensive Strategic Categories

A novice offensive player can complete a row when two of its own pieces are already placed in a row with one remaining empty position. This is, as the name suggests, the most basic offensive capability a player must have to have a chance of effectively winning against another player.

An intermediate offensive player can threaten the opponent with a one-completion attack, thus forcing the opponent to react and defend the position, possibly disrupting any planned move. For example, in Figure 6.7, suppose the opponent just placed an O in cell 6. An intermediate offensive player would threaten the opponent by placing his piece in cell 1 (or 2), forcing the opponent to occupy cell 2 (or 1).

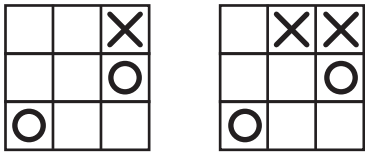


FIGURE 6.7 Intermediate offensive player

An experienced offensive player can threaten the opponent with two possible completions on two rows, thus guaranteeing a win. In Figure 6.8, the opponent just placed an O in cell 6. An experienced offensive player can force a win by placing an X in cell 2 — a two-completion attack in cell 1 and 5.

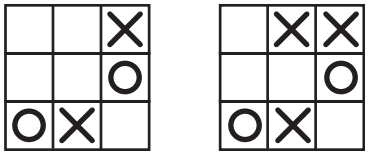


FIGURE 6.8 Experienced offensive player

Finally, an advanced offensive player can make the most aggressive opening move. Playing against an opponent executing random moves, this means occupying any one of the corner cells, giving a 7 out of 8 chance of winning. Playing against a competent opponent, the most strategic move is to occupy the center cell, denying four possible completions.

6.2.3 Team Infrastructure

Having described the different player categories, of course we eventually want to create a team of 9 agents (plus one manager) that can emulate, through their independent actions, the level of competence shown by an advanced defensive and offensive player. However, first let us describe the infrastructure available to the team members.

First, let us describe the role of the manager as shown in the algorithm below. It performs a primitive coordinating role for the agents and nothing more. In fact, the same manager can be used for Tic-Tac-Toe or other turn-based (could be multi-player) board games because the manager knows almost nothing about the game its agents are playing.

```
MANAGER
1: Wait until a new opponent piece is placed.
2: Once placed, ask all active agents to start calculation.
3: Wait for the agents to submit their responses.
4: Choose one of the responses with the highest priority.
5: Notify all agents which agent is selected.
6: If all cells are filled, then end. Otherwise, go back to 1.
```

Each agent responds back with a priority number — saying "let me handle this." In step 4, the manager selects the agent with the highest priority. If there is a tie, it selects one response (either at random or first arrival, etc.)

These priority numbers convey the subjective and private belief of each agent of how important it thinks the information it has, and the reaction it plans to do, to the success of the team (in this case, in playing the Tic-Tac-Toe game, although this could be easily extended to other applications!) The manager thus resolves any possibly conflicting subjective views by impartially (or partially, in a consistent way) selecting one of the agents. Let us now describe the role of the agents, as shown in the algorithm below:

```
AGENT
1: Wait for the instruction to start from the manager.
2: If the opponent already landed in this cell, or
   reaction is already made, then end. Otherwise, move to 3.
3: Obtain all accessible information about the board.
4: Consult the function T for a  priority number.
5: Once found, submit the priority number as a response.
6: Wait for the selection notification from the manager.
7: If selected, then react.  Otherwise, go to 1.
```

Before explaining the algorithm, let us distinguish the word "response" and "reaction". An agent responds to the manager by providing a priority number. In contrast, an agent reacts to the opponent by placing a friendly piece where the agent is assigned to operate.

Step 1 is trivial. It simply asks the agent to wait for the instruction from the manager before it begins calculating because reliable calculation has to be done based on the most current and relevant state of information available. The manager has a global knowledge of when the opponent introduces a new piece into the board. Therefore, step 1 provides the synchronizing signal for information processing that precedes decision-making.

Step 2 is also easy to understand. An agent no longer has to compute a reaction if it has already made one, or if the opponent has eliminated any reason for the agent to make one (by placing its piece where the agent is located.)

Step 3 is very crucial to an agent's operation. In one extreme case, an agent calculates the response in absence of any factual information of the board configuration, i.e., it is simply a "fortune-teller" — providing suggestion to the manager based on internal and unsubstantiated private beliefs. In another extreme, an agent has complete information on the board configuration. Clearly, these extreme cases are not desirable (or practical).

What makes our model interesting is the case where the agents have incomplete in-

formation (by design) about the board, from which they infer the best move in their own areas of responsibility. The agents then convey this inference to a manager, who then arbitrates conflicting priorities.

In what will become clear in the examples provided in this section, the agents do not have to access the same rule, level, scope, and type of information. In many simple board games, the agents can be identically programmed. However, in more complex games, the agents can easily operate within an informational and operational (rule) hierarchy.

Step 4 essentially declares the existence of a "rule book" for each agent. Of course, in some games, the agents can also be permitted to "improvise," i.e., suggesting certain actions and priorities based on their own internal probabilistic process unknown to the manager. Confronted with a board scenario (which is nothing more than the information about the board available to the agent), the agent attempts to judge, "how important is my reaction going to be compared to those of other agents?" The answer is scored by its priority number and then submitted to the manager in step 5.

Finally, in steps 6 and 7, the agent simply waits for the manager's response. If the manager decides to activate the agent, then the agent reacts and fulfills its mission. If not, it waits for another round of decision-making.

6.2.4 The Tic-Tac-Toe Team

The team infrastructure described in the previous subsection allows us to start our construction of a competent Tic-Tac-Toe player by first building a novice defensive team that perfectly emulates a novice defensive player.

Let us assume that there are three types of agents, each with their own programs and level of information access. Therefore, there are three types of functions T used in step 4.

In case of Tic-Tac-Toe, the information access rule is such that "an agent has perfect information on the states of all cells located in the same horizontal and vertical (and

whenever appropriate, diagonal) row as the cell it is in." The state could be empty, occupied by an opponent piece, or by a friendly piece. Each agent knows the cell it occupies. In Figure 6.9, the agent is marked by an X, and the cells to which it has perfect information is marked by the bullet symbols.

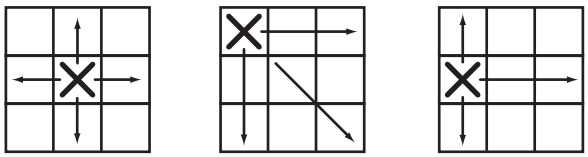


FIGURE 6.9 Information access rule

The agent located in the center cell of the grid must have what amounts to a perfect information on all the cells (see the left box). The agents in the corner cells must have access to the horizontal, vertical, and diagonal rows (6 cells — see the middle box), and the agents along the edges only have to know the horizontal and vertical rows (4 cells).

```
if n(oo) = 1 then return 1
return 2
```

FIGURE 6.10 Novice defensive strategy

We claim that for a novice defensive team, the function T can be as simple as the one shown in Figure 6.10. The notation n(oo) means the number of horizontal, vertical, and diagonal rows containing two opponent pieces. In this notation, a friendly piece is denoted by an x, and a blank cell is by a b. For example, n(bx) means the number of rows containing a blank and a friendly piece. Evaluated at the center, corner, and edge cells, the function $n(\cdot)$ can return up to four, three, and two rows, respectively.

Suppose the board configuration is shown in Figure 6.11a. In this game, the opponent pieces are marked by the O's. Figure 6.11b shows the priority numbers calculated by all the nine agents as they are submitted to the manager. The team will correctly block the attack by occupying cell 1.

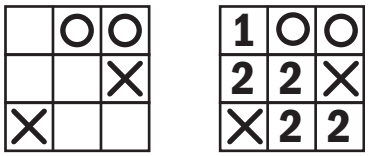


FIGURE 6.11 Novice defensive play

Of course, the opponent O could be smarter and present a two-completion attack as shown in Figure 8 below. In this scenario, all agents in a novice defensive team report a non-severe priority number of 2. The manager thus randomly (or systematically) selects one of the 5 possible agents, and only with a probability of 1/5, the manager selects the agent in cell 3, which directly neutralizes the two-completion attack.

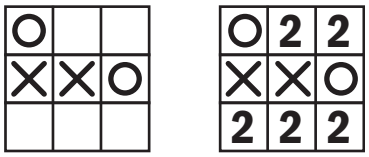


FIGURE 6.12 Failure of the novice defensive strategy

How do we prevent this uncertainty? One solution is to make team smarter — and since the manager is dumb, this means making the agents smarter. Instead of the earlier T, the agents can use a more robust T:

```
if n(oo) = 1 then return 1
if n(ob) = 2 then return 2
return 3
```

FIGURE 6.13 Intermediate defensive strategy

The function gains another line, but the agents can now collectively defend against two-completion attacks from the opponent! Now, if the agents are confronted with a scenario shown in Figure 6.14 below, they independently evaluate priority numbers shown in the right subfigure, and the manager correctly chooses the best agent to fend off the attack.

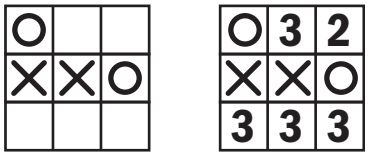


FIGURE 6.14 Intermediate defensive play

In Figure 6.12, the agents in cells 2 and 8 can also thwart the two-completion attack by launching a high-priority attack on the opponent, rather than neutralizing the opponent's lower-priority two-completion attack. But this requires more than defensive strategies, but also some offensive capabilities. By process of induction one infers that T needs to be expanded further and this is indeed correct. To emulate a novice offensive player, the agent now also has to have access to information on friendly pieces on the board.

However, now the question is which one should have a higher priority: the novice offensive strategy (which basically ends the game with a win) or the novice defensive strategy (which averts a loss by another move)?

Obviously, any offensive move that can end the game with a win should take a higher priority than other defensive moves. This principle is reflected in the version of T implementing offensive capability shown in Figure 6.15.

```
if n(xx) = 1 then return 1
if n(oo) = 1 then return 2
if n(ob) = 2 then return 3
return 4
```

FIGURE 6.15 Novice offensive strategy

If an agent is still making this calculation, then it must be located in a blank cell and has not made a reaction. Armed with an offensive strategy, the agent can win the game for the team by making a reaction. For example, suppose the X team is confronted with the board configuration shown in Figure 6.16. Using T, the agent in cell 9 reacts and secures the win.

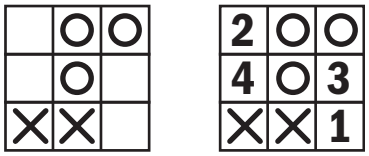


FIGURE 6.16 Novice offensive play

We can extend the function T further to implement the intermediate and experienced offensive strategies as shown in Figure 13 below. The symbol ee represents both bb and ox.

```
if n(xx) = 1 then return 1
if n(oo) = 1 then return 2
if n(bx) = 2 then return 3
if n(bx) = 1 then return 4
if n(bo) = 2 then return 5
if n(ee) = 2 then return 7
if n(ee) = 1 then return 6
```

FIGURE 6.17 Intermediate and experienced offensive strategies

An example of the board configuration that showcases the application of these new strategies is shown in Figure 6.18. In this configuration, the team is confronted with the choice of blocking a two-completion attack by occupying cell 3, or completing its own threat of two-completion attack by occupying cell 7. Using the previous function, the team can correctly adopt into an aggressive stance and make an offensive move that secures the win.

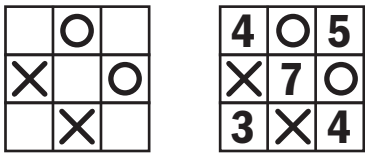


FIGURE 6.18 Intermediate and experienced offensive play

At this point, we can claim that we almost have a team of agents (plus a manager) that emulate the ability of a competent player. The function T for each agent is very simple and intuitive. Although coordinated centrally, decision is made in a decentralized manner with locally available information.

We have not discussed the all-important opening move. In Tic-Tac-Toe, this move determines the course of the game. If the team starts first, how do we customize the function T such that it starts from the center? If the opponent moves first, how do we program T so the team responds at the center if the opponent starts from the corner and vice versa?

The answer of course lies in the function T. Nowhere in this section do we require the agents to be identically programmed, i.e., they can have different function T! So suppose we use the following functions T1, T2, and T3 for the center, corner, and edge agents, respectively:

At this point, we have reached our original objective of constructing a competent Tic-Tac-Toe team that is practically indistinguishable from a competent player. In the next subsection, we discuss several theoretical issues and the issue of extending this framework to other types of board games.

```
T1: return 1
T2: if n(xx) = 1 or center != x then return 1
    if n(oo) = 1 then return 2
    if n(bo) = 2 and n(bx) = 1 and center != x
       then return 3
    if n(bo) = 2 and n(bx) = 1 then return 5
    if n(bx) = 2 then return 3
    if n(bx) = 1 then return 4
    if n(bo) = 2 then return 5
    if n(ee) = 2 then return 7
    if n(ee) = 1 then return 6
T3: if n(xx) = 1 then return 1
    if n(oo) = 1 then return 2
    if n(bx) = 2 then return 3
    if n(bx) = 1 then return 4
    if n(bo) = 2 then return 5
    if n(ee) = 2 then return 7
    if n(ee) = 1 then return 6
```

FIGURE 6.19 Full implementation of the strategy

6.2.5 Discussion

Many interesting issues arise from this new framework. For example, is it even possible for the agents to initiate a two-completion attack on the opponent given that they only have access to the information from cells on the same row? The answer is, yes, it is possible. However, it cannot be done without using indirect inference and reducing the aggressiveness of the agents. In Figure 6.20, we illustrate why this is so. Given the board situation shown on the left, the agents use the T in Figure 6.19, resulting in Figure 6.20b.

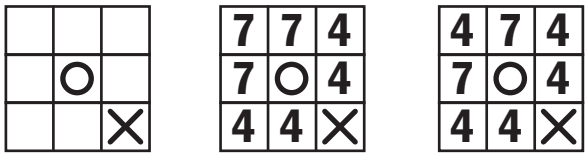


FIGURE 6.20 Initiating a two-completion attack

Using the program in Figure 6.19, the agents can launch their attacks immediately after the game starts. The corner agents can be programmed to initiate two-completion attacks if the priority number for the event n(bb)=2 is mapped to a 4, as shown in Figure 6.20c.

Now, instead of being limited to responding with immediate attacks, the agents can launch a delayed, although more potent, coordinated attack. Thus, we can say that with this change, the overall aggressiveness of the team is reduced (although we can argue that the team's finesse is increased).

Another issue is whether the team manager can ask only a subset of the active agents that are immediately affected by the opponent's move. For example, if the opponent starts from a corner, the manager could ask the agents on the two edges and one diagonal intersecting the "invaded" corner.

The advantage of this method is a reduced level of agent activity. Because the Tic-Tac-Toe grid is small, the saving is quite small. If we extend this method to a game with a much larger board (for example, checkers), the saving can be substantial. Further, this method allows for a second level of decentralization (or a hierarchy) by introducing local managers into the game. These managers then have their own areas of responsibilities and agents.

The disadvantage does not become obvious until this decentralized team faces either a monolithic player with a superior computation capacity, or another superior team opponent not constrained by limits on agent activity and communication. Such strong opponents can devise maneuvers that provoke agents from different managerial areas of responsibility to react in separate ways that might be locally optimal with respect to the

limited knowledge and coordination they have available, but nevertheless ineffective to neutralize the lethality posed by the global threat posed by the opponent.

Finally, there are possible extensions of this approach to games other than Tic-Tac-Toe. Games similar to Minesweeper (which has been proven to be NP-complete) can benefit from a decentralized approach — in fact, in real life minesweeping operations, this approach is the ONLY way!

6.2.6 Conclusion

In this section, we have presented a decentralized method of playing Tic-Tac-Toe. The method is extensible to other turn-based board games, especially those games where the pieces do not move once placed on the board. Future research may include extending this method to other games with moving pieces such as checkers, fox and geese, etc.

BIBLIOGRAPHY

[1] G.L. Mullen, R.E. Weber, "Latin cubes of order ≤ 5," *Discrete Mathematics*, vol. 32, no. 3, pp. 291–297, 1980.

[2] N.J.A. Sloane, "A Handbook of Integer Sequences," Academic Press, New York, 1973.

[3] F.J. MacWilliams and N.J.A. Sloane, "The Theory of Error-Correcting Codes," Elsevier / North-Holland, New York, 1977.

[4] R.C. Singleton, "Maximum Distance q-ary Codes," *IEEE Transactions on Information Theory*, vol. 10, pp. 116–118, 1964.

[5] M. Gardner, "Ticktacktoe Games." *Wheels, Life, and Other Mathematical Amusements*. W.H. Freeman, pp. 94–105, 1983.

[6] Wikipedia contributors, "Tic-Tac-Toe," *Wikipedia, The Free Encyclopedia,* http://en.wikipedia.org, accessed 19 December 2005.

[7] M.R. Williams, "Cambridge celebrates 50 years since EDSAC," *IEEE Annals of the History of Computing*, vol. 21, no. 3, pp. 72–72, 1999.