Path Problems in Networks

Synthesis Lectures on Communication Networks

Path Problems in Networks

John S. Baras and George Theodorakopoulos

www.morganclaypool.com

ISBN: 9781598299236 paperback
ISBN: 9781598299243 ebook

DOI 10.2200/S00245ED1V01Y201001CNT003

A Publication in the Morgan & Claypool Publishers series
SYNTHESIS LECTURES ON COMMUNICATION NETWORKS

Lecture #3
Series Editor: Jean Walrand, *University of California, Berkeley*
Series ISSN
Synthesis Lectures on Communication Networks
Print 1935-4185 Electronic 1935-4193

Path Problems in Networks

John S. Baras
University of Maryland

George Theodorakopoulos
École Polytechnique Fédérale de Lausanne

SYNTHESIS LECTURES ON COMMUNICATION NETWORKS #3

MORGAN & CLAYPOOL PUBLISHERS

ABSTRACT

The algebraic path problem is a generalization of the shortest path problem in graphs. Various instances of this abstract problem have appeared in the literature, and similar solutions have been independently discovered and rediscovered. The repeated appearance of a problem is evidence of its relevance. This book aims to help current and future researchers add this powerful tool to their arsenal, so that they can easily identify and use it in their own work.

Path problems in networks can be conceptually divided into two parts: A distillation of the extensive theory behind the algebraic path problem, and an exposition of a broad range of applications. First of all, the shortest path problem is presented so as to fix terminology and concepts: existence and uniqueness of solutions, robustness to parameter changes, and centralized and distributed computation algorithms. Then, these concepts are generalized to the algebraic context of semirings. Methods for creating new semirings, useful for modeling new problems, are provided. A large part of the book is then devoted to numerous applications of the algebraic path problem, ranging from mobile network routing to BGP routing to social networks. These applications show what kind of problems can be modeled as algebraic path problems; they also serve as examples on how to go about modeling new problems.

This monograph will be useful to network researchers, engineers, and graduate students. It can be used either as an introduction to the topic, or as a quick reference to the theoretical facts, algorithms, and application examples. The theoretical background assumed for the reader is that of a graduate or advanced undergraduate student in computer science or engineering. Some familiarity with algebra and algorithms is helpful, but not necessary. Algebra, in particular, is used as a convenient and concise language to describe problems that are essentially combinatorial.

KEYWORDS

algebraic path problem, algebraic modeling, semiring, dioid, graph, shortest path problem, communication network, routing, distributed algorithms, Dijkstra, Bellman-Ford, Floyd-Warshall

Contents

Preface

Numerous network problems can be viewed as instances of the same abstract "algebraic path problem," the most classical example being shortest path routing. In all these problems, the underlying model is a graph, and the objects of central importance are the paths between two given vertices of that graph. The edges of each path are aggregated *along* the path, and then the results are aggregated *across* all paths. In the shortest path problem, we aggregate along a path by *adding* its edge weights, and we aggregate across paths by taking the *minimum* path weight.

The differences among the various instances of the algebraic path problem lie in the meaning assigned to edge weights (delays in the case of shortest path routing), and in the operations used to aggregate these weights. For example, suppose that the weight of an edge is the probability of successfully transmitting a data packet along that edge. We can multiply edge weights along each path, and then take the maximum path weight across all paths from a given source to a given destination. Then, we have the maximum reliability path problem. As long as the set of weights and the operators satisfy certain algebraic properties (those of a *semiring*), the same facts are true and the same algorithms can be used to find the solution.

Knowing these common and reusable facts and algorithms adds a valuable modeling and evaluation tool to the network researcher's arsenal. The main motivation for writing this text was the observation that these facts were independently discovered in various areas. Therefore, it behooves one to become familiar with their common structure, so as to be able to identify it when it appears. A particularly useful feature, which seems not to have received due attention in the literature, is that algorithms described as an algebraic path problem lend themselves to distributed implementation, similarly to the distributed version of the Bellman-Ford algorithm for shortest paths.

This monograph is targeted at network researchers, engineers, and graduate students. The theoretical background assumed for the reader is that of a graduate or advanced undergraduate student in computer science or engineering. Some familiarity with algebra and algorithms is helpful, but not necessary. Algebra, in particular, should be seen as a convenient and concise language to describe problems that are essentially combinatorial. The presented applications are mostly drawn from the networking literature, so network researchers can use them as examples of how to apply the theory to their own specific areas. In fact, one could start with the applications as a quick introduction to the topic.

The content of this monograph is divided into theory and applications. In Chapter 1, we refresh the reader's memory on the classical shortest path problem, and touch upon the concepts (solutions and algorithms) that are generalized in the following chapters. In Chapter 2, we define semirings, describe ways for creating new ones, and present the main problem to be solved, namely, resolution of $x = Ax + b$ in semirings. We present properties of the solutions (existence, uniqueness, sensitivity to parameter changes) in Chapter 3, where we also generalize the algorithms for computing solutions.

In Chapter 4, we discuss numerous applications and show how they fit in the algebraic path framework. Chapter 5 gives examples from two areas that are very close to semiring path problems. First, non-semiring path problems, i.e., problems that can be formulated as path problems, but their natural formulation does not lead to a semiring. We show how one can still try to overcome some of the obstacles so as to satisfy the properties of a semiring. The second area is that of semiring non-path problems, i.e., different applications of semirings, but still in the areas of combinatorics and networking. A summary chapter (Chapter 6) at the end lists all the semirings and applications presented in Chapter 4.

In general, we have refrained from giving detailed proofs. Already existing books have approached the algebraic path problem from a more mathematical viewpoint [31], [14], [22], [21] and most recently [23]. Our choice in this monograph has been to give the main theoretical facts, and then present a broad range of applications so as to demonstrate the practical relevance of the algebraic path problem.

John S. Baras and George Theodorakopoulos
January 2010

CHAPTER 1

Classical Shortest Path

1.1 DEFINITIONS AND PROBLEM DESCRIPTION

The shortest path problem is arguably the theoretical problem that is most relevant to communication networks. Distance vector routing in the Internet uses the distributed version of Bellman-Ford, whereas link state routing uses Dijkstra's algorithm.

We will use the shortest path problem as a familiar example to touch upon some themes, which we will then revisit in the more general setting of semirings. These themes are existence, uniqueness, and sensitivity of the solution, as well as centralized and distributed computation.

We give the necessary graph theoretical definitions in the following subsection. Many textbooks exist on this topic, e.g., by Béla Bollobás [10]. Then, we formally describe the shortest path problem and conditions for the existence and uniqueness of a solution. More on the theory and practice of the shortest path problem can be found in textbooks on algorithms (e.g., [15]) or on networking (e.g., [8]).

1.1.1 GRAPH THEORY

A graph G is an ordered pair of sets (V, E). The former, V, is the set of *vertices*, and the latter, E, is the set of edges. The set E is a subset of the set of unordered pairs of V. An edge $\{u, v\}$, $u, v \in V$ will be denoted uv. We will use the variable n for the number of vertices, and m for the number of edges, both of which will be assumed finite. A graph $G' = (V', E')$ is a *subgraph* of G if $V' \subseteq V$ and $E' \subseteq E$. If $V' = V$, then G' is a *spanning* subgraph of G.

Two vertices $u, v \in V$ are called *neighbors* of each other if $uv \in E$. The set of neighbors of a vertex u is denoted by $N(u)$, and the number of its neighbors is called the *degree* of u.

A *path* is an ordered sequence of vertices, such that an edge exists between two successive vertices: $(v_0, v_1, \ldots, v_k)$ is a path if and only if $v_i v_{i+1} \in E, \forall i = 0, \ldots, k-1$. The vertex v_0 is the *initial* vertex of the path, and v_k the *terminal* vertex. The set of paths whose initial vertex is u and terminal vertex is v is denoted by P^{uv}. A path with no repetitions of vertices is an *elementary* path. A *cycle* is a path in which the initial and terminal vertices coincide. A set of paths is *independent* if their common vertices are exactly their initial and terminal vertices. Such paths are also called *internally disjoint*.

If there is a path from u to v, then v is *reachable* from u. If every vertex is reachable from every other vertex, the graph is *connected*. A connected graph without cycles is a *tree*. A maximal connected subgraph is called a *connected component*. An edge is a *bridge* if its removal increases the number of connected components of the graph. Similarly, a vertex is a *cut vertex* if its removal increases the number of connected components of the graph.

An edge (resp. vertex) weighted graph (G, w) is a graph G together with a weight function w defined on the edges (resp. vertices).

A directed graph is a graph whose edges are ordered pairs, as opposed to unordered. Therefore, the edges (u, v) and (v, u), where $u, v \in V$, are distinct, and existence of one does not imply existence of the other. The notion of neighbors is now generalized to *in-neighbors*: the set $\{v \in V | (v, u) \in E\}$ and *out-neighbors*: the set $\{v \in V | (u, v) \in E\}$. Paths are directed paths, and reachability is no longer symmetric. Connectivity is replaced by strong and weak connectivity: A directed graph is *strongly connected* if every vertex is reachable from every other vertex. A directed graph is *weakly connected* if for every pair of vertices u, v, either u is reachable from v, or v is reachable from u.

One usual way of representing a graph is by its *adjacency matrix* A, of dimension $n \times n$, defined as

$$A_{ij} = \begin{cases} 1, & \text{if } ij \in E \\ 0, & \text{if } ij \notin E. \end{cases} \tag{1.1}$$

If the graph is edge-weighted, then A_{ij} is equal to the weight of edge ij. It is easy to see that the adjacency matrix of an undirected graph is symmetric, which is in general false for a directed graph. Since the correspondence between graphs and their adjacency matrices is one-to-one, it is possible, given an arbitrary square matrix A, to define an associated graph $G(A)$.

1.1.2 THE SHORTEST PATH PROBLEM

We are given a directed graph $G = (V, E)$ with a real-valued edge weight function $w : E \longrightarrow \mathbb{R} \cup \{\infty\}$. We will assume that all possible edges in $V \times V$ exist, but some may have a weight of ∞.

We define the weight of a path $p = (v_0, v_1, \ldots, v_k)$ to be the sum of the weights of its edges:

$$w(p) = \sum_{i=0}^{k-1} w(v_i, v_{i+1}). \tag{1.2}$$

The shortest path weight from vertex u to vertex v is the smallest path weight among all paths that start at u and end at v:

$$\min_{p \in P^{uv}} w(p). \tag{1.3}$$

If there is no path from u to v, then the shortest path weight is defined to be infinite. Any path whose weight is equal to the shortest path weight, is a shortest path from u to v.

Existence If a negative weight cycle exists in G and is reachable from s, then there is no shortest path to any destination vertex t that is reachable from any vertex of the cycle: For any finite number $M \in \mathbb{R}$, there are $s - t$ paths with weight less than M. In this case, we define the shortest path weight for the appropriate destinations to be $-\infty$.

Uniqueness If there are no negative weight cycles, then the shortest path weight is unique although the shortest path might not be. If there are zero-weight cycles, there will be an infinity of shortest paths traversing the zero-weight cycle an arbitrary number of times. If all the cycles have positive weight, then the shortest paths will be elementary.

The Single Source Shortest Path problem (SSSP) amounts to finding the shortest paths from a given vertex s to all other vertices of G.

The All-Pairs Shortest Path problem (APSP) amounts to finding the shortest paths from every vertex $s \in V$ to every other vertex of G.

There are two remarks: The algorithms that follow compute the shortest path weight. It is easy to augment them to compute the corresponding path, too. Note also that in network routing, an alternative version is used, that of the single destination shortest path. But this is equivalent to the SSSP; all that is needed is to reverse the direction of all the edges.

1.2 COMPUTATION OF SHORTEST PATHS

All three algorithms that we present next maintain a vector $d[v]$, $v \in V$ (or a matrix $D(u, v)$, $u, v \in V$ in the case of Floyd-Warshall) with estimates of the shortest path weight to each node v. Then they iterate to produce estimates that converge to the correct shortest path weights. Their difference is that each algorithm iterates on a different increasing set of paths.

1.2.1 DIJKSTRA

Dijkstra's algorithm [16] solves the SSSP when all edges have non-negative weights. It iterates over progressively longer path weights.

```
Dijkstra($G, w, s$)
1  $d[s] \leftarrow 0$
2  for $v \in V \setminus \{s\}$
3        do $d[v] \leftarrow \infty$
4  $Q \leftarrow V$
5  while $Q \neq \emptyset$
6        do $u \leftarrow v$ s.t. $d[v] = \min\{d[x] | x \in Q\}$
7           $Q \leftarrow Q \setminus \{u\}$
8           for each vertex $v \in Q$
9                do $d[v] \leftarrow \min\{d[v], d[u] + w(u, v)\}$
```

The invariant of Dijkstra's algorithm is that, in line 5, the set $V \setminus Q$ contains vertices whose shortest path weights have been determined. At the beginning of each while loop, the vertex v with the lowest shortest path estimate $d[v]$ is removed from Q. It can be proven that at the time of this removal, $d[v]$ is equal to the weight of the shortest $s - v$ path. Therefore, when Q is empty, all shortest path weights have been determined.

The time complexity of Dijkstra's algorithm is $O(n^2)$. If the graph is sparse, i.e., many edge weights are ∞, we can improve it to $O(n \log n + m)$ by relaxing only the finite edge weights and maintaining the estimates $d[v]$, $v \in V$ as a Fibonacci heap data structure[1].

1.2.2 BELLMAN-FORD

The Bellman-Ford algorithm (based on Bellman [6] and Ford [33]) solves the SSSP even when there are negative weight edges, but it is slower than Dijkstra. It can be also used to detect negative cycles.

It iterates over the number of edges in the paths. At each execution of the loop (lines 4 - 6), paths that have exactly k edges are taken into account. So, at the end of the loop for a given value of k, the estimate $d[v]$ is equal to the shortest $s - v$ path weight among all paths, not necessarily elementary, with at most k edges.

There are two slightly different versions of the Bellman-Ford algorithm, which are equivalent only if the initial distance estimates to all non-source nodes are set to values higher than the true ones.

```
Bellman-Ford-v1($G, w, s$)
1  $d[s] \leftarrow 0$
2  for $v \in V \setminus \{s\}$
3      do $d[v] \leftarrow \infty$
4  for $k \leftarrow 1$ to $n - 1$
5      do for each edge $(u, v) \in E$
6             do $d[v] \leftarrow \min\{d[v], d[u] + w(u, v)\}$
```

This version of Bellman-Ford takes $O(nm)$ time, which is equal to $O(n^3)$ if the graph is dense. From line 6, we can see that the value of each $d[v]$ can only decrease at each iteration, so the initial conditions need to be higher than the true shortest path weights. Otherwise, the algorithm will converge, if at all, to a solution that is irrelevant for the shortest path problem.

```
Bellman-Ford-v2($G, w, s$)
1  $d^0[s] \leftarrow 0$
2  for $v \in V \setminus \{s\}$
3      do $d^0[v] \leftarrow \infty$
4  for $k \leftarrow 1$ to $n - 1$
5      do for each node $v \in V \setminus \{s\}$
6             do $d^k[v] \leftarrow \min_{u \in V}\{d^{k-1}[u] + w(u, v)\}$
```

We will use the term *relaxation* for the operation of line 6:

$$d^k[v] \leftarrow \min_{u \in V}\{d^{k-1}[u] + w(u, v)\}. \tag{1.4}$$

[1] For more on data structures and running time of algorithms, we refer the reader to textbooks on algorithms [15].

This version of Bellman-Ford takes $O(n^3)$ time, which can be shortened to $O(nm)$ if in line 6 we take the minimum over existing edges only. The only solution to which this version can converge is the correct shortest path weights. Moreover, the initial conditions $d^0[v]$ for $v \neq s$ are arbitrary. If there is no convergence after $n-1$ iterations, then there is a negative weight cycle in the graph.

Observe that convergence is equivalent to *Bellman's equation*:

$$d[v] = \min_{u \in V}\{d[u] + w(u, v)\}, \forall v \in V \setminus \{s\} \tag{1.5a}$$

$$d[s] = 0. \tag{1.5b}$$

Bellman-Ford-v2 can become equivalent to Bellman-Ford-v1 if we add self-loops (edges (u, u)) for all nodes $u \in V$, with weight 0.

1.2.3 FLOYD-WARSHALL

The Floyd-Warshall algorithm computes the shortest paths between all pairs of vertices (APSP), as opposed to just one source and all destinations.

It iterates over increasing sets of vertices allowed to be used in the paths. At step k of the outermost loop below, D_{ij}^k contains the shortest path weight from i to j among the paths that only use nodes $1, \ldots, k$ as intermediate nodes.

Recall that A is the weighted adjacency matrix of G.

```
Floyd-Warshall($A$)
1  $D^0 \leftarrow A$
2  for $k \leftarrow 1$ to $n$
3      do for $i \leftarrow 1$ to $n$
4          do for $j \leftarrow 1$ to $n$
5              do $D_{ij}^k \leftarrow \min\{D_{ij}^{k-1}, D_{ik}^{k-1} + D_{kj}^{k-1}\}$
```

Since line 5 is executed n^3 times, and each execution takes constant time, the time complexity of Floyd-Warshall is $\Theta(n^3)$.

1.3 DISTRIBUTED COMPUTATION OF SHORTEST PATHS

As we mentioned in the beginning of this chapter, the shortest path problem is relevant to the problem of routing in communication networks. Since the edge weights (link costs between the network nodes) may change during the lifetime of the network, e.g., due to link failures, it is important to be able to compute – initially – and then recompute the shortest paths if and when needed. Moreover, such a computation needs to take into account the distributed nature of the network. The ideal protocol, running continuously on all network nodes, would react to link cost changes and update the shortest path weights correctly and as efficiently as possible in terms of time to convergence and number of messages exchanged.

One way of carrying out the shortest path computation in a network would be by broadcasting the whole topology, and any changes thereafter, to each network node. Then, each node would run a centralized version of a shortest path algorithm to compute shortest paths to all other nodes. This is the way link-state routing protocols work, such as Open Shortest Path First (OSPF) [37] and Intermediate System to Intermediate System (IS-IS) [13]. Since link costs in real networks are never set to negative values, Dijkstra's algorithm can be used to compute shortest paths.

Distance-vector protocols, such as Routing Information Protocol (RIP) [26], operate differently. Each node has a vector of size n, the distance vector, with its estimated distances (shortest path weights) from itself to all other nodes. Nodes send their distance vectors to their neighbors, and then they update their own distance vector taking into account the link cost to each neighbor and the vector received from that neighbor. This exchanging and updating operation continues for the whole lifetime of the network.

In what follows, we will describe more precisely the operation of distance-vector protocols. In particular, we will see that a distance-vector protocol corresponds to n distributed versions of the Bellman-Ford algorithm, one for each destination node, running in parallel and independent of each other.

1.3.1 COMMUNICATION MODEL FOR TOTALLY ASYNCHRONOUS ALGORITHMS

When describing distributed algorithms, one has to define the timing rules for message exchanges between the nodes that participate in the algorithm. Here, we will mention only two sets of rules: the synchronous model, and the totally asynchronous model. Textbooks on distributed algorithms [9] provide more information about timing models, also called demons.

In the *synchronous* model, one assumes that nodes process and exchange information in a synchronized way, in rounds. In each round, nodes receive all messages sent to them in the previous round, process these messages, and potentially send out new messages. Only after all nodes have received, processed, and sent all appropriate messages can the next round begin. The rounds in this model are very similar to the iterations of the centralized version of an algorithm like Bellman-Ford.

In the *totally asynchronous* model, different nodes can process messages without waiting for other nodes to finish, and communication channels between nodes can introduce arbitrary delays. So, a node might not use the most recent version of a variable updated by a different node. This model is closer to the operation of a real communication network where message delays cause nodes to have to work on outdated information. This is the timing model we will be assuming for the description of the distributed version of the Bellman-Ford algorithm next.

1.3.2 DISTRIBUTED BELLMAN-FORD

In the distributed Bellman-Ford algorithm, at each time t, each node v has an estimate $d^t[v]$ of its own shortest distance to a given destination, say node 1. Also, node v has estimates of the shortest distance of each of its neighbors $u \in N(v)$ to the same destination: $d_v^t[u]$. Note that we are now referring to the single destination shortest path problem instead of the SSSP since it is the one used in routing.

We would like to assume as little as possible for the initial conditions at each node, for the reasons given in the beginning of the chapter. It turns out that the initial conditions can be arbitrary for all nodes except node 1:

$$d^t[1] = 0, \forall t \geq 0 \tag{1.6}$$
$$d^t_v[1] = 0, \forall t \geq 0,\ v \in N(1). \tag{1.7}$$

Nodes asynchronously receive their neighbors' estimates and update their local copies of their neighbors' estimates. Then, they compute new estimates of their own shortest distances. At node v the following happens:

$$d^t[v] = \min_{u \in N(v)} \{w(v, u) + d^t_v[u]\}. \tag{1.8}$$

After this update, node v sends these estimates to its neighbors. The algorithm converges within finite time to the correct shortest distance values, as long as:

- The nodes never stop sending/receiving estimates from their neighbors and computing new estimates of their own shortest distances.
- Nodes stop sending old distance estimates some finite time after computing them.

Because of the arbitrary choice of initial conditions, the algorithm may need an excessive number of iterations to terminate. This is called the "counting to infinity" problem. It occurs, for example, after the following sequence of events: Consider two successive links (u, v) and (v, w) on the shortest path from u to the destination. The link (v, w) breaks, i.e., its cost becomes infinite. Node u sends to v its estimate for its shortest path weight to the destination, and v thinks that u has a path with that cost to the destination. This, however, is not true since u's path was using the now broken link (v, w). But u is not aware of the link break, and v is not aware that u's path was using the link (v, w). Node v performs the update (1.8) and sends to u its new estimate. Node u, in turn, uses the new estimate received from v to update its own estimate, which u then sends to v. The message exchanges continue until the estimates reach the cost of the next shortest path to the destination, or, if no other path exists, until the estimates reach the value chosen by the algorithm to represent infinity.

CHAPTER 2
The Algebraic Path Problem

2.1 SEMIRINGS AND THE ALGEBRAIC PATH PROBLEM

The essence of the shortest path problem is the computation

$$d_{st} = \min_{p \in P^{st}} w(p), \tag{2.1}$$

where the weight of a path $p = (v_0, v_1, \dots, v_k)$ is equal to the sum of the weights of its edges:

$$w(p) = w(v_0, v_1) + w(v_1, v_2) + \dots + w(v_{k-1}, v_k). \tag{2.2}$$

Many more network (and not only) problems can be solved by finding appropriate operators to replace min and +. We will see examples in Chapter 4. We will denote by $\oplus$ the operator that replaces min and by $\otimes$ the operator that replaces +. However, the operators cannot be arbitrary: they have to satisfy, at the very least, some properties. If the operators, defined on an appropriate set, form a *semiring*, then the resulting problem is called the *algebraic path problem*.

Definition 2.1 A *semiring* is a set S, the *carrier set*, endowed with two binary operators $\oplus$ and $\otimes$ that satisfy the following properties:

- $(S, \oplus)$ is a commutative monoid, i.e., $\oplus$ is commutative, associative, with neutral element $\textcircled{0} \in S$:

$$\begin{aligned} a \oplus b &= b \oplus a \\ (a \oplus b) \oplus c &= a \oplus (b \oplus c) \\ a \oplus \textcircled{0} &= a. \end{aligned}$$

- $(S, \otimes)$ is a monoid, i.e., $\otimes$ is associative, with neutral element $\textcircled{1} \in S$. Also, $\textcircled{0}$ is absorbing for $\otimes$. We will be omitting the symbol $\otimes$ when it can be deduced from the context:

$$\begin{aligned} (ab)c &= a(bc) \\ a\textcircled{1} &= \textcircled{1}a = a \\ a\textcircled{0} &= \textcircled{0}a = \textcircled{0}. \end{aligned}$$

- $\otimes$ distributes over $\oplus$:

$$\begin{aligned} a(b \oplus c) &= ab \oplus ac \\ (b \oplus c)a &= ba \oplus ca. \end{aligned}$$

The algebraic path problem is the computation of the $\oplus$-sum of the weights of all the paths p from s to t (see Figure 2.1):

$$d_{st} = \bigoplus_{p \in P^{st}} w(p), \tag{2.3}$$

where the weight of a path p is computed from the weights of its edges using the $\otimes$ operator:

$$w(p) = w(v_1, v_2) \otimes w(v_2, v_3) \otimes \ldots \otimes w(v_{k-1}, v_k). \tag{2.4}$$

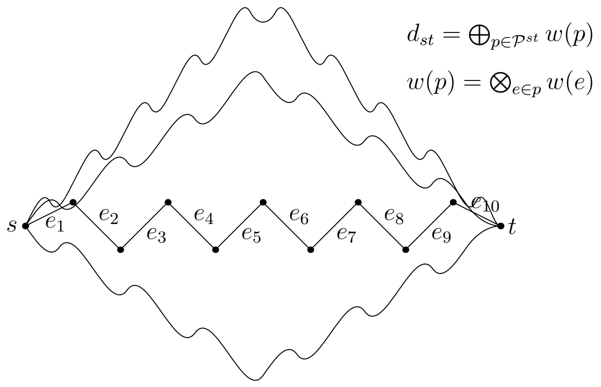
$d_{st} = \bigoplus_{p \in \mathcal{P}^{st}} w(p)$
$w(p) = \bigotimes_{e \in p} w(e)$
s
e_1
e_2
e_3
e_4
e_5
e_6
e_7
e_8
e_9
e_{10}
t

Figure 2.1: The algebraic path problem computes the $\oplus$-sum of the weights of all paths from a source s to a destination d. Each path weight is the $\otimes$-product of the weights of its edges.

2.1.1 WHY SEMIRINGS?

Let us now examine more closely the relevance of the semiring properties.

Associativity and non-commutativity of $\otimes$ For computing the path weight from its constituent edge weights, associativity means that it should not matter in which sequence one performs the pairwise operations. But since, in general, we may need to preserve the order of the edges on the path, $\otimes$ is not necessarily commutative. If it is, the semiring is called *commutative*, as is the case for the $(\min, +)$ semiring. Note that the non-commutativity of $\otimes$ implies that the weight of a cycle is not uniquely determined: one also has to specify an initial vertex. For each choice of initial vertex, the weight of the corresponding cycle is in general different.

Commutativity and Associativity of $\oplus$ The $\oplus$ operator aggregates the weights of a set of paths. A set does not imply an order among its elements, therefore $\oplus$ applied to $\{a, b\}$ should give the same result as $\oplus$ applied to $\{b, a\}$. Moreover, any sequence of pairwise aggregations should have the same result.

Neutral and absorbing elements The element $⓪$ represents the weight of a non-existing edge/path. So, it is ignored when aggregated by $\oplus$. Also, it is natural that the weight of any edge/path extended with a non-existing edge/path is also $⓪$. The element $①$ is more a technical requirement. Sometimes, it corresponds to the best possible path weight.

We can always add neutral elements for both $\oplus$ and $\otimes$ if they do not exist. For $\oplus$, for example, we just define a new element $⓪$ with the property

$$a \oplus ⓪ = ⓪ \oplus a = a, \forall a \in S, \tag{2.5}$$

and we can also define that

$$a \otimes ⓪ = ⓪ \otimes a = ⓪, \forall a \in S. \tag{2.6}$$

However, it should be the case that $⓪ \neq ①$; otherwise, it can be proven that $S = \{⓪\}$.

Distributivity of $\otimes$ over $\oplus$ Distributivity comes into play when aggregating paths with overlapping edges. If the paths being aggregated are independent, then distributivity is irrelevant. But if there is an edge overlap, the APP definition dictates that path weights should be computed individually with $\otimes$, and only then it is aggregated with $\oplus$. However, this is inefficient if there are many edge overlaps. The number of paths is exponential in the number of nodes in the graph, but this complexity can be severely decreased if distributivity holds. With distributivity, a common edge can be "factored out" and be taken into account in the calculations only once for all the paths that go through it. This is in general what is leveraged by the algorithms that we will present later in the book.

Absence of distributivity or $\otimes$-associativity When modeling a path problem, it might be the case that the most natural way of assigning edge weights does not lead to a semiring. Nevertheless, it might still be possible to find appropriate weights and operators that solve the same problem but satisfy the semiring properties. When distributivity, $\otimes$-associativity, or both are lacking, Section 5.1 provides hints for how to proceed.

2.1.2 ORDERED SEMIRINGS

Semirings in which an order relation can be defined form an important special class.

Definition 2.2 A binary relation $\preceq$ on a set S is an *order* if it is reflexive, transitive, and antisymmetric. That is, for all $a, b, c \in S$ the following statements hold:

- (reflexivity) $a \preceq a$.
- (transitivity) If $a \preceq b$ and $b \preceq c$, then $a \preceq c$.
- (antisymmetry) If $a \preceq b$ and $b \preceq a$, then $a = b$.

Definition 2.3 An order relation $\preceq$ on S is a *total order* if for all $a, b \in S$, it holds that either $a \preceq b$, or $b \preceq a$. That is, all pairs of elements of S can be compared.

Definition 2.4 A semiring is called a *dioid* if it is *canonically ordered*. That is, if the following relation $\preceq$ is an order on S:

$$a \preceq b \Leftrightarrow \exists c \in S : a \oplus c = b. \tag{2.7}$$

Is there a smallest element in any semiring? That is, is there an element $x \in S$, such that $\forall a \in S, x \preceq a$? What about a largest element?

We can see that the properties of this order depend only on the properties of $\oplus$. Because of the properties of $\oplus$ within a semiring, reflexivity and transitivity are automatically satisfied; antisymmetry is not. Actually, if $\oplus$ is such that antisymmetry is satisfied (i.e., the semiring is a dioid), then the semiring cannot be a ring. In other words, dioids and rings are mutually exclusive structures. Note that there are semirings that are neither dioids, nor rings.

If $\oplus$ is idempotent ($\forall a \in S, a \oplus a = a$), then S is canonically ordered since the antisymmetry of $\preceq$ can be deduced.

If $\oplus$ is selective ($\forall a, b \in S, a \oplus b = a$ or b), then S is canonically ordered, and the order $\preceq$ is total: Selectivity implies idempotency, and it also implies that for any pair $a, b \in S$, either $a \preceq b$ or $b \preceq a$.

In a dioid, the direction of the order $\preceq$ is preserved when multiplying both sides with the same element or adding to both sides the same element:

$$a \preceq b \Rightarrow \forall x \in S, \begin{cases} a \oplus x \preceq b \oplus x, \\ a \otimes x \preceq b \otimes x. \end{cases} \tag{2.8}$$

We can also multiply or add inequalities:

$$a \preceq b, c \preceq d \Rightarrow \begin{cases} a \oplus c \preceq b \oplus d, \\ a \otimes c \preceq b \otimes d. \end{cases} \tag{2.9}$$

2.2 CREATING NEW SEMIRINGS

Let $(S, \oplus, \otimes)$ be a semiring. We will consider methods that can be used to construct new semirings from S. Such methods are useful when modeling a new problem.

2.2.1 VECTORS AND MATRICES

Vectors and matrices whose elements are in S, also form a semiring with addition being the elementwise $\oplus$-addition, and multiplication being the generalization of the usual matrix multiplication.

$$A \oplus B = \left(a_{ij} \oplus b_{ij}\right) \tag{2.10}$$

$$A \otimes B = \left(\bigoplus_k a_{ik} \otimes b_{kj}\right). \tag{2.11}$$

For the connection between matrix powers and paths of an appropriately defined graph, see Section 3.1.

2.2.2 CONVOLUTION

If $(M, *)$ is a monoid with identity $\bar{1}$, then the set of functions $f : M \longrightarrow S$ forms a semiring under $+$ and $\odot$:

$$(f + g)(m) = f(m) \oplus g(m) \tag{2.12}$$

$$(f \odot g)(m) = \bigoplus_{m' * m'' = m} f(m') \otimes g(m''). \tag{2.13}$$

For example, let the monoid M have two elements $\{\bar{1}, x\}$, with the operation $*$ defined as

$$\bar{1} * \bar{1} = \bar{1} \qquad \bar{1} * x = x \tag{2.14}$$

$$x * \bar{1} = x \qquad x * x = \bar{1}. \tag{2.15}$$

Then, the $\odot$ operator of the resulting semiring is

$$(f \odot g)(\bar{1}) = f(\bar{1})g(\bar{1}) \oplus f(x)g(x) \tag{2.16}$$

$$(f \odot g)(x) = f(\bar{1})g(x) \oplus f(x)g(\bar{1}). \tag{2.17}$$

It is isomorphic to a semiring on $S \times S$ with operators

$$(a, b) + (c, d) = (a \oplus c, b \oplus d) \tag{2.18}$$

$$(a, b) \odot (c, d) = (ac \oplus bd, ad \oplus bc). \tag{2.19}$$

Now, consider the following monoid: $M = \{x^k, k \in \mathbb{N}\}$, where x is a formal variable, and $*$ is defined by $x^{k_1} * x^{k_2} = x^{k_1+k_2}$, $k_1, k_2 \in \mathbb{N}$. The functions $f : M \longrightarrow S$ can be viewed as formal power series in x with coefficients in S: E.g., $f(x_k) = a_k \Leftrightarrow f = a_0 \oplus a_1 x \oplus a_2 x^2 \oplus \ldots \oplus a_n x^n \oplus \mathbb{0} x^{n+1} \oplus \ldots$. Then,

$$f + g = \bigoplus_{k=0}^{\infty} (a_k \oplus b_k) x^k \tag{2.20}$$

$$f \odot g = \bigoplus_{k=0}^{\infty} c_k x^k, \tag{2.21}$$

where $c_k = \bigoplus_{i+j=k} a_i b_j$.

Another useful monoid to keep in mind is the set of paths of a graph, under concatenation of paths.

2.2.3 REDUCTION SEMIRINGS

A function $f : S \to S$ is called a *reduction* [50] if and only if

$$f(\mathbb{0}) = \mathbb{0} \tag{2.22}$$
$$f(\mathbb{1}) = \mathbb{1} \tag{2.23}$$
$$f(a \oplus b) = f(f(a) \oplus b), \forall a, b \in S \tag{2.24}$$
$$f(ab) = f(f(a)b) = f(af(b)), \forall a, b \in S. \tag{2.25}$$

Given a reduction function f, the triplet $(S_f, +, \odot)$ is a semiring, where

$$S_f = \{a \in S | f(a) = a\}, \tag{2.26}$$

and

$$a + b = f(a \oplus b) \tag{2.27}$$
$$a \odot b = f(ab). \tag{2.28}$$

An example where a reduction function is used is the network reliability semiring, given in Section 4.8.

2.2.4 SEMIRINGS OF ENDOMORPHISMS

Consider the set H of endomorphisms of S, i.e., the set of functions from S to S such that

$$\forall h \in H, \forall a, b \in S, h(a \oplus b) = h(a) \oplus h(b) \tag{2.29}$$
$$\forall h \in H, h(\mathbb{0}) = \mathbb{0}. \tag{2.30}$$

The first way to create a new semiring is by defining addition and multiplication in H as [23] [28]

$$(h_1 + h_2)(a) = h_1(a) \oplus h_2(a) \tag{2.31}$$
$$h_1 \odot h_2(a) = h_2(h_1(a)). \tag{2.32}$$

That is, endomorphism addition is defined as pointwise addition, and endomorphism addition is defined as composition.

Then, the triplet $(H, +, \odot)$ is a semiring. If $(S, \oplus, \otimes)$ is a dioid then $(H, +, \odot)$ is a dioid. Note that the operator $\otimes$ is not used at all in the definition of H, so all we need from S are the properties of $\oplus$ and the canonical order.

Note that the result of finding A^* in this setting amounts to finding functions, rather than numbers. In general, this means finding the image of every element of S for which a function is defined; this might not be computationally tractable. So, the algorithms for computing A^* will be used to compute the image of a single element $a \in S$.

An application of this semiring creation technique can be found in Section 4.7.

Alternatively, if we fix two endomorphisms $h_1, h_2 \in H$, then the set S together with the two following operations is a semiring [28]:

$$a + b = a \oplus b \tag{2.33}$$

$$a \odot b = h_1(a) \otimes h_2(b). \tag{2.34}$$

CHAPTER 3

Properties and Computation of Solutions

3.1 ALTERNATIVE VIEWPOINTS: PATHS AND MATRICES

We have described the algebraic path problem as an $\oplus$-sum of $s - t$ paths in a graph $G = (V, E)$, $s, t \in V$. We call this the path viewpoint:

$$d_{st} = \bigoplus_{p \in P^{st}} w(p). \tag{3.1}$$

Note that the set of $s - t$ paths will in general have an infinite number of terms. To deal with issues of infinite sums, we will define for every element $a \in S$

- $a^{(k)} = \textcircled{1} \oplus a \oplus a^2 \oplus \ldots \oplus a^k$, $k \in \mathbb{N}$,
- $a^* = \lim_{k \to \infty} a^{(k)}$, if the sum exists. This is the *closure* of a.

There is an equivalent formulation of the algebraic path problem as a matrix equation, at the heart of which is Bellman's equation (1.5):

$$d[v] = \min_{u \in V}\{d[u] + w(u, v)\}, \forall v \in V \setminus \{s\} \tag{3.2a}$$
$$d[s] = 0. \tag{3.2b}$$

We write Bellman's equation in matrix form using the operators of the semiring $(\mathbb{R} \cup \infty, \min, +)$. Matrix multiplication is done in the usual way, except that we use the semiring addition and multiplication:

$$A \oplus B = C, [c_{ij}] = \left[a_{ij} \oplus b_{ij}\right], \tag{3.3}$$

$$AB = C, [c_{ij}] = \left[\bigoplus_k a_{ik} \otimes b_{kj}\right]. \tag{3.4}$$

Under these two operations, the set of square matrices with elements from S is a semiring.

The matrix formulation of the algebraic path problem is

$$d^T = d^T A \oplus \textcircled{1}_s^T, \tag{3.5}$$

where d^T is the row vector of shortest path weights from vertex s to all vertices, A is the weighted adjacency matrix of the graph G, and $\textcircled{1}_s^T$ is the row vector with all entries equal to $\textcircled{0}$, except the s-th column entry which is equal to $\textcircled{1}$.

The Bellman-Ford iteration can also be seen as a matrix iteration of the previous equation:

$$(d^{k+1})^T = (d^k)^T A \oplus \textcircled{1}_s^T. \tag{3.6}$$

For the all pairs shortest paths problem, the matrix formulation is

$$D = DA \oplus I, \tag{3.7}$$

where $D = \left(d_{ij}\right)$ contains the shortest path weights from i to j, for all $i, j \in V$, and $I = \begin{pmatrix} \textcircled{1} & \textcircled{0} & \cdots & \textcircled{0} \\ \textcircled{0} & \textcircled{1} & \cdots & \textcircled{0} \\ \vdots & \vdots & \ddots & \vdots \\ \textcircled{0} & \textcircled{0} & \cdots & \textcircled{1} \end{pmatrix}$.

Repeatedly substituting $D = DA \oplus I$ into the right-hand side of (3.7) we get

$$\begin{aligned} D &= DA \oplus I && (3.8)\\ &= (DA \oplus I)A \oplus I = DA^2 \oplus A \oplus I && (3.9)\\ &= (DA \oplus I)A^2 \oplus A \oplus I = DA^3 \oplus A^2 \oplus A \oplus I && (3.10)\\ &= \ldots && (3.11)\\ &= DA^{k+1} \oplus A^k \oplus \ldots \oplus A^2 \oplus A \oplus I. && (3.12) \end{aligned}$$

If the series converges, the limit matrix is A^*:

$$A^* = \lim_{k \to \infty} I \oplus A \oplus A^2 \oplus \ldots \oplus A^k. \tag{3.13}$$

By directly substituting A^* for D into (3.7), we can verify that A^* is a solution of (3.7).

For each matrix A^k, the element A_{ij}^k is equal to the sum of the weights of all k-edge paths from i to j. For $k = 0$, A^0 is defined to be I. The partial sums in (3.13) can then be interpreted similarly: The paths summed have up to k edges, instead of exactly k edges.

We can see that the elements of the matrices A^k for $k \geq n$ necessarily correspond to the weights of paths that contain cycles. So, the convergence of (3.13) is closely linked to the weights of paths that traverse cycles an increasing number of times. For example, if there are no cycles, then there are no paths with n or more edges, and so $A^n = A^{n+1} = \ldots = 0$. Then, from (3.13), we get that $A^* = \bigoplus_{k=0}^{n-1} A^k$.

3.1.1 EXISTENCE OF A^*

We will give some sufficient conditions for the existence of A^*. First, we need to define the concept of a p-stable element of a semiring.

Definition 3.1 An element a of a semiring S is called *p-stable*, if there exists $p \in \mathbb{N}$ such that $a^{(p)} = a^{(p+1)} = \ldots$.

Under any of the conditions below, the convergence to A^* happens within a finite number of steps. The conditions guarantee that after traversing any existing cycles enough times, the sum no longer changes.

- For every cycle c in $G(A)$ it holds that

$$w(c) \oplus \textcircled{1} = \textcircled{1}. \tag{3.14}$$

  In this case, the limit is reached for $k \leq n - 1$. This condition corresponds to the non-negativity of cycles in the classical shortest path problem.

- $\otimes$ is commutative and, for every cycle c in $G(A)$, $w(c)$ is p-stable for some $p \geq 1$. In this case, the limit is reached for $k \leq n - 1 + pnt$ steps, where t is the number of elementary cycles of $G(A)$.

- The set of entries a_{ij} of A is *p-nilpotent*: There exists a $p \in \mathbb{N}$, such that, for any sequence of $p+1$ elements of A, $a_1, a_2, \ldots, a_{p+1}$, it holds that $a_1 \otimes a_2 \otimes \ldots \otimes a_{p+1} = \textcircled{0}$. The limit is reached for $k \leq p$.

3.1.2 UNIQUENESS OF A^*

In general, the solution of (3.7) is not unique. But if the semiring is a dioid, then the solution A^* is minimal with respect to the canonical order. Assume there exists $k_0 \in \mathbb{N}$ such that $A^* = \bigoplus_{k=0}^{k_0} A^k$. Then, from the last line of (3.8), we can see that for $k > k_0$ we have from (3.8) that any D should satisfy

$$D = DA^k \oplus A^*. \tag{3.15}$$

We conclude that A^* is $\preceq$-smaller than any matrix D that satisfies (3.7). Therefore, if there is another matrix B^* with the same property, then we would have $A^* \preceq B^*$ and $B^* \preceq A^*$, that is, $A^* = B^*$.

3.2 EDGE SENSITIVITIES

We now introduce edge sensitivities for the algebraic path problem. The edge weights will be restricted to belong to a commutative selective dioid S where $\textcircled{1}$ is the $\preceq$-largest element. In such a dioid, even if it is not commutative, we can prove that extending a path/edge with another path/edge never increases its weight.

$$\begin{aligned}
\forall a, b \in S, a \otimes b \oplus a &= a \otimes (b \oplus \textcircled{1}) && (3.16)\\
&= a \otimes \textcircled{1} && (3.17)\\
&= a && (3.18)\\
\Rightarrow \forall a, b \in S, a \otimes b &\preceq a. && (3.19)
\end{aligned}$$

In the algebraic path sensitivity problem, we are given a connected undirected edge-weighted graph $G = (V, E, w)$, with associated weighted adjacency matrix A, and an edge $e \in E$. We want

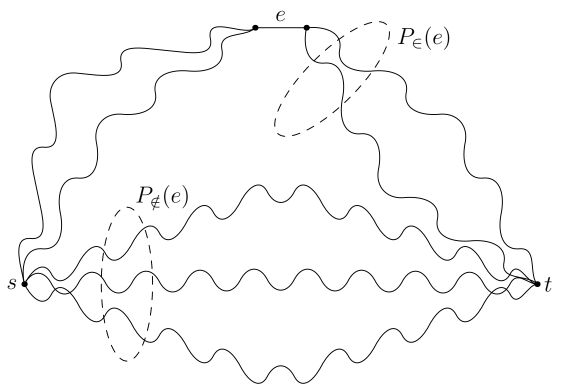

Figure 3.1: Partitioning the $s - t$ paths according to whether they use edge e or not.

to see the effect of changing the weight $w(e)$ on the optimal paths between any pair of nodes. We assume that for each pair of nodes there exists exactly one path in G that achieves the optimal weight.

For a given source-destination node pair (s, t), and an edge $e \in E$, we define the upper and lower tolerance of e with respect to the pair (s, t): The *upper tolerance* $u(e)$ of e is the $\preceq$-maximum value that $w(e)$ can have without violating the optimality of the $s - t$ path that was optimal in G. The *lower tolerance* $l(e)$ is similarly defined as the $\preceq$-minimum value that $w(e)$ can have without violating the optimality of the $s - t$ path that was optimal in G.

In what follows, we will show how to compute upper and lower tolerances of an edge e with respect to all (s, t) pairs.

Take an arbitrary (s, t) pair, and recall that P^{st} is the set of all $s - t$ paths. For brevity, we will be omitting the superscript and use instead just P. We partition P into two subsets $P_{\in}(e)$, $P_{\notin}(e)$ according to whether a path includes edge e or not (Figure 3.1):

$$P_{\in}(e) = \{p \in P | e \in p\}, \tag{3.20}$$

$$P_{\notin}(e) = P \setminus P_{\in}(e). \tag{3.21}$$

We define $w(X)$, where X is a set of paths with the same source and the same destination, to be the $\oplus$-sum of the weights of the paths in X. So, we have that

$$w(P) = A^*[s, t] = w(P_{\in}(e)) \oplus w(P_{\notin}(e)), \tag{3.22}$$

where by $A^*[s, t]$, we denote the (s, t) entry of the matrix A^*.

We will now proceed to compute the weights $w(P_{\in}(e))$ and $w(P_{\notin}(e))$.

We write $w(P_{\in}(e))$ so as to isolate the effect of $w(e)$:

$$w(P_{\in}(e)) = \left[\left(\bigoplus_{p\in P^{si}} w(p)\right) w(e) \left(\bigoplus_{p\in P^{jt}} w(p)\right)\right] \oplus \left[\left(\bigoplus_{p\in P^{sj}} w(p)\right) w(e) \left(\bigoplus_{p\in P^{it}} w(p)\right)\right]. \tag{3.23}$$

Since the semiring S is commutative, we can factor out $w(e)$:

$$w(P_{\in}(e)) = w(e)\left[\left(\bigoplus_{p\in P^{si}} w(p)\right) \textcircled{1} \left(\bigoplus_{p\in P^{jt}} w(p)\right) \oplus \left(\bigoplus_{p\in P^{sj}} w(p)\right) \textcircled{1} \left(\bigoplus_{p\in P^{it}} w(p)\right)\right]. \tag{3.24}$$

But the quantity inside the square brackets is exactly equal to the optimal $s - t$ path weight in a graph with $w(e) = \textcircled{1}$. We denote by $A_{\textcircled{1}}$ the weighted adjacency matrix of the graph G where the weight of edge e has been replaced with $\textcircled{1}$. Then, we can write (3.24) as

$$w(P_{\in}(e)) = w(e)A^*_{\textcircled{1}}[s,t]. \tag{3.25}$$

By $A^*_{\textcircled{1}}$ we denote the matrix $\left(A_{\textcircled{1}}\right)^*$, and similarly for $A^*_{\textcircled{0}}$.

For $w(P_{\notin}(e))$, all we need to observe is that the total weight of the $s - t$ paths that do not use edge e is equal to the total weight of all $s - t$ paths in a graph where the weight of edge e is $\textcircled{0}$. As previously, we can write

$$w(P_{\notin}(e)) = A^*_{\textcircled{0}}[s,t]. \tag{3.26}$$

Substituting $w(P_{\in}(e))$ and $w(P_{\notin}(e))$ into (3.22) we get

$$A^*[s,t] = w(e)A^*_{\textcircled{1}}[s,t] \oplus A^*_{\textcircled{0}}[s,t]. \tag{3.27}$$

The optimal path weight $A^*[s,t]$ is either equal to $w(e)A^*_{\textcircled{1}}[s,t]$, if e is on the optimal path, or equal to $A^*_{\textcircled{0}}[s,t]$, if e is not on the optimal path.

Edge e is on the optimal path if and only if the optimal path weight $A^*[s,t]$ is equal to $w(e)A^*_{\textcircled{1}}[s,t]$. In this case, the lower tolerance is the lowest value of $w(e)$ for which $w(e)A^*_{\textcircled{1}}[s,t]$ still exceeds $A^*_{\textcircled{0}}[s,t]$. The upper tolerance is $\textcircled{1}$.

Edge e is not on the optimal path if and only if the optimal path weight $A^*[s,t]$ is equal to $A^*_{\textcircled{0}}[s,t]$. In this case, the lower tolerance is $\textcircled{0}$. The upper tolerance is the largest value of $w(e)$ for which $w(e)A^*_{\textcircled{1}}[s,t]$ does not exceed $A^*_{\textcircled{0}}[s,t]$.

3.3 CENTRALIZED COMPUTATION OF A^*

If $\oplus$ is idempotent, then it can be seen that $(I \oplus A)^k = \bigoplus_{i=1}^{k} A^i$. Then, by repeated squaring of $I \oplus A$ we can compute A^* if, for example, we know that convergence happens in a finite number of iterations.

3.3.1 GENERALIZATION OF BELLMAN-FORD

By transforming Bellman-Ford to semiring notation, we get the following algorithm. Recall that $|V| = n$, and $\textcircled{1}_s = (\textcircled{0}, \ldots, \textcircled{0}, \textcircled{1}, \textcircled{0}, \ldots, \textcircled{0})$, where $\textcircled{1}$ is at the s-th position.

```
Semiring Bellman-Ford($G, w, s$)
1  $d \leftarrow \textcircled{1}_s$
2  for $i \leftarrow 1$ to $n - 1$
3       do $d \leftarrow \textcircled{1}_s \oplus d \otimes W$
```

By extension of the relaxation operation (1.4), we call *algebraic relaxation* the operation in line 3 above. For a single element $d[v]$, $v \in V \setminus \{s\}$ of the vector d, not corresponding to the source node s, algebraic relaxation means

$$d[v] = \bigoplus_{u \in V} \{d[u] \otimes w(u, v)\}. \tag{3.28}$$

Bellman-Ford does not require both left and right distributivity. Only one of the two is necessary.

3.3.2 GENERALIZATION OF DIJKSTRA

The semiring has to be a selective dioid where $\textcircled{1}$ is the largest element. By transforming Dijkstra to semiring notation, we get the following algorithm:

```
Semiring Dijkstra($G, w, s$)
1  $d[s] \leftarrow \textcircled{1}$
2  for $v \in V \setminus \{s\}$
3       do $d[v] \leftarrow \textcircled{0}$
4  $Q \leftarrow V$
5  while $Q \neq \emptyset$
6       do $u \leftarrow \arg \oplus \{d[v] | v \in Q\}$
7          $Q \leftarrow Q \setminus \{u\}$
8          for each vertex $v \in Q$
9               do $d[v] \leftarrow d[v] \oplus d[u] \otimes w(u, v)$
```

The line $u \leftarrow \arg \oplus \{d[v] | v \in Q\}$ means that to the variable u we assign the vertex $v \in Q$ that satisfies $d[v] = \bigoplus_{u \in Q} d[u]$. Such a vertex exists and is unique if the semiring is a selective dioid but not in general. The condition for $①$ being the largest element is equivalent to the condition for non-negativity of edges in the classical algorithm of Dijkstra.

3.3.3 GENERALIZATION OF FLOYD-WARSHALL

To generalize the Floyd-Warshall algorithm to semirings, we have to be able to compute the closures a^* of the semiring elements a.

```
Semiring Floyd-Warshall($A$)
1  for $k \leftarrow 1$ to $n$
2      do $a_{kk} \leftarrow (a_{kk})^*$
3         for $i \leftarrow 1$ to $n, i \neq k$
4             do $a_{ik} \leftarrow a_{ik} \otimes a_{kk}$
5         for $i \leftarrow 1$ to $n, i \neq k$
6             do for $j \leftarrow 1$ to $n, j \neq k$
7                    do $a_{ij} \leftarrow a_{ij} \oplus a_{ik} \otimes a_{kj}$
8                for $j \leftarrow 1$ to $n, j \neq k$
9                    do $a_{kj} \leftarrow a_{kk} \otimes a_{kj}$
```

3.4 DECENTRALIZED COMPUTATION OF A^*

Define the *distributed algebraic path* computation as follows:

Definition 3.2 Given a graph $G = (V, E)$ and a semiring S, each node $v \in V$ owns a variable $d[v]$ and it continually performs the algebraic relaxation step $d[v] = \bigoplus_{u \in V} \{d[u] \otimes w(u, v)\}$ with the variables received from its neighbors and sends its updated variable to its neighbors.

We focus on the totally asynchronous timing model for message exchanges among the nodes. We are interested in the following question: What are the necessary and sufficient properties that the operators $\oplus$ and $\otimes$ of S need to have in order for the distributed algebraic path computation to converge?

3.4.1 FORMAL MODEL OF THE DISTRIBUTED ALGEBRAIC PATH PROBLEM

In previous chapters, we have dealt with the solution of the system

$$x = Ax \oplus b, \tag{3.29}$$

where the vector b was actually equal to $①_s$ for a source vertex s, but b can be more general. Also, we were describing the system with row vectors, when the vector x would multiply A on the left, instead of on the right as now. But the two versions are completely equivalent.

We will now describe a model and convergence results for the distributed asynchronous iterative computation of the above system when the timing model is the totally asynchronous one. As a reminder, the synchronous iteration is

$$y(k+1) = Ay(k) \oplus b, k = 0, 1, \dots \tag{3.30}$$

In the distributed version, we assume that each component $x_i \in X_i, i = 1, \dots, n$ of x is assigned to a processor i. We denote by $x_i(t)$ the value of x_i at time t. We call X the Cartesian product of the $X_i : X = X_1 \times X_2 \times \dots \times X_n$. Each processor updates its value x_i at times $t \in T_i$, using the locally stored values of the components of the vector x. Note that these locally stored values may not be the most up-to-date versions. So, the update at processor i at a time $t \in T_i$ is

$$x_i(t) = \bigoplus_j a_{ij} x_j(\tau_{ji}(t)) \oplus b_i, \tag{3.31}$$

where $\tau_{ji}(t)$ satisfies

$$0 \leq \tau_{ji}(t) \leq t. \tag{3.32}$$

The functions $\tau_{ji}(t)$ are used to model the communication delay from processor j to processor i. A conceptual model is that processor j sends the value x_j to processor i at time $\tau_{ji}(t)$; processor i wakes up at time $t \in T_i$, collects the values sent to it up to that time, and performs the update. Note that the formal model covers even the case where $j = i$, i.e., when a processor does not have the most up-to-date version of its own variable.

Apart from the totally asynchronous timing model, we make some more assumptions on the schedule of updates:

- Each processor performs the update (relaxation step) infinitely often, i.e., the sets T_i are infinite (non-starvation condition), and

- Old information can be used for some time, but is purged from the system eventually, i.e., if $t_k \in T_i, k = 1, 2, \dots$ then $\lim_{k\to\infty} t_k = \infty \Rightarrow \forall j \lim_{k\to\infty} t_{ji}(t_k) = \infty$.

3.4.2 CONDITIONS FOR THE CONVERGENCE OF THE DISTRIBUTED COMPUTATION

A very general result under the schedule of the previous section is the Asynchronous Convergence Theorem [9, Ch.6], which gives sufficient conditions for convergence.

Theorem 3.3 *If there is a sequence of non-empty sets* $\{X(k)\}$

$$\cdots \subset X(k+1) \subset X(k) \subset \cdots \subset X \tag{3.33}$$

such that

- *(Synchronous convergence condition)*

$$x \in X(k) \Rightarrow Ax \oplus b \in X(k+1), \forall k$$

and if $\{y^k\}$ *is a sequence such that* $y^k \in X(k), \forall k$, *then every limit point of* $\{y^k\}$ *is a solution of* $x = Ax \oplus b$.

- *(Box Condition) For every* k, *there exist sets* $X_i(k) \subset X_i$ *such that*

$$X(k) = X_1(k) \times X_2(k) \times \cdots \times X_n(k),$$

and the initial value of the vector $x(0) = (x_i(0), i = 1, \ldots, n)$ *is in the set* $X(0)$, *then every limit point of* $\{x^k\}$ *is a solution of* $x = Ax \oplus b$.

Üresin and Dubois [49] show an equivalent form of these conditions to also be *necessary* when the set X is finite. They then make the point that X will always be finite in digital computers with finite memory space and word length.

3.4.3 CONDITIONS FOR CONVERGENCE IN DIOIDS

Of particular interest for us is the case when the semiring is a dioid. Then, it turns out that the mapping $x \rightarrow Ax \oplus b$ is monotone:

$$\forall z, w \in X, z \preceq w \Rightarrow Az \oplus b \preceq Aw \oplus b. \tag{3.34}$$

In this case, the asynchronous iteration with initial data $x(0) \in X$ converges in a finite number of steps if

UD1 For all $x \in X$, $Ax \oplus b \in X$, i.e., the iteration is closed in X.

UD2 The synchronous iteration converges in a finite number of steps starting from initial data $y(0) = x(0)$.

UD3 The synchronous iteration is non-expansive at all steps starting from initial data $y(0) = x(0)$: $y(k) \succeq Ay(k) \oplus b, k = 0, 1, \ldots$.

If X is finite, conditions **UD2** and **UD3** can be replaced with **UD3** on all elements of X.

An alternative set of sufficient conditions is [9, 6.4] that the solution is unique, and that two vectors $\underline{x}, \overline{x} \in X$ exist, and are such that

- $\underline{x} \preceq A\underline{x} \oplus b \preceq A\overline{x} \oplus b \preceq \overline{x}$.
- The synchronous iteration converges with initial data $\underline{x}$ and with initial data $\overline{x}$.
- The initial data $x(0)$ of the asynchronous iteration satisfies $\underline{x} \preceq x(0) \preceq \overline{x}$.

Then, we could define the sets $X(k)$ required by the Asynchronous Convergence Theorem as

$$X(k) = \{x \in X | A^k \underline{x} \oplus (A^{k-1} \oplus \ldots \oplus I)b \preceq x \preceq A^k \overline{x} \oplus (A^{k-1} \oplus \ldots \oplus I)b\}. \tag{3.35}$$

We know that $⓪$ is the $\preceq$-minimum element of S. If there is also a $\preceq$-largest element, say $\top$, then two candidates for $\underline{x}$ and $\overline{x}$ are

- For $\underline{x}$: $(⓪, \ldots, ⓪)$.
- For $\overline{x}$: $(\top, \ldots, \top)$.

CHAPTER 4
Applications

In this chapter, we describe a variety of applications of semirings to path problems. That is, we describe the application area, and provide the semiring that solves the corresponding problem. References are given to the literature where these application were first proposed; if none are given, the default references are the tutorial paper by Rote [44] and the book by Gondran and Minoux [23]. A good exercise for the reader is to verify the semiring properties of each pair of operators. At times, we show how semiring creation techniques from Section 2.2 were (or could have been) used to find the new semirings.

4.1 PATH ENUMERATION

Perhaps the most general path problem solvable with a semiring is that of enumerating all the paths between two vertices of a graph.

The carrier set S is the powerset of the set of ordered vertex sequences. For a graph $G = (V, E)$, where $V = \{v_1, v_2, \ldots, v_n\}$, the set of ordered vertex sequences is $V \cup V^2 \cup \ldots \cup V^n$. So, the elements of S are the subsets of $V \cup V^2 \cup \ldots \cup V^n$, which include the empty set.

The addition $\oplus$ of two sets of sequences is defined to be their set union. The result of the multiplication $\otimes$ of two sets of sequences is defined to be the set whose members are concatenations of sequence pairs, where the first member of the pair is taken from the first set and the second member from the second set:

$$\begin{aligned} W_1 \otimes W_2 = \{w_1 w_2 | w_1 \in W_1, w_2 \in W_2, &\text{ and the last vertex of } w_1 \\ &\text{coincides with the first vertex of } w_2.\}\,. \end{aligned} \tag{4.1}$$

In the concatenation $w_1 w_2$, we do not duplicate the common vertex (last vertex of w_1 and first of w_2).

For example,

$$\{(v_1, v_2), (v_1, v_3)\} \otimes \{(v_2, v_5), (v_1, v_4)\} = \{(v_1, v_2, v_5)\}. \tag{4.2}$$

The weight of edge $e = (i, j)$ is the element $\{(i, j)\} \in S$.

This problem is very related to finite automata and the formal languages that such automata accept [44]. Formal languages and parsing is another area where semirings find applications [24] [36] [17], but it is outside the scope of this book.

It is also possible to enumerate all paths that have a specific property P, for example, all elementary paths. One should define a reduction function (see Sec. 2.2.3) that operates on a member

s of S and returns only those elements of s that satisfy property P. Then, a construction as in Sec. 2.2.3 would give the appropriate semiring: One needs only to apply the reduction function to the result of $\oplus$ and of $\otimes$ as defined above.

4.2 EXPECTATION SEMIRINGS

The general model for the semirings of this section is a directed graph, where a probability $p_e \equiv p_{ij}$ is associated with each edge $e = (i, j)$. A conceptual model is that of a random walk on the graph, where, given that the walk has reached node i, edge $e = (i, j)$ is traversed with probability p_e.

Markov Chain semiring The nodes of the graph are the states of a Markov chain, and the edge probabilities are the Markov chain's one step transition probabilities. So, we require $\sum_j p_{ij} = 1, \forall i$. The semiring operators are the usual addition and multiplication and the carrier set is $[0, \infty) \cup \{\infty\}$.

The element (i, j) of the k-th power A^k of the weighted adjacency matrix A is equal to the probability that the Markov chain is in state j after step k, given that it was in state i after step 0.

Fletcher semiring We do not require $\sum_j p_{ij} = 1$, only that $\sum_j p_{ij} \leq 1$. It is possible that the random walk stops at vertex i with probability $1 - \sum_j p_{ij}$. We want to compute the expected number of visits of the random walk to a vertex t, given that it starts at vertex s.

The carrier set of the semiring is $S = [0, \infty) \cup \{\infty\}$, and the two operations are the usual addition and multiplication. We need the ∞ element to account for cases when there is a subgraph where the random walk might stay forever.

This semiring appeared at [19] as an example of a non-idempotent semiring that applies to a reasonable problem.

Eisner semiring It is required that $\sum_j p_{ij} = 1$. Also, a value $v_e \in \mathbb{R}$ is associated with each edge e. Traversal of edge e brings value (cost or benefit) v_e, and value accumulates additively along a path. The objective is to calculate the expected value of paths from a vertex s to a vertex t.

The carrier set of the semiring is $S = ([0, \infty) \times \mathbb{R})$, and the two operations are

$$(a, b) \oplus (c, d) = (a + c, b + d) \tag{4.3}$$
$$(a, b) \otimes (c, d) = (ac, ad + bc). \tag{4.4}$$

But to use this semiring for the problem described above, we set the weight of edge e to be $(p_e, p_e v_e)$. In other words, the second element of the weight is the expected value of edge $e = (i, j)$ given that we are at i. This semiring was used by Eisner in the context of natural language processing [17].

4.3 MINIMUM WEIGHT SPANNING TREE

We are given an undirected, connected graph G. In the minimum weight spanning tree problem, we want to find a subgraph of G that connects all its vertices ("spans" the graph G) and has minimum weight among all such spanning subgraphs. Any such subgraph is a tree (i.e., it is acyclic), and the minimum weight spanning tree (MST) is unique if the edge costs are distinct. The importance of the MST for networking is that it is the least cost subgraph that can be used for communication from any vertex to any vertex of G.

The crucial observation that we will use is that an edge (i, j) is not in the MST if and only if $w(i, j)$ is larger than the maximum weight of any edge along the $i - j$ path on the MST. Equivalently, an edge (i, j) is in the MST if and only if any other $i - j$ path on G has an edge with weight larger than $w(i, j)$ (see Fig. 4.1).

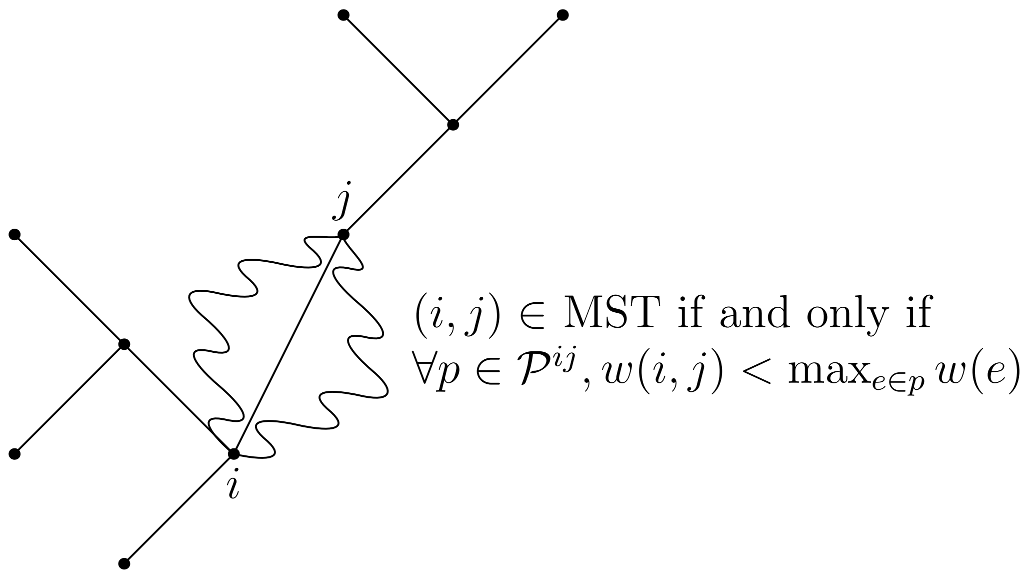
j
i
$(i, j) \in \text{MST if and only if}$
$\forall p \in \mathcal{P}^{ij}, w(i, j) < \max_{e \in p} w(e)$

Figure 4.1: Condition for an edge (i, j) to belong to the minimum spanning tree.

So, we need to compute the minimum cost paths between all pairs of vertices where the cost of a path is equal to the maximum of its edge weights. Then, if the minimum path cost between vertices i and j is equal to the weight of the edge (i, j), that edge would be on the MST, otherwise not.

The carrier set of the semiring is the set of possible edge weights $S = \mathbb{R}$, and the operations are $\oplus = \min$, $\otimes = \max$. Maggs and Plotkin [35] first proposed this semiring for constructing MSTs.

4.4 LONGEST PATH

In an edge-weighted directed graph G, we want to find the longest path from a source vertex s to a destination vertex t. For this, we use the semiring $(S, \oplus, \otimes) = (\mathbb{R} \cup \{-\infty\}, \max, +)$. The rest proceeds similarly to the shortest path semiring. For example, the problem is ill-defined if there are positive-weight cycles in the graph. Note that the longest *elementary* path problem is much harder: It is NP-complete. The longest path problem has attracted a lot of attention in the area of Performance Evaluation of Discrete Event Systems [3] (see also Section 5.2.1), and also applies to some problems of operation research: If the graph G describes a manufacturing process, then the longest path corresponds to the path that determines the rate at which products can be manufactured. Then, it is useful, for example, to identify an edge whose weight, if changed, would result in the biggest reduction in the longest path weight.

4.5 QUALITY OF SERVICE (QOS) ROUTING

In quality of service routing, we associate with each edge a measure of quality, such as delay, bandwidth, and loss. We want to compute the best path from a source to a destination according to that metric. Extensions include taking two metrics into account at the same time, with a given priority among them, or computing not only the best path, but also the second-best, third-best, up to k-th best path.

Shortest path Edge weights correspond to communication delays. We want to compute the shortest path from node s to node t. The semiring is $(S, \oplus, \otimes) = (\mathbb{R} \cup \{\infty\}, \min, +)$. This semiring was studied extensively in the first chapter of this book.

Widest path Edge weights correspond to transmission bandwidth/capacity. We want to compute the largest capacity path from node s to node t. Since the capacity of a path is equal to the bottleneck capacity of that path (the smallest along the path), the appropriate semiring is $(S, \oplus, \otimes) = ([0, \infty) \cup \{\infty\}, \max, \min)$.

Most reliable path The weight of an edge corresponds to its reliability: the probability of successful packet transmission along the edge. We want to compute the most reliable path from node s to node t. We assume that successful packet transmissions are independent across edges, so the reliability of a path is just the product of edge reliabilities. The appropriate semiring is $(S, \oplus, \otimes) = ([0, 1], \max, \times)$.

Widest-shortest path This is a multiobjective problem: Edge weights are pairs of values, corresponding to the delay and to the bandwidth of a link. Among the shortest paths, if there is more than one, we want to compute the one with largest bandwidth. The carrier set is $S = \{(d, b) \in$

$([0, \infty) \cup \{\infty\}, [0, \infty) \cup \{\infty\}\}$. Addition is selective:

$$(d_1, b_1) \oplus (d_2, b_2) = \begin{cases} (d_1, b_1) & \text{if } d_1 < d_2 \text{ or } d_1 = d_2 \text{ and } b_1 > b_2, \\ (d_2, b_2) & \text{otherwise.} \end{cases} \tag{4.5}$$

Multiplication combines the two multiplications of the shortest path and the widest path problem:

$$(d_1, b_1) \otimes (d_2, b_2) = (d_1 + d_2, \min(b_1, b_2)). \tag{4.6}$$

Consider the superficially similar problem of shortest widest paths. That is, among the largest bandwidth paths, if there is more than one, we wish to select the shortest. This problem is not solvable with a semiring, at least, not with a simple modification of the previous one, i.e., with addition being

$$(d_1, b_1) \oplus (d_2, b_2) = \begin{cases} (d_1, b_1) & \text{if } b_1 > b_2 \text{ or } b_1 = b_2 \text{ and } d_1 < d_2, \\ (d_2, b_2) & \text{otherwise,} \end{cases} \tag{4.7}$$

and multiplication being the same as before. These two operators do not form a semiring.

In general, one can define more than two criteria per link (delay, bandwidth, loss, jitter, monetary cost, etc.) and define a lexicographic preference among them. But the existence of a semiring is not guaranteed in the general case.

k-shortest paths We want to find the k shortest paths (not necessarily elementary, i.e., repetitions of edges and vertices are allowed) between all pairs of vertices. Edges have weights from a set W as in the usual shortest path problem.

The carrier set is the set of k-tuples of W, $S = W^k$, and the operations are

$$(a_1, \ldots, a_k) \oplus (b_1, \ldots, b_k) = \text{min-k}\{a_1, \ldots, a_k, b_1, \ldots, b_k\}, \tag{4.8}$$

i.e., the k smallest elements in either one of the two vectors, and

$$(a_1, \ldots, a_k) \otimes (b_1, \ldots, b_k) = \text{min-k}\{a_i + b_j | i, j = 1, \ldots, k\}. \tag{4.9}$$

4.6 BGP ROUTING

The Internet is split into Autonomous Systems (ASs). An AS is a group of one or more networks run by one or more network operators which has a single and clearly defined routing policy. The routing policy controls the exchange of routing information between ASs, that is, what network destinations will be announced ("exported") to other ASs, and what will be accepted ("imported") from other ASs. The Border Gateway Protocol (BGP) [42] provides mechanisms for defining and executing such routing policies.

The commercial relationships among ASs define their relative roles. An AS A can be a *customer* of an AS B. In that case, B is a *provider* for A, and A pays B for access to the Internet. Alternatively, A and B are *peers*, that is, they exchange traffic between their customers without charging.

In general, an AS will import more than one route per destination. Then, it needs to decide how to choose among them. Note that, when discussing the shortest path problem, we assumed that the shortest path would always be chosen. This is what happens inside an AS (intra-domain routing), but there are different guidelines for routing among ASs (inter-domain routing).

Routing policy rules and guidelines have been proposed [20] that we want the paths to adhere to:

- An AS does not export to a provider or peer any routes that it learned from other providers and other peers.
- An AS can export to its customers any route it knows of.
- Routes learned from customers should be preferred to routes learned from either providers or peers.
- Preferences among customer routes, as well as among peer and provider routes, are left unspecified.

The situation described above is modeled as an edge-weighted directed graph where each node represents an AS. The set of edge weights contains the five elements $\{c, p, r, ①, ⓪\}$ (see Fig. 4.2): Weight c signifies a provider-to-customer link, p signifies a customer-to-provider link, and r signifies a peer-to-peer link. The weight ① is given to trivial paths comprising of a single node, whereas the weight ⓪ is given to forbidden paths, i.e., to paths that cannot be exported.

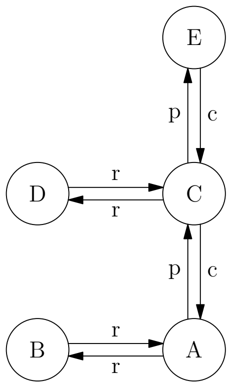

Figure 4.2: BGP example network. Weights c, p, r indicate, respectively, provider-to-customer links, customer-to-provider links, and peer-to-peer links between autonomous systems. For instance, AS C is a provider of AS A, a customer of AS E, and a peer of AS D.

The two operators are defined (4.1) so as to be faithful to the rules and guidelines stated. The preferences among paths are encoded in the canonical order induced by the operators. Roughly speaking, $\otimes$ encodes which paths are allowed, and $\oplus$ encodes the preferences among the allowed paths.

$\oplus$	c	r	p	①	⓪
c	c	c	c	①	c
r	c	r	r	①	r
p	c	r	p	①	p
①	①	①	①	①	①
⓪	c	r	p	①	⓪

$\otimes$	c	r	p	①	⓪
c	c	⓪	⓪	c	⓪
r	r	⓪	⓪	r	⓪
p	p	p	p	p	⓪
①	c	r	p	①	⓪
⓪	⓪	⓪	⓪	⓪	⓪

For example, $r \otimes p =$ ⓪ encodes the rule that an AS (e.g., A in Fig. 4.2) does not export to a peer a route that it learnt from a provider: To a valid (non-⓪) route learnt from a provider, the customer AS A prepends weight p (customer-to-provider link weight). So, the weight of that route will become p. If this new route is further sent to a peer AS of A (e.g., B), it would have to be prepended with weight r (peer-to-peer link). To encode that this is not allowed, the result of $r \otimes p$ is ⓪.

The canonical order is defined as

$$a \preceq b \Leftrightarrow a \oplus b = b. \tag{4.10}$$

To encode the preference for routes learned from customers rather than from peers or providers, Table 4.1 indicates that $c \oplus r = c \oplus p = c$, i.e., $c \succeq r$ and $c \succeq p$. And since $c \neq r, p$, we have that $c \succ r$ and $c \succ p$.

Note that in Table 4.1, we have set $r \oplus p = p \oplus r = r$, i.e., we have specified a preference between provider routes and peer routes in favor of the latter.

This modeling of BGP, with a slightly different notation, was given by Sobrinho [47].

4.7 SHORTEST PATH WITH TIME-INHOMOGENEOUS EDGES

Consider a communication network of mobile nodes. The nodes generate data packets and cooperatively forward their own and other nodes' packets to the appropriate destinations. Contemporaneous paths from a source to a destination do not necessarily exist. In consequence, forwarding takes place only when two nodes come into communication range of each other. This kind of network has been called *opportunistic network* and has similarities with Delay Tolerant Networks [25].

The shortest path problem for time-inhomogeneous edges is defined as follows: For a given source node s, destination node d, and time t_0 for the packet departure from the source, we want to compute the earliest possible time of arrival at node d. To solve this problem an endomorphism-based semiring (Section 2.2.4) can be used [22].

We start with the shortest path semiring $S = \mathbb{R} \cup \{\infty\}, \oplus = \min, \otimes = +$, but remember that we do not need the operator $\otimes$.

With each pair of nodes (i, j) we associate a non-decreasing function $w_{ij} : \mathbb{R} \cup \{\infty\} \to \mathbb{R} \cup \{\infty\}$, such that $\lim_{t\to\infty} w_{ij}(t) = \infty$. The interpretation of w_{ij} is that a packet sent by node i at time $t \in \mathbb{R}$ having node j as next hop will reach node j at time $w_{ij}(t)$. We emphasize that $w_{ij}(t)$ is *not* the delay incurred by a packet transmitted at time t: $w_{ij}(t)$ is the *time of arrival* at j, so the delay is $w_{ij}(t) - t$. For instance, setting $w_{ij}(t) = c_{ij} + t$ takes us back to the classical shortest path problem where the weight (delay) of edge (i, j) is equal to c_{ij} for all times t.

Seen from the opportunistic network viewpoint, $w_{ij}(t)$ is the first time instant after time t that i comes within communication range of j. Of course, with $w_{ij}(t)$ one can also model the time varying delay of a *wired* network link.

The functions w_{ij} as defined above can be seen to be endomorphisms of S. Therefore, the semiring of the endomorphisms w_{ij} with addition $+ = \min$, and multiplication $\odot =$ function composition can be used to compute the shortest path for time-inhomogeneous edges.

4.8 NETWORK RELIABILITY

4.8.1 DETERMINISTIC FAILURES

Reachability The most classical network reliability problem is that of determining whether a vertex t is reachable from a vertex s. The appropriate semiring is the Boolean semiring: $S = \{0, 1\}$, $\oplus = \max$, $\otimes = \min$, where $\oplus$ and $\otimes$ can also be seen as logical-or ($\vee$) and logical-and ($\wedge$), respectively. Edges with weight 0 are the failed/non-existing edges, and edges with weight 1 are the functioning/active edges.

Bridge and cut vertex finding In an undirected graph, an edge is called a *bridge* if its removal increases the number of connected components of the graph. Similarly, a vertex is called a *cut vertex* if its removal increases the number of connected components of the graph. If we try to define bridges in terms of paths, we could propose the following definition: An edge e is a bridge if there exists a pair of nodes s, t such that e is on all $s - t$ paths. So, to find the bridges of a graph, we need to find for each pair s, t the intersection of the sets of all $s - t$ path edges.

The carrier set of the semiring is the powerset of edges $S = 2^E \cup \{⓪\}$, with $\oplus = \cap$ (the usual set intersection), and $\otimes = \cup$ (the usual set union). The weight of an edge $e \in E$ is the singleton set $\{e\}$. Then, the weight of a path can be seen to be the set of its edges. The element ⓪, neutral element of $\oplus$ and absorbing element of $\otimes$, is the weight assigned to non-existing edges: If $a \in S$, then we define $a \cap ⓪ = a$ and $a \cup ⓪ = ⓪$. The element ①, neutral element of $\otimes = \cup$, is the empty set: $① \equiv \emptyset$.

To see that this semiring finds the bridges recall the fundamental formulation of the algebraic path problem (2.3):

$$d_{st} = \bigoplus_{p \in P^{st}} w(p). \tag{4.11}$$

Since $\oplus$ is set intersection and the weight of a path is the set union of its edges, the set d_{st} consists of the edges that are on all paths from s to t. All these edges are bridges, since their removal disconnects s from t. Also, every bridge of the graph G must be a member of at least one set d_{st} for some nodes s, t.

For finding the cut vertices we use a similar semiring, only the carrier set is now the powerset of vertices: $S = 2^V \cup \{\mathbb{0}\}$, $\oplus = \cap$ (set intersection), $\otimes = \cup$ (set union). The weight of edge (i, j) is the set $\{i, j\}$.

4.8.2 PROBABILISTIC FAILURES

Edges in a graph G are subject to random failure, possibly dependent on the failure of other edges. We want to compute, for a given pair $s - t$, the probability that at least one $s - t$ path survives, i.e., none of its edges fail. This probability is called the 2-terminal reliability $R_{st}(G)$.

A formal monomial term w_e is associated with each edge e, corresponding to the probability of survival of e. The monomial can be a single variable x_e or a product of variables $x_{e_1}x_{e_2}\ldots x_{e_k}$ *distinct from each other*. These variables correspond to components that can fail independently of each other and cause the failure of edges. Note that one or more components might be shared among multiple edges, thus modeling dependent edge failures. The idea is that if we substitute numerical values p_{e_i} for the formal variables x_{e_i}, the value of each monomial w_e will be equal to the survival probability of edge e.

We want to construct a formal polynomial in the variables x_{e_i}, such that, when substituting numerical values p_{e_i}, the result is $R_{st}(G)$.

We start with monomial variables as defined above, and we define multiplication and addition:

$$w_1 \otimes' w_2 = w_1 w_2 \tag{4.12}$$
$$w_1 \oplus' w_2 = w_1 + w_2 - w_1 w_2. \tag{4.13}$$

We see that $\otimes'$ will in general result in products where one or more variables are raised to an exponent larger than 1. Also, $\oplus'$ will in general result in polynomials, rather than monomials. The former is undesirable since we want components common to multiple edges to appear only once. The latter is not a problem since we want $w_1 \oplus' w_2$ to correspond to the probability that at least one of w_1 or w_2 survives. However, because of the way multiplication is defined, distributivity does not hold:

$$w_1 \otimes' (w_2 \oplus' w_3) = w_1 w_2 + w_1 w_3 - w_1 w_2 w_3 \tag{4.14}$$
$$w_1 \otimes' w_2 \oplus' w_1 \otimes' w_3 = w_1 w_2 + w_1 w_3 - w_1^2 w_2 w_3. \tag{4.15}$$

We can rectify the situation by resorting to a reduction function (Section 2.2.3) that reduces all exponents to 1:

$$\text{If } w = \prod_{e \in E, i \in I} {x_{e_i}}^{n_{e_i}}, n_{e_i} \geq 1, \tag{4.16}$$

$$\text{then } f(w) = \prod_{e \in E, i \in I} x_{e_i}. \tag{4.17}$$

The function f can be seen to satisfy the properties of a reduction function.

We now redefine the semiring operators

$$w_1 \otimes w_2 = f(w_1 w_2) = \prod \{\text{all } x_e \text{ in } w_1 \text{ or } w_2\}, \tag{4.18}$$

and

$$w_1 \oplus w_2 = w_1 + w_2 - w_1 \otimes w_2. \tag{4.19}$$

The carrier set S of the semiring should be the set of all polynomials that can be formed by the two operators $\oplus$ and $\otimes$.

This semiring is a dioid, where the canonical order relation is defined as

$$w_1 \preceq w_2 \Leftrightarrow w_1 \oplus w_2 = w_2. \tag{4.20}$$

The order is consistent with the usual order of the real numbers, in the sense that $w_1 \preceq w_2 \Leftrightarrow w_1 \leq w_2$ for any polynomials $w_1, w_2 \in S$ and any numerical values of x_{e_i}.

Shier [46] discusses this semiring and its application to reliability calculations at greater length.

4.9 SHORTEST PATHS WITH GAINS/LOSSES ON THE EDGES

A certain product circulates on a directed graph $G = (V, E)$. There is a multiplier $\mu_{uv} > 0$ associated with each edge $(u, v) \in E$, with the meaning that for each unit of the product that enters edge (u, v), μ_{uv} units exit. Note that μ_{uv} can be less than one (loss), or greater than one (gain). Also associated with each edge is a cost $c_{uv} \in \mathbb{R}$ which is the transportation cost for each unit of the product that crosses edge (u, v).

The shortest path problem with gains (or losses) is, given two vertices $s, t \in V$, to determine the $s - t$ path of minimum cost per product unit that reaches the destination vertex t. Note that for each unit of the product that leaves the source vertex s and follows path p, $\mu(p) = \prod_{e \in p} \mu_e$ units will reach the destination, and the transportation cost $c(p)$ paid will depend on how many units of the product entered each edge of the path p. The optimal path is the one that minimizes the ratio $\frac{c(p)}{\mu(p)}$.

In applications, edges can correspond to the transformation of one object to another, for example through currency exchange or a manufacturing process. The multiplier μ quantifies the conversion coefficient (e.g., dollar to euro exchange rate), and c quantifies the transformation cost (e.g., bank commission per dollar exchanged).

Alternatively, edges can correspond to a quantitative increase or reduction of a commodity. The multiplier corresponds to the increase or decrease coefficient (leakage probability, evaporation percentage), and the cost is the per unit transportation cost.

One can construct a semiring for this problem by defining addition as

$$(c_1, \mu_1) \oplus (c_2, \mu_2) = \begin{cases} (c_1, \mu_1) & \text{if } \frac{c_1}{\mu_1} < \frac{c_2}{\mu_2}, \text{ or if } \frac{c_1}{\mu_1} = \frac{c_2}{\mu_2} \text{ and } \mu_1 > \mu_2 \\ (c_2, \mu_2) & \text{if } \frac{c_1}{\mu_1} > \frac{c_2}{\mu_2}, \text{ or if } \frac{c_1}{\mu_1} = \frac{c_2}{\mu_2} \text{ and } \mu_1 < \mu_2 \end{cases}, \tag{4.21}$$

and multiplication as

$$(c_1, \mu_1) \otimes (c_2, \mu_2) = (c_1 + \mu_1 c_2, \mu_1 \mu_2). \tag{4.22}$$

Note that this semiring is only right-distributive. But think back to the iterative algorithms: They do not require both left and right distributivity, only one is necessary.

Similar problems have appeared under the name “generalized network flow” [39], or “generalized minimum cost maximum flow” [18].

4.10 TRUST-REPUTATION

We define trust as a set of weighted pairwise relations among entities in a network. The relations are established on the basis of a body of supporting assurance (trust) evidence. A *trust metric* is a function applied to that evidence that produces the weights for each relation. Examples of relations are: friends in a social network, buyers/sellers in an online marketplace, peers in a peer-to-peer network, and issuers of public key certificates in a PGP [51] web of trust.

We view the trust computation problem as a generalized shortest path problem on a weighted directed graph $G(V, E)$ (*trust graph*). The vertices of the graph are the users/entities in the network. A weighted edge from vertex i to vertex j corresponds to the *opinion* that entity i, also referred to as the *issuer*, has about entity j, also referred to as the *target*. The weight function is $w(i, j) : V \times V \longrightarrow S$, where S is the opinion space. The objective of the problem is, given a source vertex s and a target vertex t, to aggregate the opinions on the $s - t$ paths, in order to compute an indirect (transitive) opinion that s should have about t.

The *reputation* of an entity is often used in a similar manner to trust; the difference is that the reputation value depends only on one entity (the target, in the terminology above). That is, all sources should end up with the same value for the same destination.

We present two trust metrics, a model of PGP’s web of trust computations, a reputation algorithm, and a Semantic Web trust management algorithm.

In both of the following two metrics, each opinion consists of two numbers: the *trust* value, and the *confidence* value. The former corresponds to the issuer’s estimate of the target’s trustworthiness. For example, a high trust value may mean that the target is one of the good guys, or that the target is able to give high quality location information, or that a digital certificate issued for the target’s public key is believed to be correct. On the other hand, the confidence value corresponds to the accuracy of the trust value assignment. A high confidence value means that the target has taken a large number of tests that the issuer has set, or that the issuer has interacted with the target for a long time. Since opinions with a high confidence value are more useful in making trust decisions, the confidence value is also referred to as the *quality* of the opinion.

4.10.1 PATH SEMIRING

In the first semiring, the opinion space is $S = \{(t, c) \in [0, 1] \times [0, 1]\}$. Our choice for the $\otimes$ and $\oplus$ operators is

$$(t_1, c_1) \otimes (t_2, c_2) = (t_1 t_2, c_1 c_2) \tag{4.23}$$

$$(t_1, c_1) \oplus (t_2, c_2) = \begin{cases} (t_1, c_1) & \text{if } c_1 > c_2 \\ (t_2, c_2) & \text{if } c_1 < c_2 \\ (\max\{t_1, t_2\}, c_1) & \text{if } c_1 = c_2 \end{cases} . \tag{4.24}$$

Since both the trust and the confidence values are in the [0, 1] interval, they both decrease when aggregated along a path. When opinions are aggregated across paths, the one with the highest confidence prevails. If the two opinions have equal confidences but different trust values, we pick the one with the highest trust value. We could have also picked the lowest trust value; the choice depends on the desired semantics of the application.

This semiring essentially computes the trust distance along the most confident trust path to the destination. An important feature is that this distance is computed along a single path, since the $\oplus$ operator picks exactly one path. Other paths are ignored, so not all available information is being taken into account. One of the advantages is that if the trust value turns out to be high, then a trusted path to the destination has also been discovered. Also, fewer messages are exchanged for information gathering.

4.10.2 DISTANCE SEMIRING

The distance semiring is based on the *Expectation semiring* defined by Eisner (Section 4.2):

$$(a_1, b_1) \otimes (a_2, b_2) = (a_1 b_2 + a_2 b_1, b_1 b_2) \tag{4.25}$$

$$(a_1, b_1) \oplus (a_2, b_2) = (a_1 + a_2, b_1 + b_2). \tag{4.26}$$

The opinion space is $S = [0, \infty] \times [0, 1]$. Before using this semiring, the pair (trust, confidence)=(t, c) is mapped to the weight $(c/t, c)$. The motivation for this mapping becomes clear when we describe its effect on the results of the operators. The binary operators are then applied to this weight, and the result is mapped back to a (trust, confidence) pair. For simplicity, we only show the final result without the intermediate mappings.

$$(t_1, c_1) \otimes (t_2, c_2) \to \left(\frac{1}{\frac{1}{t_1} + \frac{1}{t_2}}, c_1 c_2 \right) \tag{4.27}$$

$$(t_1, c_1) \oplus (t_2, c_2) \to \left(\frac{c_1 + c_2}{\frac{c_1}{t_1} + \frac{c_2}{t_2}}, c_1 + c_2 \right). \tag{4.28}$$

So, when aggregating along a path, both the trust and the confidence decrease. The component trust values are combined like parallel resistors. Recall that two resistors in parallel offer lower resistance than either of them in isolation. Also, a zero trust value in either opinion will result in a zero trust value in the resulting opinion (absorbing element), while a trust value equal to infinity will cause the corresponding opinion to disappear from the result (neutral element). On the other hand, the component confidence values are between 0 and 1, and they are multiplied, so the resulting confidence value is smaller than both.

When aggregating across paths, the total trust value is the weighted harmonic average of the components, with weights according to their confidence values. So, the result is between the two component values, but closer to the more confident one. Note, also, the behavior caused by extreme (zero or infinity) trust values: A zero trust value dominates the result (unless its corresponding confidence is zero); a trust value equal to infinity results in an increase in the trust value given by the other opinion. In order for the resulting trust value to be the maximum possible, both opinions have to assign the maximum. So, in general, we can say that this operator is conservative. A zero confidence value (neutral element) causes the corresponding opinion to disappear from the result.

4.10.3 PGP TRUST COMPUTATION MODEL

We now present a model of PGP's web of trust computations [51] as a semiring.

Briefly, a problem in public key cryptography is the reliable distribution of public keys: A sender wants to find the public key of a destination, but how can the sender be sure that a given key is the correct one? In PGP's solution, users sign digital certificates with their keys. A digital certificate can be seen as a statement from the signer that a certain public key belongs to a certain user. Having a certificate for the destination's key, the problem now becomes finding a certificate for the signer's public key. Continuing recursively, at least one path of certificates needs to be constructed that starts at the sender and ends at the destination.

The resulting graph of certificates issued by users for each other is called the *web of trust*. The web of trust concept is used to establish the validity of a binding between a public key and a user, as seen from the point of view of a particular user (the *source*). The input to the validity computation algorithm consists of three things: the source node, the web of trust, and the trust values for each user as assigned by the source. These user trust values are *manually* assigned by the source, not computed by the algorithm. They correspond to how much the source trusts certificates signed by these nodes.

Note that the validity of all key-to-user bindings has to be verified since only certificates signed by valid keys are taken into account, and any certificate may influence the validity of a key-to-user binding.

The validity of the key-to-user binding for user i will be deduced from the vector $d_i \in \mathbb{N}^k$, where k is the number of different trust levels defined by PGP. It seems that k is 4 ("unknown", "untrusted", "marginally trusted", "fully trusted"), but some include a fifth level: "ultimately trusted". Our analysis is independent of the exact value of k. The vector d_i will hold the number of valid certificates for user i that have been signed by users of each trust level. For example, $d_i = (0, 1, 2, 3)$ means that one "untrusted", two "marginally trusted", and three "fully trusted" users have issued certificates for user i's public key. In addition, all six of these certificates are signed by valid keys, i.e., keys for which the key-to-user binding has been verified.

In order to verify the actual validity of the binding, we will use the function $\texttt{valid} : N^k \to \mathbb{V}$, where $\mathbb{V}$ is the space of admissible results. For simplicity, we will be assuming that $\mathbb{V}$ ={"invalid", "valid"}, although values such as "marginally valid" have also been proposed. The output of `valid` for a specific input is determined by thresholds such as: "A key-to-user binding is valid if at least two "marginally trusted" users have issued a certificate for it". These thresholds are incorporated in `valid` and will be transparent for the analysis that follows. Finally, to simplify the presentation, we will be assuming that $\mathbb{V} = \{0, 1\}$, where "invalid"= 0, and "valid"= 1.

The edge weights $w_{ij} \in \mathbb{N}^k$, $1 \le i, j \le n$, where n is the number of users, characterize the certificate issued by i about j's alleged public key. A weight can only have one of $k + 1$ possible values. Either it consists only of 0s, or of exactly $k - 1$ 0s and one 1. An all-zero weight means that there is no certificate from i about j's key. An 1 in the position that corresponds to trust level k' means that the source has assigned trust level k' to i, and i has issued a certificate for j.

The $\otimes$ operator is defined as follows ($a, b \in \mathbb{N}^k$):

$$a \otimes b = \texttt{valid}(a)b \in \mathbb{N}^k. \tag{4.29}$$

The $\oplus$ operator is defined exactly as vector addition in $\mathbb{N}^k$.

The three semirings above were proposed in [48].

4.10.4 EIGENTRUST

The EigenTrust algorithm was proposed [30] in the context of peer-to-peer file sharing networks as a way to combat malicious peers that provide inauthentic files to the network. A high value for a peer corresponds to a high expectation that a file coming from that peer will be authentic.

The weights in this case are real numbers between 0 and 1: $S = [0, 1]$. The weights are normalized per user, i.e., for each user i, $\sum_{j \in N(i)} w(i, j) = 1$.

Addition is defined as

$$t_1 \oplus t_2 = t_1 + t_2, \tag{4.30}$$

i.e., usual addition.

Multiplication is defined as

$$t_1 \otimes t_2 = t_1 t_2, \tag{4.31}$$

i.e., usual multiplication.

4.10.5 SEMANTIC WEB SEMIRING

The philosophy behind the Semantic Web is to structure and annotate the content of web pages so that computers become capable of analyzing it. The relevance of trust for the semantic web is that the created content has widely varying degrees of quality. Even more, different consumers of the same content will have varying opinions on its quality.

A centralized evaluation of all the content is out of the question, both due to the sheer quantity of data, and also due to the lack of a centralized authority trusted by all to perform this task. Consequently, semirings suggest themselves as a framework within which to calculate trust values in a distributed way.

The weights in this case [43] are real numbers between 0 and 1: $S = [0, 1]$.

Addition was proposed to be maximum, minimum, or average. Since average is not associative, it was proposed to calculate it using a separate semiring calculation based on *sum* and *count*.

Multiplication is defined as

$$t_1 \otimes t_2 = t_1 t_2, \tag{4.32}$$

i.e., usual multiplication.

4.11 SOCIAL NETWORKS

In social network theory [4], graphs are used to represent communities. Vertices represent members of the community, and (possibly weighted) edges encode the relations among the members. In the first part of this section (4.11.1 and 4.11.2 below) we will use the concept of a signed graph:

Definition 4.1 A *signed* graph is an edge-weighted directed graph, where the set of weights is $\{-1, 1\}$. The sign of a path is defined to be the product of the weights of its edges.

A positive (negative) edge (i, j) signifies that i likes (dislikes) j.

Definition 4.2 A signed graph is called *partitionable* (or *clusterable*) if its vertices can be partitioned into disjoint subsets (clusters) so that only positive edges exist between vertices in the same subset, and only negative edges exist between vertices of different subsets.

In other words, a partitionable graph models a community where people can be separated into groups of friends, without putting two persons that dislike each other in the same group, or two friends in separate groups. A special case is when we are able to separate the people into only two such groups.

Definition 4.3 A partitionable graph is called *balanced* if the number of its clusters is equal to 2.

4.11.1 BALANCE SEMIRING

We want to construct a semiring for identifying balanced graphs. To this end, we make use of the following theorem to express the property of balance in terms of paths in the graph:

Theorem 4.4 *A signed graph is balanced if and only if every cycle (ignoring, however, the direction of the edges) has weight equal to 1.*

The carrier set of the semiring is $S = \{0, -1, 1, a\}$. The weight of non-existing edges/paths is 0, while 1 and -1 are defined (for edges) and then computed (for paths) to comply with Definition 4.1 above. The element a signifies that both negative and positive paths exist between two given nodes.

The two operations are shown in the following table:

$\oplus$	0	-1	1	a
0	0	-1	1	a
-1	-1	-1	a	a
1	1	a	1	a
a	a	a	a	a

$\otimes$	0	-1	1	a
0	0	0	0	0
-1	0	1	-1	a
1	0	-1	1	a
a	0	a	a	a

Theorem 4.5 *A signed graph is balanced if and only if the diagonal of A^* contains only 1s.*

4.11.2 CLUSTER SEMIRING

We want to construct a semiring for identifying partitionable graphs. We make use of the following theorem:

Theorem 4.6 *A signed graph is partitionable if and only if it contains no cycle (ignoring the direction of the edges) with exactly one negative edge.*

The carrier set of the semiring is defined as $S = \{0, -1, 1, a, b\}$. The weight 0 corresponds again to non-existing edges/paths. The meaning of the rest, however, changes:

-1	At least one path with exactly one negative arc; no path with only positive arcs.
1	At least one path with only positive arcs; no path with exactly one negative arc.
a	At least one path with only positive arcs; at least one path with exactly one negative arc.
b	Each path has at least two negative arcs.

The two operations are shown in the following table:

$\oplus$	0	-1	1	a	b
0	0	-1	1	a	b
-1	-1	-1	a	a	-1
1	1	a	1	a	1
a	a	a	a	a	a
b	b	-1	1	a	b

$\otimes$	0	-1	1	a	b
0	0	0	0	0	0
-1	0	b	-1	-1	b
1	0	-1	1	a	b
a	0	-1	a	a	b
b	0	b	b	b	b

Theorem 4.7 *A signed graph is partitionable if and only if the diagonal of A^* contains only 1s.*

4.11.3 GEODETIC SEMIRING

We now switch gears. In this and the next subsection, we study quantities that relate to the proximity of two community members. We model the community as a directed but not weighted graph.

In this subsection, we present the geodetic semiring, which helps compute the length and number of shortest paths between each pair of vertices. Since the graph is not weighted, the length of a path is equal to the number of its edges.

The carrier set of the semiring is: $S = ([0, \infty) \cup \{\infty\}) \times (\mathbb{N} \cup \{\infty\})$, where the first entry (denoted d below) corresponds to the length of the shortest paths between two given nodes, and the second entry (denoted n below) corresponds to the number of shortest paths between these two nodes.

Addition is defined as

$$(d_1, n_1) \oplus (d_2, n_2) = \begin{cases} (d_1, n_1) & \text{if } d_1 < d_2 \\ (d_2, n_2) & \text{if } d_1 > d_2 \\ (d_1, n_1 + n_2) & \text{if } d_1 = d_2 \end{cases} \tag{4.33}$$

and multiplication as

$$(d_1, n_1) \otimes (d_2, n_2) = (d_1 + d_2, n_1 n_2). \tag{4.34}$$

The ⓪ and ① of the semiring are

$$⓪ = \{(\infty, 0)\} \tag{4.35}$$
$$① = \{(0, 1)\}. \tag{4.36}$$

4.11.4 GEOSETIC SEMIRING

In this subsection, we want to compute, for each pair of vertices s, t, the set of vertices of the shortest paths from s to t. Again, the graph is directed but not weighted.

The elements of the carrier set are the finite sets of quadruples of the form

$$(i, P, j, d), i, j \in V, P \subseteq V, d \in \mathbb{N}. \tag{4.37}$$

Each quadruple encodes an initial vertex i, a terminal vertex j, the set P of vertices on the shortest $i - j$ paths (including i and j), and the length d of these paths. The length d can only be zero in the case of a trivial one node path. Additionally, in any member (set of quadruples) of the carrier set there can be at most one quadruple with a given pair (i, j) of initial-terminal vertices.

Addition is defined as follows:

$$s_1 \oplus s_2 = \text{Reduction}(s_1 \cup s_2) \tag{4.38}$$

where $\text{Reduction}(s)$ is a reduction function (see also Sec. 2.2.3) that merges quadruples with identical (i, j) by repeated application of the following merge rule:

$$\text{merge}((i, P_1, j, d_1), (i, P_2, j, d_2)) = \begin{cases} (i, P_1, j, d_1) & \text{if } d_1 < d_2 \\ (i, P_2, j, d_2) & \text{if } d_1 > d_2 \\ (i, P_1 \cup P_2, j, d_1) & \text{if } d_1 = d_2 \end{cases}. \tag{4.39}$$

In other words, $\text{Reduction}(s_1 \cup s_2)$ is a set containing one quadruple for each distinct (i, j) pair in $s_1 \cup s_2$. For a given pair (i, j), this quadruple contains the length and the set of vertices of the shortest $i - j$ paths. The $⓪$ of the semiring is

$$⓪ = \{(*, \emptyset, *, \infty)\}, \tag{4.40}$$

where $*$ stands for "any vertex."

Before defining multiplication between sets of quadruples, we define it between singletons, i.e., between exactly two quadruples.

$$(i_1, P_1, j_1, d_1) \odot (i_2, P_2, j_2, d_2) = \begin{cases} (i_1, P_1, j_1, d_1) & \text{if } d_2 = 0,\ j_1 = i_2 \\ (i_2, P_2, j_2, d_2) & \text{if } d_1 = 0,\ j_1 = i_2 \\ (i_1, P_1 \cup P_2, j_2, d_1 + d_2) & \text{if } i_1, j_1, \text{and } j_2 \text{ are all distinct,} \\ & \quad \text{and } P_1 \cap P_2 = \{j_1\},\ j_1 = i_2 \\ & \text{or if } j_1 \neq i_1 = j_2, \\ & \quad \text{and } P_1 \cap P_2 = \{i_1, j_1\},\ j_1 = i_2 \\ ⓪ & \text{otherwise.} \end{cases} \tag{4.41}$$

The intuition goes as follows: First of all, if the two quadruples are not consecutive, i.e., the terminal vertex of the first does not coincide with the initial vertex of the second, then the result is ⓪. If they are consecutive and either one is the trivial quadruple with length 0 (*not* the ⓪ quadruple), then the result is equal to the other quadruple. The remaining case, which is the general case of consecutive non-trivial quadruples, can be dealt with by way of graph theoretical arguments. An important point for this latter case is that if the intersection of P_1 and P_2 contains more vertices

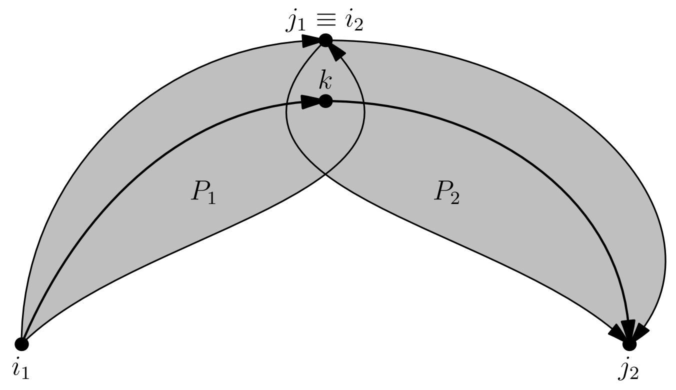

Figure 4.3: Multiplication in the geosetic semiring. If $\exists k \in P_1 \cap P_2$, such that $k \neq j_1 \equiv i_2$, then $i_1 - j_2$ paths that go through j_1 cannot be shortest.

than the endpoints of the quadruples, then the $i_1 - j_2$ paths that go through $j_1 \equiv i_2$ cannot be shortest paths (see Fig. 4.3). Consequently, the result of the multiplication would again be $⓪$.

Now, we use multiplication for singletons to define multiplication for multi-element sets:

$$s_1 \otimes s_2 = \bigoplus_{w_1 \in s_1, w_2 \in s_2} w_1 \odot w_2. \tag{4.42}$$

The $①$ of the semiring is the set

$$① = \{(i, \{i\}, i, 0) : i \in V\}. \tag{4.43}$$

4.12 TRAFFIC ASSIGNMENT

A road network is modeled as an edge-weighted directed graph. For a set of origin-destination node pairs $s - t$ there exist traffic demands t_{st} that model the traffic that should go from node s to node t. An edge weight models the travel time (delay) spent on the corresponding road segment. The delay of a road segment may depend or not on the traffic of that road segment. Note that there are no capacities on the edges; we assume that an edge can carry any amount of traffic. Also, the delay may be deterministic or stochastic.

The objective of the *traffic assignment* problem is to find the amount of traffic that will go through each edge of the network [34], assuming that each traffic demand t_{st} is split among the possible $s - t$ paths so as to minimize its $s - t$ expected travel time. The delay of a path $d(p)$ is equal to the sum of its edge delays.

When the edge delays are deterministic and do not depend on the traffic that flows through them, then the traffic assignment problem reduces to finding shortest paths from s to t for all source-destination pairs s, t. After finding the shortest paths, we can easily compute the total traffic through each edge.

When the edge delays are deterministic but depend on the traffic, the solution concept that one looks for is called a *Wardrop equilibrium*: At a Wardrop equilibrium, for every pair s, t, the delay of all the $s - t$ paths used by the $s - t$ flow is the same, and it is smaller than that of any other $s - t$ path. The Wardrop equilibrium can be computed iteratively.

We deal with the case of stochastic path delays that do not depend on the traffic. We assume that a user chooses a complete origin-destination path, as opposed to making individual decisions at all crossroads. The user's choice is probabilistic among all possible paths. The probability that a given path is chosen is the probability that this path is the shortest.

The standard way of choosing paths is the *Logit Assignment*: A given path p is chosen for the traffic demand t_{st} with probability

$$\Pr\{\text{Path } p \text{ is chosen from } s \text{ to } t\} = \frac{e^{-\mu d_p}}{\sum_{p \in P^{st}} e^{-\mu d_p}}, \tag{4.44}$$

where $\mu > 0$ is a parameter that tunes how much a lower delay path is favored over a higher delay path. When $\mu \longrightarrow \infty$, the lowest delay path is chosen with probability 1, but when $\mu \longrightarrow 0$, all paths are chosen with the same probability. The fraction of t_{st} that goes through a given $s - t$ path is given by the probability that the Logit Assignment dictates.

A brief justification for this assignment is as follows: We assume that the delay $d(p)$ of a path p is equal to a constant d_p plus a zero mean Gumbel distributed random variable E_p:

$$d(p) = d_p + E_p, \Pr\{E_p < x\} = e^{-e^{-\mu x - \gamma}}, \tag{4.45}$$

where $\gamma = 0.577\ldots$ is Euler's constant. The Gumbel random variables of different paths are assumed to be independent and to have the same variance, which is equal to $\frac{\pi^2}{6\mu^2}$. Then, the probability of path p having the minimum delay is given by (4.44).

We now create a semiring to compute the traffic assignment given the road network and the traffic demands. The elements of the semiring are the non-negative real numbers. The semiring operations are defined as follows:

$$a \oplus b = -\frac{1}{\mu} \log\left(e^{-\mu a} + e^{-\mu b}\right) \tag{4.46}$$

$$a \otimes b = a + b. \tag{4.47}$$

We create the weighted adjacency matrix $A = [a_{ij}]$, where a_{ij} is equal to average delay of edge (i, j). We can then compute the closure matrix A^*. Denoting the elements of A by $w(i, j)$ and the elements of A^* by $d(i, j)$, the fraction of $s - t$ traffic that goes through edge (i, j) is given by [5]

$$p_{stij} = \frac{e^{-\mu d(s,i)} e^{-\mu w(i,j)} e^{-\mu d(j,t)}}{e^{-\mu d(s,t)}} \tag{4.48}$$

and so to compute the total *amount* of traffic on edge (i, j) we have to sum over all traffic demands, weighted by the probability that they use edge (i, j):

$$t_{ij} = \sum_{st} p_{stij} t_{st}. \tag{4.49}$$

Since the semiring is not a dioid, we have issues of convergence of the closure computation. A sufficient condition for convergence is that the matrix $W = [w_{ij}] = [e^{-\mu a_{ij}}]$ is such that $\lim_{n\to\infty} W^n = 0$. This can always be guaranteed for large enough μ.

4.13 APPLICATIONS OF SENSITIVITY ANALYSIS

The practical networking significance of computing upper and lower edge tolerances is varied. From a security point of view, an attacker may want to cause the maximum performance degradation by destroying a single link. Then, he would be interested to know the link tolerances since they provide a measure for how difficult it is to replace a link (cost of alternative paths). It follows that, from a defender's viewpoint, the same edge should be protected the most. Further, if a network administrator wants to achieve the largest performance improvement by improving (or adding) one link, then the upper tolerances would be useful: they give insight to the effect that link improvements have.

Computing tolerances is also relevant for judging the stability of routing paths. If a link weight is close to its upper or lower tolerance, then it is more likely that a random fluctuation in the link weight will cause the corresponding optimal path to change. This is undesirable, and a network administrator or designer would benefit from knowing it could happen.

Seen from an economic viewpoint, the value of an edge for a source-destination pair can be measured by the effect its disappearance has on the weight of the optimal path between the pair. For instance, if the disappearance of a link partitions the network and leaves the source and destination unable to communicate, then the value of that link is theoretically infinite. This value has a game theoretic interpretation as the *Vickrey* price of that edge. It is relevant for determining how much compensation should go to an agent who controls that edge. The same reasoning could be used for evaluating, e.g., how much a certain Service Level Agreement between ISPs is worth. Related work is [27], and originally [38].

The relevance of tolerances also extends to traffic redirection attacks, discussed in subsection 4.13.1, which in wireless ad-hoc networks have been called wormhole attacks [29]. An attacker may try to attract traffic to the links controlled by him. The lower tolerances of edges are the thresholds below which traffic will be redirected to these newly "improved" edges. Alternatively, the attacker may break certain links so as to force traffic to pass through compromised network elements (links and routers).

4.13.1 TRAFFIC REDIRECTION ATTACKS

In this section, we will see how knowing edge tolerances helps understand the effects of traffic redirection attacks on routing. Traffic redirection attacks aim to cause traffic to go through links or nodes of the attacker's choosing by making these links appear shorter than the other candidate links. Their effect can be bad for various reasons:

- The redirected traffic could cause congestion to the part of the network it is redirected to.
- The redirected traffic will experience worse than optimal conditions (for instance, longer delay), assuming, of course, that before the attack it was being routed optimally.
- The attacker could eavesdrop on the traffic, modify it, or drop it.

There are two general mechanisms that can be used to achieve traffic redirection: traffic attraction and traffic repulsion. Traffic attraction occurs when a link is advertised to have much lower weight than it really has, thus luring traffic to go through it. Traffic repulsion occurs when the advertised weight of the link is much *larger* than in reality, thus causing traffic to avoid it.

Traffic attraction attacks have been known for a long time in the networking field [7], [40]. In wireless networks, the false advertisement of non-existent routes has been called the wormhole attack [29]. The effect is a traffic attraction attack.

The relevance of tolerances for traffic attraction attacks is the following: If the weight of a link is changed (advertised) to a value better than its upper tolerance, then the associated end-to-end traffic will be redirected to a path that crosses the changed link. Advertising non-existent links can be handled by finding upper tolerances for edges with initial weight ⓪. Depicted in Figure 4.4 are the initial weights and upper tolerances of a simple topology. We can see that not all links can be used for an attraction attack. Only links that are "close enough" to the currently optimal end-to-end path in question are candidates.

In the case of traffic repulsion attacks lower tolerances are relevant (Figure 4.5). They show how much a link's weight needs to deteriorate before traffic is redirected. The new path can also be computed, which will show which new links and routers the new path goes through.

If we know the tolerances for all the edges in the network, we can see which edges, if artificially improved, destroyed, or degraded, will cause traffic to be redirected through which path. So, we can see the effects of the possible single-link attacks, and plan accordingly. For example, one could check the trustworthiness of routers that could be used as replacements, or guard certain parts of a path whose replacements are judged undesirable for one reason or another.

Algorithms for upper and lower tolerances for shortest paths and for maximum capacity paths were given in [41] by reducing these problems to the same problem, called Minimum Cost Interval problem. In [27], an algorithm was given that computes how much the length of the shortest path changes when an arbitrary link disappears. This is equivalent to computing lower tolerances.

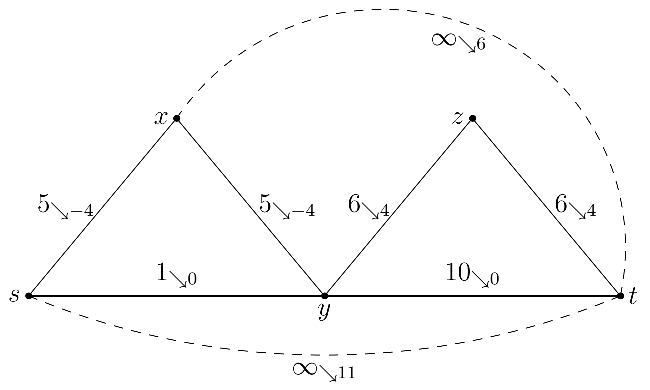

Figure 4.4: Traffic attraction attack: By artificially improving the weight of a link beyond its upper tolerance, the attacker can lure traffic away from the optimal path. Shown next to each link are the initial weights and upper tolerances (remember that the upper tolerance of an edge is the tolerance in the direction of improvement, i.e., weight reduction in the (min, +) semiring). Upper tolerances of two non-existent edges are also shown. Note that the negative values of tolerances are only shown for completeness purposes: negative values for delays cannot be admitted. The optimal path is painted thick.

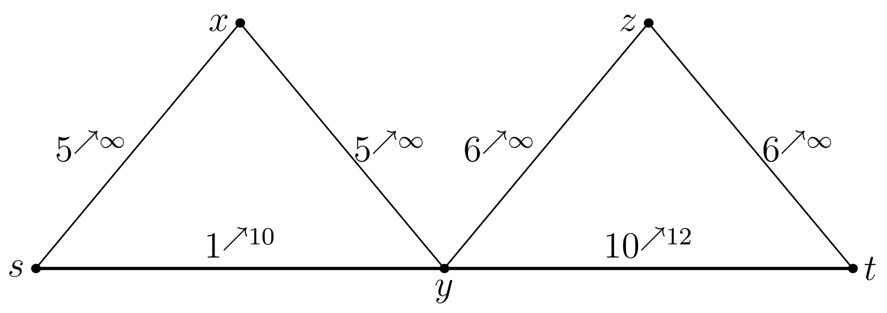

Figure 4.5: Traffic repulsion attack: By decreasing the weight of a link beyond its lower tolerance, the attacker can lure traffic away from the optimal path. Remember that the lower tolerance of an edge is the tolerance in the direction of deterioration, i.e., weight increase in the (min, +) semiring. Shown next to each link are the initial weights and lower tolerances. The optimal path is painted thick.

CHAPTER 5
Related Areas

5.1 NON-SEMIRING PATH PROBLEMS

Certain problems seem to resist a natural description in terms of semirings. A certain class of such problems have been called "non Markovian path problems" [45]. There, the contribution of an edge to the total path weight depends not only on the weight of that edge but also on the weights of the edges already traversed by the path. The consequence is that subpaths of optimal paths are not necessarily optimal. Algebraically, the distributivity property is violated.

As an example, consider the problem of finding shortest paths with discounting. In this problem, we are still looking for shortest paths, but now the cost of a path $p = (v_0, v_1, \ldots, v_k)$ is the discounted sum of its edge weights:

$$w(p) = \sum_{i=0}^{k-1} \delta^i w(v_i, v_{i+1}), 0 < \delta < 1. \tag{5.1}$$

An attempt to model this problem with a semiring would be to choose the carrier set to encode both the (discounted) length, and the number of edges of a path:

$$S = \mathbb{R} \times \mathbb{N}. \tag{5.2}$$

Then, $\oplus$ is lexicographic minimum, and $\otimes$ is

$$(c_1, n_1) \otimes (c_2, n_2) = (c_1 + c_2\delta^{n_1}, n_1 + n_2). \tag{5.3}$$

The weight of edge e is $(c_e, 1)$.

Figure 5.1 shows a case where distributivity does not hold.

Consider, in Figure 5.1, the weight resulting from appending an edge e at the end of the two parallel paths p_1, p_2. One can confirm that

$$(w(p_1) \oplus w(p_2))\, w(e) \neq w(p_1)w(e) \oplus w(p_2)w(e). \tag{5.4}$$

By taking, for example, $\delta = 0.5$, we see that

$$\begin{aligned}(w(p_1) \oplus w(p_2))\, w(e) &= ((3, 3) \oplus (1, 1))\,(8, 1) && (5.5)\\ &= (1, 1) \otimes (8, 1) && (5.6)\\ &= (5, 2), && (5.7)\end{aligned}$$

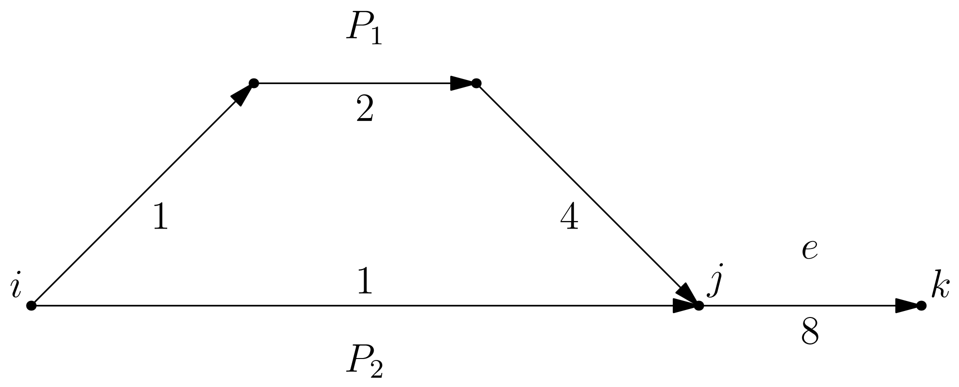

Figure 5.1: Distributivity does not hold in the problem of shortest paths with discounting.

whereas

$$
\begin{aligned}
w(p_1)w(e) \oplus w(p_2)w(e) &= ((3,3) \otimes (8,1)) \oplus ((1,1) \otimes (8,1)) && (5.8)\\
&= (4,4) \oplus (5,2) && (5.9)\\
&= (4,4). && (5.10)
\end{aligned}
$$

However, there are methods [32] for transforming certain non-semiring costs to semirings. In particular, the non-distributivity of $\otimes$ over $\oplus$ can be overcome by enlarging the carrier set and selecting an appropriate reduction function. Which reduction function is appropriate depends on the particular problem.

In the shortest path problem with discounting, the obstacle to distributivity comes from $\oplus$-adding the weights of paths with a different number of edges. So, we first enlarge the carrier set to have as elements the *multisets* of weights of the form (c, n). We define a reduction function that operates on a multiset M, returning a set with only one element for each value of n that appears in M: the element with the minimum cost c. Addition of two multisets is the reduction function applied to their union. Multiplication of two multisets is the reduction function applied to the multiset of pairwise products of weights, one from each multiset.

One can verify that the resulting algebraic structure is indeed a semiring. Applying this semiring to the problem instance in Fig. 5.1, the result obtained is the set $\{(4,4), (5,2)\}$. If we want to compute the optimal $i - k$ path, we can just compare lexicographically the two elements of this set, and conclude in favor of $(4,4)$.

The non-associativity of $\otimes$ can also be circumvented, but we will not go into the details here; we will only give a motivating example [32]. Non-associativity appears when edge/path weights do not provide enough information on how to multiply them. For example, in the shortest path problem with discounting, we may want to discount only every other edge:

$$w(p) = \sum_{i=0}^{k-1} \delta^{\left\lfloor \frac{i}{2} \right\rfloor} w(e_i), 0 < \delta < 1. \tag{5.11}$$

But the multiplication operator

$$(c_1, n_1) \otimes (c_2, n_2) = (c_1 + c_2 \delta^{\left\lfloor \frac{n_1}{2} \right\rfloor}, n_1 + n_2) \tag{5.12}$$

is not associative. Take $\delta = 0.5$ for example. Then

$$((1, 1) \otimes (1, 1)) \otimes (2, 1) = (3, 3) \neq (4, 3) = (1, 1) \otimes ((1, 1) \otimes (2, 1)) . \tag{5.13}$$

Intuitively, augmentation of the weights can provide the necessary context, by having a different weight for each context. In this case, we could define two contexts of multiplication, odd and even. Then, a different multiplication operator will be used in each context. The interested reader is referred to [32].

Still, finding a semiring for some path problems that can be straightforwardly described, e.g., find the k-shortest disjoint $s - t$ paths, seems to be an open problem.

5.2 SEMIRING NON-PATH PROBLEMS

5.2.1 THE MAX-PLUS SEMIRING REVISITED

A line of research parallel to the algebraic path problem has focused on max-plus algebra and its applications to performance evaluation of discrete event systems [3] [1]. The emphasis has been on developing a system theory that is linear in the semiring $(\mathbb{R} \cup \{-\infty\}, \max, +)$, hence the name max-plus. The fundamental notions in that research are generalizations of state-space equations, input-output functions, feedback loops, etc. Rather than computing the closure of a matrix, one is interested in computing eigenvalues and eigenvectors of matrices with entries in the max-plus semiring.

A linear system in the max-plus semiring was used to model the packet level dynamical behavior of two variants of TCP (Reno and Tahoe) over a single end-to-end connection [2]. The max-plus eigenvalue of an appropriately defined matrix was shown to be inversely proportional to the throughput of the connection. Various results in the TCP literature were confirmed or refined.

5.2.2 SEMIRINGS AS AN ALGEBRA FOR COMBINATORICS

The combinatorial nature of the max-plus semiring has inspired some reformulations of classical combinatorial problems in max-plus terms [12]. For instance, the assignment problem is equivalent to computing the permanent of a matrix (see (5.14)) in the max-plus semiring.

The classical example of the assignment problem is to optimally match n employees to n tasks. If employee i is assigned to task j, there is accrued value s_{ij}. The objective is to maximize the total value, after no employee and no task is left unmatched.

In graph theoretic terms, the assignment problem is to find a maximum weight perfect matching in a complete, bipartite, weighted graph $G = (L \cup R, E)$, where $|L| = |R|$. For a graph G, a set M of edges is a *matching* if no pair of edges of M has a common vertex. A matching is *perfect* when every vertex of the graph is incident to exactly one edge of the matching. For an edge weight function w, a matching M is a maximum weight matching when $w(M) = \sum_{e \in M} w(e)$ is maximum among all matchings.

Let $A_{n \times n} = (a_{ij})$ and denote by P_n the set of all permutations of the set $\{1, \ldots, n\}$. The *permanent* of the matrix A is

$$\sum_{\pi \in P_n} \prod_{i=1}^{n} a_{i,\pi(i)}. \tag{5.14}$$

Let A be the weighted adjacency matrix of graph G. We write the permanent in semiring notation as

$$\bigoplus_{\pi \in P_n} \bigotimes_{i=1}^{n} a_{i,\pi(i)}. \tag{5.15}$$

Using the max-plus semiring, the permanent computation becomes

$$\max_{\pi \in P_n} \sum_{i=1}^{n} a_{i,\pi(i)}. \tag{5.16}$$

So, the value of the max-plus permanent is equal to the value of the optimal assignment. The permutation that achieves the maximum is the optimal assignment.

5.2.3 NETWORK CALCULUS

Network calculus [11] is a theory and an associated set of results about deterministic queueing systems in networks. The mathematical basis of network calculus is the min-plus semiring, but this semiring is not used to find shortest paths between nodes. Rather, the convolution in the min-plus semiring is used to aggregate the effect of network elements (e.g., traffic shapers) on a flow traversing them along a path.

Here, we will only briefly give some basic definitions and the fundamental result of concatenation of network elements. A data flow is represented through a left-continuous nondecreasing function $R(t)$, defined as the number of bits seen on the flow in the time interval $[0, t]$. A network element (node) is defined through the service it provides to a flow that goes through it.

Suppose that a node provides a minimum rate guarantee $r \frac{\text{bits}}{\text{sec}}$, and that the node's buffer is infinite, so there can be no overflow. Consider an input flow R, which starts at time t_0, goes through such a node, and produces a corresponding output flow R^*. The minimum rate guarantee implies that, for all $t \geq t_0$,

$$R^*(t) - R^*(t_0) \geq r(t - t_0). \tag{5.17}$$

At time t_0 there is no data accumulated in the node (i.e., there is no backlog), so $R(t_0) = R^*(t_0)$. Substituting into (5.17), we can see that for all t

$$R^*(t) \geq \inf_{0 \leq s \leq t} \{R(s) + r(t - s)\}. \tag{5.18}$$

But the right-hand side is exactly the min-plus convolution of the input flow $R(t)$ and the function $f_r(t) = rt$.

The concept of a *service curve* is now defined, aiming to capture the general effect that a network element has on a flow that traverses it: An element S offers a service curve β to a flow with input function R and output function R^*, if and only if β is non-decreasing, $\beta(t) = 0, t < 0$ and $R^* \geq R * \beta$, where $*$ denotes convolution in the min-plus semiring.

Consider now a flow traversing two network elements E_1, E_2, connected in series, with input and output flows, consecutively, R_1, R_1^*, R_2, R_2^*. Element E_i offers a service curve $\beta_i, i = 1, 2$ to the flow. Applying the definition of the service curve at E_1 and E_2, we have

$$R_1^* \geq R_1 * \beta_1 \tag{5.19}$$
$$R_2^* \geq R_2 * \beta_2. \tag{5.20}$$

Because of the connection in series, it holds that $R_1^* = R_2$. Also, convolution preserves the order of the inequality and is associative. Therefore,

$$R_2^* \geq R_2 * \beta_2 = R_1^* * \beta_2 \geq (R_1 * \beta_1) * \beta_2 = R_1 * (\beta_1 * \beta_2). \tag{5.21}$$

So, comparing with the definition of the service curve, we can see that the concatenation of the two elements offers a service curve of $\beta_1 * \beta_2$ to the flow.

CHAPTER 6

List of Semirings and Applications

In this chapter, we list the semirings and corresponding applications that we described in Chapter 4. We assume that a graph $G = (V, E)$ has been defined, with $|V| = n$. Note that for the applications "Expected cost of paths from vertex s to vertex t," and "Trust: Distance semiring" we need to preprocess the weights before using the semiring.

Application	S	$\oplus$	$\otimes$
Path enumeration	Subsets of $\cup_{i=1}^{n} V^i$	Set union	Latin multiplication
Markov chain stationary distribution	$[0, \infty) \cup \{\infty\}$	+	$\times$
Expected number of visits of random walk from vertex s to t	$[0, \infty) \cup \{\infty\}$	+	$\times$
Expected cost of paths from vertex s to t	$\{(p, c) \in ([0, \infty) \times \mathbb{R})\}$	$(p_1 + p_2, c_1 + c_2)$	$(p_1 p_2, p_1 c_2 + p_2 c_1)$
Minimum weight spanning tree	$\mathbb{R}$	min	max
Longest path	$\mathbb{R} \cup \{-\infty\}$	max	$+$
Shortest path	$\mathbb{R} \cup \{\infty\}$	min	$+$
Widest path	$[0, \infty) \cup \{\infty\}$	max	min
Most reliable path	$[0, 1]$	max	$\times$
Widest-shortest path	$\{(d, b) \in [0, \infty) \times [0, \infty)\}$	lexicographic min	$(d_1 + d_2, \min(b_1, b_2))$
k-shortest paths	$a, b \in \{\mathbb{R} \cup \{\infty\}\}^k$	min-k$\{a_1, a_2, \ldots, a_k, b_1, b_2, \ldots, b_k\}$	min-k$\{a_i + b_j \| i, j = 1, \ldots, k\}$
BGP routing	$\{c, p, r, \textcircled{1}, \textcircled{0}\}$	see Table 4.1	see Table 4.1
Shortest path with time-inhomogeneous edges	$w_1, w_2 : \mathbb{R} \cup \{\infty\} \to \mathbb{R} \cup \{\infty\}$ and $t \to \infty \Rightarrow w(t) \nearrow \infty$	$\min\{w_1(t), w_2(t)\}$	$w_2(w_1(t))$
Reachability under deterministic edge failures	$\{0, 1\}$	max	min
Bridge and cut vertices	$2^E \cup \{\textcircled{0}\}$	Set intersection	Set union
Reachability under probabilistic edge failures	formal polynomials w_e in the variables $x_{ei}, e \in E, i \in \mathbb{N}$	$w_1 + w_2 - w_1 \otimes w_2$	$\prod\{\text{all } x_e \text{ in } w_1 \text{ or } w_2\}$
Shortest paths with edge gains or losses	$\{(c, \mu) \in \mathbb{R} \times (0, \infty)\}$	$\begin{cases} (c_1, \mu_1) & \text{if } \frac{c_1}{\mu_1} < \frac{c_2}{\mu_2} \\ (c_1, \mu_1) & \text{if } \frac{c_1}{\mu_1} = \frac{c_2}{\mu_2} \text{ and } \mu_1 > \mu_2 \\ (c_2, \mu_2) & \text{otherwise} \end{cases}$	$(c_1 + \mu_1 c_2, \mu_1 \mu_2)$

Application	S	$\oplus$	$\otimes$
Trust: Path semiring	$\{(t, c) \in [0,1] \times [0,1]\}$	(t_1t_2, c_1c_2)	$\begin{cases}(t_1, c_1) & \text{if } c_1 > c_2\\(t_2, c_2) & \text{if } c_1 < c_2\\(\max\{t_1, t_2\}, c_1) & \text{if } c_1 = c_2\end{cases}$
Trust: Distance semiring	$\{(a, b) \in ([0, \infty] \times [0, 1])\}$	$(a_1 + a_2, b_1 + b_2)$	$(a_1a_2, a_1b_2 + a_2b_1)$
Trust: PGP computation model	$a_1, a_2 \in \mathbb{N}^k, k \in \mathbb{N}$	+	$\texttt{valid}(a_1)a_2$
Trust: EigenTrust	$[0, 1]$	+	$\times$
Trust: Semantic web	$[0, 1]$	max	$\times$
Social: balance semiring	$\{0, -1, 1, a\}$	see Table 4.2	see Table 4.2
Social: cluster semiring	$\{0, -1, 1, a, b\}$	see Table 4.3	see Table 4.3
Social: Geodetic semiring (length and number of shortest paths)	$\{(d, n) \in ([0, \infty) \cup \{\infty\}) \times (\mathbb{N} \cup \{\infty\})\}$	$\begin{cases}(d_1, n_1) & \text{if } d_1 < d_2\\(d_2, n_2) & \text{if } d_1 > d_2\\(d_1, n_1 + n_2) & \text{if } d_1 = d_2\end{cases}$	$(d_1 + d_2, n_1n_2)$
Social: Geosetic semiring (set of vertices of shortest paths)	finite sets with quadruples of the form $(i, P, j, d), i, j \in V, P \subseteq V, d \in \mathbb{N}$	see Sec. 4.11.4	see Sec. 4.11.4
Traffic assignment	$a_1, a_2 \in [0, \infty)$	$-\frac{1}{\mu} \log\left(e^{-\mu a_1} + e^{-\mu a_2}\right)$	$a_1 + a_2$
Edge sensitivity: Traffic attraction attack	$[0, \infty) \cup \{\infty\}$	min	+
Edge sensitivity: Traffic repulsion attack	$[0, \infty) \cup \{\infty\}$	min	+

Bibliography

[1] http://maxplus.org/. 5.2.1

[2] F. Baccelli and D. Hong. TCP is Max-Plus Linear and what it tells us on its throughput. *ACM SIGCOMM Computer Communication Review*, 30(4):219–230, 2000. DOI: 10.1145/347057.347548 5.2.1

[3] F. L. Baccelli, G. Cohen, G. J. Olsder, and J.-P. Quadrat. *Synchronization and Linearity: An Algebra for Discrete Event Systems*. John Wiley & Sons, 1992. 4.4, 5.2.1

[4] V. Batagelj. Semirings for social networks analysis. *Journal of Mathematical Sociology*, 19(1):53–68, 1994. 4.11

[5] M. G. H. Bell. Alternatives to dial's logit assignment algorithm. *Transportation Research Part B: Methodological*, 29(4):287 – 295, 1995. DOI: 10.1016/0191-2615(95)00005-X 4.12

[6] R. Bellman. On a routing problem. *Quarterly of Applied Mathematics*, 16(1):87–90, 1958. 1.2.2

[7] S. M. Bellovin. Security problems in the TCP/IP protocol suite. *Computer Communications Review*, 19(2):32–48, 1989. DOI: 10.1145/378444.378449 4.13.1

[8] D. Bertsekas and R. Gallager. *Data Networks*. Prentice-hall Englewood Cliffs, NJ, 2nd edition, 1992. 1.1

[9] D. Bertsekas and J. Tsitsiklis. *Parallel and Distributed Computation: Numerical Methods*. Athena Scientific, 1997. 1.3.1, 3.4.2, 3.4.3

[10] B. Bollobás. *Modern Graph Theory*. Number 184 in Graduate Texts in Mathematics. Springer Verlag, 1998. 1.1

[11] J.-Y. L. Boudec and P. Thiran. *Network Calculus: A Theory of Deterministic Queueing Systems for the Internet*, volume 2050 of *Lecture Notes in Computer Science*. Springer-Verlag, 2001. 5.2.3

[12] P. Butkovič. Max-algebra: the linear algebra of combinatorics? *Linear Algebra and its applications*, 367:313–335, 2003. DOI: 10.1016/S0024-3795(02)00655-9 5.2.2

[13] R. W. Callon. RFC 1195: Use of OSI IS-IS for routing in TCP/IP and dual environments, December 1990. 1.3

[14] B. A. Carré. *Graphs and Networks*. Clarendon Press, Oxford, UK, 1979. (document)

[15] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. MIT Press, Cambridge, MA, 1st edition, 1990. 1.1, 1

[16] E. W. Dijkstra. A note on two problems in connection with graphs. *Numerische Mathematik*, 1:269–271, 1959. DOI: 10.1007/BF01386390 1.2.1

[17] J. Eisner. Parameter estimation for probabilistic finite-state transducers. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 1–8, Philadelphia, July 2002. DOI: 10.3115/1073083.1073085 4.1, 4.2

[18] L. Fleischer and K. Wayne. Fast and simple approximation schemes for generalized flow. *Mathematical Programming*, 91(2):215–238, 2002. DOI: 10.1007/s101070100238 4.9

[19] J. Fletcher. A more general algorithm for computing closed semiring costs between vertices of a directed graph. 1980. DOI: 10.1145/358876.358884 4.2

[20] L. Gao and J. Rexford. Stable internet routing without global coordination. *IEEE/ACM Transactions in Networking*, 9(6):681–692, 2001. DOI: 10.1109/90.974523 4.6

[21] J. Golan. *Semirings and their Applications*. Springer, 1999. (document)

[22] M. Gondran and M. Minoux. *Graphs and algorithms*. John Wiley & Sons, Inc, New York, NY, 1979. (document), 4.7

[23] M. Gondran and M. Minoux. *Graphs, Dioids and Semirings: New Models and Algorithms*. Springer Publishing Company, Incorporated, 2008. (document), 2.2.4, 4

[24] J. Goodman. Semiring parsing. *Computational Linguistics*, 25(4):573–605, 1999. 4.1

[25] D. T. N. R. Group. http://www.dtnrg.org. 4.7

[26] C. L. Hedrick. RFC 1058: Routing information protocol, June 1988. 1.3

[27] J. Hershberger and S. Suri. Vickrey prices and shortest paths: What is an edge worth? In *Proceedings of the 42nd IEEE Symposium on Foundations of Computer Science*, pages 252–259, Las Vegas, NV, October 2001. 4.13, 4.13.1

[28] L. Hooi-Tong. Notes on Semirings. *Mathematics Magazine*, 40(3):150–152, 1967. 2.2.4, 2.2.4

[29] Y.-C. Hu, A. Perrig, and D. B. Johnson. Wormhole attacks in wireless networks. *IEEE Journal on Selected Areas in Communications*, 24(2):370–380, February 2006. DOI: 10.1109/JSAC.2005.861394 4.13, 4.13.1

[30] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina. The EigenTrust algorithm for reputation management in p2p networks. In *WWW2003*, May 2003. DOI: 10.1145/775152.775242 4.10.4

[31] J. Kuntzmann. *Théorie des réseaux (graphes)*. Dunod, 1972. (document)

[32] T. Lengauer and D. Theune. Unstructured path problems and the making of semirings (preliminary version). In *Algorithms and Data Structures: 2nd Workshop, WADS'91, Ottawa, Canada, August 14-16, 1991: Proceedings*, page 189. Springer, 1991. DOI: 10.1007/BFb0028261 5.1, 5.1

[33] J. Lestor R. Ford and D. R. Fulkerson. *Flows in Networks*. Princeton University Press, 1962. 1.2.2

[34] P. Lotito, E. Mancinelli, and J. Quadrat. Traffic Assignment and Gibbs-Maslov Semirings. In *Idempotent Mathematics and Mathematical Physics: International Workshop, February 3-10, 2003, Erwin Schrödinger International Institute for Mathematical Physics, Vienna, Austria*, volume 377, page 209. American Mathematical Society, 2005. 4.12

[35] B. M. Maggs and S. A. Plotkin. Minimum-cost spanning tree as a path-finding problem. *Information Processing Letters*, 26(6):291–293, January 1988. DOI: 10.1016/0020-0190(88)90185-8 4.3

[36] M. Mohri. Semiring frameworks and algorithms for shortest-distance problems. *Journal of Automata, Languages and Combinatorics*, 7(3):321–350, 2002. 4.1

[37] J. Moy. RFC 1131: OSPF specification, October 1989. 1.3

[38] N. Nisan and A. Ronen. Algorithmic mechanism design. In *Proceedings of the 31st ACM Symposium on Theory of Computing*, pages 129–140, Atlanta, GA, 1999. 4.13

[39] J. D. Oldham. Combinatorial approximation algorithms for generalized flow problems. *Journal of Algorithms*, 38(1):135 – 169, 2001. DOI: 10.1006/jagm.2000.1130 4.9

[40] R. Perlman. *Network layer protocols with Byzantine robustness*. PhD thesis, M.I.T., 1988. 4.13.1

[41] R. Ramaswamy, J. B. Orlin, and N. Chakravarti. Sensitivity analysis for shortest path problems and maximum capacity path problems in undirected graphs. *Mathematical Programming*, 102(2):355–369, 2005. DOI: 10.1007/s10107-004-0517-8 4.13.1

[42] Y. Rekhter, T. Li, and S. Hares. RFC 4271: A border gateway protocol 4 (BGP-4), January 2006. 4.6

[43] M. Richardson, R. Agrawal, P. Domingos, et al. Trust management for the semantic web. *Lecture Notes in Computer Science*, pages 351–368, 2003. 4.10.5

[44] G. Rote. Path problems in graphs. *Computing Supplementum*, 7:155–189, 1990. 4, 4.1

[45] A. Sen, K. Candan, A. Ferreira, B. Beauquier, and S. Perennes. On Shortest Path Problems with "Non-Markovian" Link Contribution to Path Lengths. *LECTURE NOTES IN COMPUTER SCIENCE*, pages 859–870, 2000. DOI: 10.1007/3-540-45551-5_72 5.1

[46] D. R. Shier. *Network reliability and algebraic structures*. Clarendon Press, New York, NY, USA, 1991. 4.8.2

[47] J. L. Sobrinho. An algebraic theory of dynamic network routing. *IEEE/ACM Transactions on Networking*, 13(5):1160–1173, 2005. DOI: 10.1109/TNET.2005.857111 4.6

[48] G. Theodorakopoulos and J. S. Baras. On trust models and trust evaluation metrics for ad hoc networks. *IEEE Journal on Selected Areas in Communications*, 24(2):318–328, February 2006. DOI: 10.1109/JSAC.2005.861390 4.10.3

[49] A. Üresin and M. Dubois. Parallel asynchronous algorithms for discrete data. *Journal of the ACM (JACM)*, 37(3):588–606, 1990. DOI: 10.1145/79147.79162 3.4.2

[50] A. Wongseelashote. Semirings and path spaces. *Discrete Mathematics*, 26(1):55–78, 1979. DOI: 10.1016/0012-365X(79)90061-X 2.2.3

[51] P. R. Zimmermann. *The Official PGP User's Guide*. MIT Press, 1995. 4.10, 4.10.3

Authors' Biographies

JOHN S. BARAS

John S. Baras received a B.S. in Electrical Engineering from the National Technical University of Athens, Greece, in 1970, M.S. and Ph.D. in Applied Mathematics from Harvard Univ. in 1971 and 1973. Founding Director of the Institute for Systems Research from 1985 to 1991. Since August 1973 he has been with the Electrical and Computer Engineering Department, and the Applied Mathematics Faculty, at the University of Maryland, College Park. Since 1991 Dr. Baras has been the Director of the Maryland Center for Hybrid Networks (HYNET). His many awards include the 1980 George S. Axelby Prize of the IEEE Control Systems Society and the 2007 IEEE Communications Society Leonard G. Abraham Prize in the Field of Communication Systems. Fellow of the IEEE and a Foreign Member of the Royal Swedish Academy of Engineering Sciences (IVA). He has given many invited plenary lectures at prestigious international conferences including the IEEE CDC, ECC, ECAI, Mobicom. His research interests include control, communication and computing systems.

GEORGE THEODORAKOPOULOS

George Theodorakopoulos is a senior researcher at the École Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland. He received his Ph.D. and M.Sc. from the University of Maryland, College Park, and his B.Sc. from the National Technical University of Athens, Greece, all in Electrical and Computer Engineering, in 2007, 2004, and 2002, respectively.

Half of his Ph.D. work was on the application of the algebraic path problem to trust computation. This work resulted in two awards: the best paper award at the ACM Wireless Security workshop (WiSe 2004) and the IEEE Leonard G. Abraham prize (IEEE Journal on Selected Areas in Communications 2006).

His research interests also include game theory (the other half of his Ph.D.), trust and reputation systems, and network security.