Institutional Repository - Research Portal
Dépôt Institutionnel - Portail de la Recherche
researchportal.unamur.be

University of Namur

THESIS / THÈSE

DOCTOR OF SCIENCES

The algorithm-to-algebra method applied to route aggregation : a tale of roadsigns.

Dynerowicz, Seweryn

Award date:
2015

Awarding institution:
University of Namur

Link to publication

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain
• You may freely distribute the URL identifying the publication in the public portal ?

Take down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Download date: 03. Jul. 2025

The Algorithm-to-Algebra Method Applied to Route Aggregation

A Tale of Roadsigns

Seweryn DYNEROWICZ

Committee

Prof. Gildas Avoine *(external reviewer)*, University of Louvain-la-Neuve (UCL)

Prof. Jean-Noël Colin *(supervisor)*, University of Namur (UNamur)

Prof. Timothy Griffin *(external reviewer)*, University of Cambridge

Prof. Ingrid Moerman *(external reviewer)*, University of Gent (UGent)

Prof. Laurent Schumacher *(supervisor)*, University of Namur (UNamur)

Prof. Pierre-Yves Schobbens *(chair)*, University of Namur (UNamur)

Thesis submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the subject of Computer Science. Publicly defended on June 15, 2015 in Namur (Belgium).

UNIVERSITÉ DE NAMUR

FACULTÉ D'INFORMATIQUE

Faculty of Computer Science, University of Namur

Acknowledgements

I believe that the progress and shape of this doctoral thesis is intimately linked to all the people I encountered along the way. As such, it is impossible to pay tribute to everyone who contributed and helped me in one way or another. While the results presented herein are of a theoretical and technical nature, they are only one of the products of a process spanning several years that was the interspersion of technicality, mathematics, philosophy and human relations among other things. I could focus solely on the technical aspects and the guidance received but this would not honour the complete picture. With this in mind I shall nevertheless attempt this daunting task.

My greatest thanks go to my advisor, Jean-Noël Colin, and my co-advisor, Laurent Schumacher, for granting me the opportunity to pursue the complex adventure of a doctoral thesis. Their ongoing support and trust in my ability allowed me to see it through to the end. I must apologize for all the digressions that occured along the way and thank them for their patience under all circumstances.

To my family and friends for their ongoing support and patience. I tried as much as possible to shield them from the frustration that my investigation brought me and I thank them for their understanding.

I had the pleasure of being part of the Faculty of Computing Science at the University of Namur and after all these years, I can hardly imagine a more pleasant workplace. I am immensely indebted to my fellow colleagues who were part of it. To Gabriel Schwanen for triggering my second fascination for mathematics. To Benjamine Lurquin and Anne-Marie Breny for the delightful moments spent both in front and away from kitchen stoves. To Benoît Vanderose for all the discussions far beyond the scope of computing science. To Gille Gomand whose cheerful disposition never ceased to amaze me and was a great source of motivation. To Marie-Ange Remiche for giving me the best piece of advice when I needed it the most.

My journey was marked by one of the most inspiring internship a doctoral student could ask for. I want to express my gratitude to Timothy Griffin for inviting me to the Computer Laboratory of the University of Cambridge.

Last but not least, I want to thank Julie Heughebaert and Elisabeth Vander Essen, for their presence during the last parts of this difficult endeavour and all the memorable moments we shared together.

Finding appropriate and concise words to thank everyone is well beyond my abilities. I have no other choice but to borrow them from someone else.

> “We all change, when you think about it, we’re all different people; all through our lives, and that’s okay, that’s good you’ve gotta keep moving, so long as you remember all the people that you used to be. I will not forget one line of this, not one day, I swear. I will always remember when the doctor was me.”

To all the people who passed through my life in the recent years and steered me towards this point, I am deeply grateful for your contribution.

Contents

Chapter 1

Notation

Excuse me, I'm making perfect sense. You're just not keeping up.

The Eleventh Doctor.

Notation	Name
$\forall\, x \in X : \mathcal{P}(x)$	Universal set quantification
$\exists\, x \in X : \mathcal{P}(x)$	Existential set quantification
$\neg\, \mathcal{P}(x)$	Logical negation
$\Rightarrow$	Logical implication
$\Leftrightarrow$	Logical equivalence
$\wedge$	Logical conjunction
$\vee$	Logical disjunction
$\triangleq$	Definition
$\oplus$	Additive law
$\otimes$	Multiplicative law
$\bar{0}$	Zero element
$\bar{1}$	Unit element
$\boxplus$	Square matrix additive law
$\boxtimes$	Square matrix multiplicative law
$\mathbf{O}$	Zero matrix
$\mathbf{I}$	Unit matrix
$\leqslant_{\oplus}$	Natural order relation
$\Vert_{\oplus}$	Natural incomparability relation
$+$	Integer addition
$\times$	Integer multiplication
$\cup$	Set union
$\cap$	Set intersection
$\varnothing$	Empty set
$\sqsubseteq$	Subprefix relation
$\mid$	Path alternative
$\diamond$	Path concatenation
$\lesssim$	Pre-order relation
$\vec{\times}$	Lexicographic order
$A, B, \ldots, Y, Z$	Sets of elements
$\mathbf{A}, \mathbf{B}, \ldots, \mathbf{Y}, \mathbf{Z}$	Square matrices over a semiring
$\mathrm{list}(X)$	Set of all lists with elements taken from set X
$\lesssim_{\mathrm{length}}$	Preference relation on lists based on their lengths
$\sim_{\mathrm{length}}$	Equality of lists based on their length
$\mathcal{V}$	Set of vertices in a graph
$\mathcal{A}$	Set of arcs between vertices in a graph
$\leqslant_{\mathcal{V}}$	Preference relation on vertices in a graph
$\mathcal{P}erm(X)$	Set of permutations of the elements of a set

Chapter 2

Introduction

A beginning is the time for taking the most delicate care that the balances are correct.

Manual of Muad'Dib, Princess Irulan Corrino-Atreides.

Over the decades that followed its inception, the Internet evolved in a strongly organic but loosely-engineered way. The nexuses that formed at its top level, known today as Autonomous System (AS), are managed mostly by commercially-driven organizations and are thus governed by relationships where all information does not flow freely to the rest of the world for the establishment of the routes throughout the Internet. This evolution has given rise to a plethora of protocols which were designed and adapted on an as-needed basis, resulting in monolithic entities that require highly qualified administrators for their daily management. Certain specific configurations, colloquially known as configuration errors, can result in route oscillations or disruptions on a global scale with a potential to affect communications beyond the local scope. Furthermore, an intimate knowledge of the inner workings of various protocols is not sufficient to solve the problems that can arise due to the concomitance of various conditions distributed across several domains managed by different organisations.

The complexity and problems that arise through the use of protocols such as BGP [47], IS-IS [42], OSPF [40], RIPv2 [37] or EIGRP [51] stir several questions, both for the understanding of the origin of the problems that are observed in real-world situations and for the design of the protocols that will supersede them. An important guideline cherished by computing scientists is modularity of a solution to a complex problem; its construction being based on the assembly of well understood building blocks. While the longing for such modularity has often been voiced by the networking community as an increasingly pressing necessity [13], it seems to be still missing from the vast literature of its research field. To make things worse, no comprehensive theory of routing and forwarding exists as of today to synthesize the compiled list of protocols and their associated problems.

Over the past decades, the problem of identifying the best paths in a graph has been an object of study from an algebraic perspective [9, 4, 24, 38, 55]. This approach was always somehow limited to metrics and was never fully applied to real-world routing protocols to attempt capturing as many aspects as possible. This changed in the recent years, with attempts to reduce the distance between the algebraic theory of path finding and various mechanisms which can be found in protocols in use nowadays. Metarouting [27, 29, 26, 8, 54] has emerged as a promising approach embracing Dijkstra's separation of concerns [17] by looking at routing protocols from an algebraic perspective. Instead of adopting the radical viewpoint of throwing away the entire corpus of routing protocols and starting from scratch, an approach doomed to fail in the real-world, Griffin and Sobrinho proposed to deconstruct existing routing protocols and study how their common building blocks fit together. A guiding principle behind this approach is the Algorithm-to-Algebra Method [19], whereby aspects that were thought to be algorithmic in nature are progressively transferred to an algebraic dimension of the central problem that routing protocols are solving. The application of this method started with [27], where metrics were captured into structures formed by a total order for path-ranking together with a binary law for producing weights of paths out of those of their constituent links. Later work progressively shifted the interest towards ideas borrowed from operations research as presented by Gondran and Minoux in [25]. The total order was replaced by a binary law, giving rise to a structure known as a *semiring* in universal algebra. This different formalism focused on a pair of linear recurrence equations to capture the solutions produced by link-state and distance-vectoring approaches upon which routing protocols are based. Other concerns such as local versus global optimality [54, 28], Route Redistribution and Administrative Distance [2], Hot and Cold Potato forwarding [8] were progressively studied from this algebraic perspective. Recent work [19] has made explicit the paths associated with the weights manipulated by this framework, thus shifting away from a purely algorithmic perception.

The goal of this thesis is to transfer two related mechanisms in use on the Internet into an algebraic form. The first one, known as Route Aggregation (RA) [36], was introduced as a means of alleviating the load incurred on infrastructure routers by the use of the classful addressing scheme[45] and the associated forwarding scheme. This approach required one entry in every forwarding table for every destination which lied in a shallow hierarchical structure. In this context, a destination typically meant the network of an organisation to which a range of IP addresses was allocated. Route aggregation was enabled by a prior restructuring of the addressing scheme and a rethinking of the allocation method for addresses. In the Classless Inter-Domain Routing (CIDR), not only are end-system addresses grouped into prefixes but prefixes themselves can be regrouped into increasingly more general prefixes. By virtue of the concept of an IP prefix, it became possible to factor common attributes (e.g. next-hop, route length) shared by the routes towards various destinations having IPs falling within the same prefix.

The second mechanism is known as Longest-Match Prefix rule and alters the way forwarding decisions are performed. Among the potentially multiple routes that can be matched against a given address, the one with the most specific prefix (i.e. greatest number of identical most significant bits covered by the network mask) is selected. This means that from the perspective of two physical entities (a source and a destination), the route established between them can be labelled by the most specific prefix in the forwarding table of the source that matches the destination, with the possibility of several other physical destinations being matched by that particular entry.

Figure 2.1 presents the infrastructure which we will be using for the design of a route aggregation model. It involves three ISPs interconnected together, each having various customers to which they are providing connectivity to the rest of the network. The prefixes allocated to a corresponding entity are given next to it. For example, ISP 2 has been allocated the prefix `10.2.0.0/16` with the possibility to sub-allocate it further, as it did for Customer 6. The inter-connected group of ISPs forms the backbone of the infrastructure and provides connectivity between the various customers. In order to be reachable in the global routing infrastructure, an end-system must have an address assigned to it; for example host **H4** has been assigned address `10.2.8.4` which belongs to the prefix allocated to Customer 8. Customer 7 has been allocated prefixes `10.2.7.0/24` from ISP 2 and `10.3.7.0/24` from ISP 3 and represents a case of multi-homing through both of its providers. The internal network of Customer 8 is detailed on Figure 2.1 and represents the case of a re-homed domain which has kept its allocated prefix (a sub-prefix of `10.2.0.0/16` allocated by ISP 2) when it changed its connectivity provider to ISP 3. For any of its host to be reachable from the other customers, that host has to be assigned an IP within the prefix `10.2.8.0/24`.

Destination	Next-hop	Hop count
`10.2.8.1`	**R1**	2
`10.2.8.2`	**R1**	3
`10.2.8.3`	**R0**	3
`10.2.8.4`	**R0**	2

Figure 2.2: Routing table of router **B0** of Customer 8.

In practice, the border router **B0** needs to keep track of how to reach the individual hosts within its internal network in its routing table (see Figure 2.2). With respect to the outside, it need only to advertise it is able to reach any address in the prefix `10.2.8.0/24`. This means that the border router **B1** of ISP 3 receives one route advertisement from router **B0** of Customer 8 for prefix `10.2.8.0/24` which will result in a unique entry in its forwarding table (Figure 2.3). The forwarding table of **B1** without route aggregation would contain 4 entries, for destinations `10.2.8.1` through `10.2.8.4`, all pointing to **C8.B0** as the next-hop. The use of route aggregation enables to merge all those entries into a single one, covering the 256 possible addresses contained in the prefix `10.2.8.0/24` allocated to Customer 8.

At a higher level, prefixes influence how connectivity is established between physically connected enti-

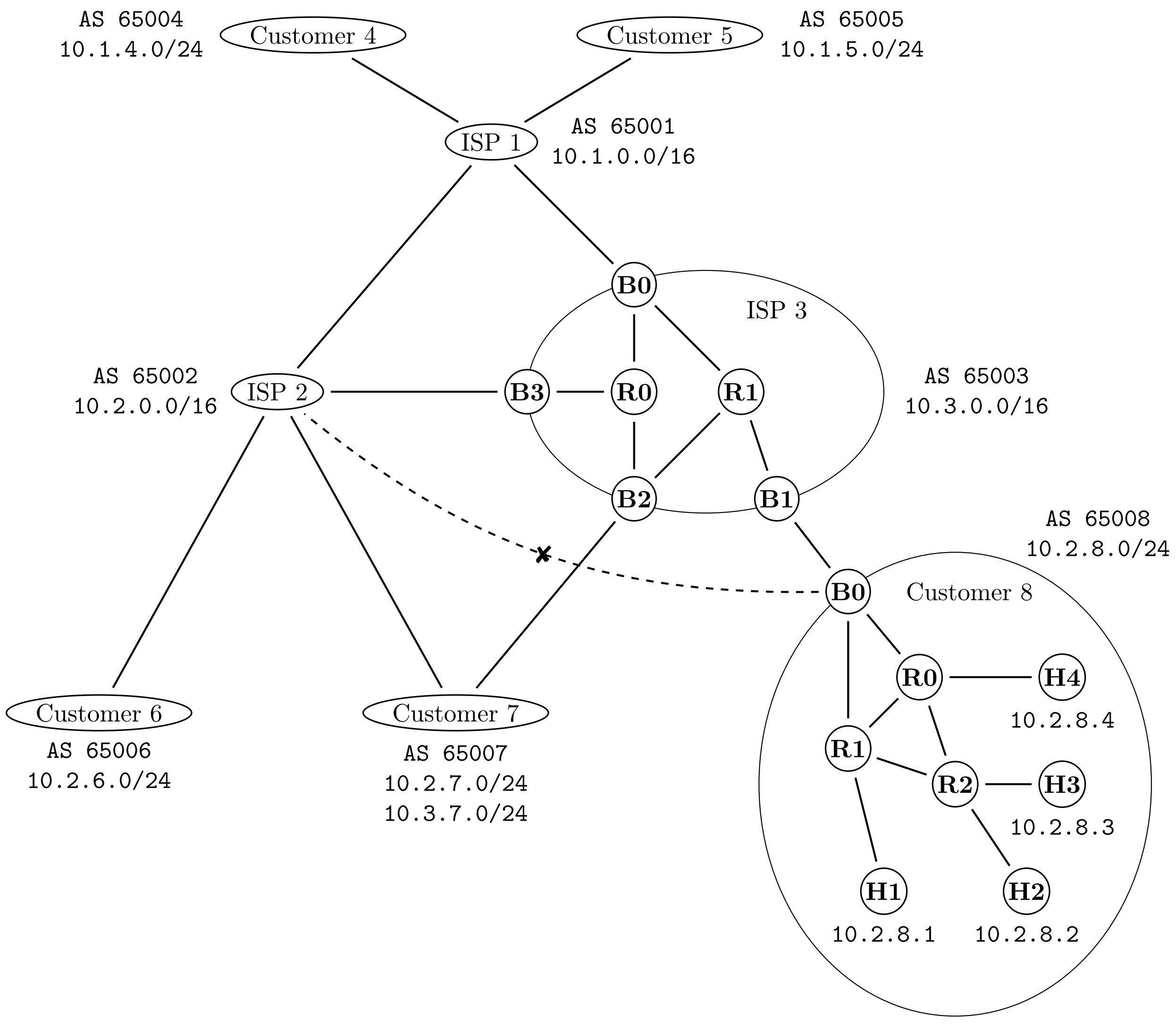


Figure 2.1: High-level view of a networking infrastructure with a detailed view of the internals of ISP 3 and Customer 8. Ellipses represent the domains taking part in the RA process while circles represent end-systems or routers.

ties. For example, ISP 1 will have an alternative to reach Customer 7; a route for the prefix `10.2.0.0/16` (going through ISP 2) and another for the prefix `10.3.0.0/16` (going through ISP 3).

We will present in Chapter 3 how the interplay of the interior and exterior protocols result in the forwarding tables that are used to transmit traffic through the infrastructure in order to reach the destinations intended by the destination address used. We will present the way route information is exchanged and transformed as it traverses multiple domains. The route aggregation mechanism is introduced in this process along with the way routers perform the merging of multiple routes. This method can introduce reachability issues and we will illustrated some of them.

Chapter 4 will cover the theoretical foundations surrounding the constructs relevant for algebraic routing; the study of shortest-paths problems where shortest is understood in the sense of optimal. The exposition will be limited to the elements which are relevant for our needs, a more thorough and wider treatment of this vast topic can be found in [25].

In Chapter 5 we will describe how the tools and results from algebra can be used to model the metrics that are used in real-world scenarios to rank the paths that are progressively identified by routing protocols throughout their execution, an aspect which is relevant for the routing perspective, but also the path-related information which somehow implements the actual paths that are found by those routing

Destination	Next-hop	Distance
`10.1.0.0/16`	**R1**	3
`10.2.0.0/16`	**R1**	5
`10.2.8.0/24`	**C7.B0**	3
`10.3.0.0/16`	**R1**	3

Figure 2.3: Forwarding table used by router **B1** of ISP 3.

protocols, an aspect relevant from a forwarding perspective. We will relate the standard Dijkstra and Bellman-Ford algorithms to the linear recurrence equations introduced in Chapter 4 by proving that these algorithms are computing fixpoints to such equations.

The two inter-related mechanism, route aggregation and longest-match prefix, described above will be modelled within a unique construct in Chapter 6. We will first design a model of route aggregation as it is performed by external peers by removing the internal details of the domains. In this context, routers use admistrator-configured aggregation rules in order to merge routes and provide them consistently to all their neighbors. We will illustrate how re-homing and multi-homing are captured in our model. This attempt will endeavour to reduce the computation of forwarding tables to the same framework which can express the computation of shortest paths, the fixpoints to linear recurrence equations. We will conclude this chapter by showing how reachability issues can be identified by making explicit the relationship between the fixpoints obtained and the routes effectively used. The second part will cover how route aggregation as it is performed internally to a domain can be expressed. In this case, we will discuss the ability of our approach to capture the formation of forwarding loops in the produced solutions.

Chapter 3

An overview of Route Aggregation

Ever since its humble beginnings as a network connecting a dozen of hosts together, the Internet has evolved into the biggest routing infrastructure known today. In order to be able to provide connectivity to its ever-increasing number of hosts, a variety of architectural considerations have been incorporated into its inner workings. Over the decades of evolution and growth it has undergone, Autonomous Systems emerged as the top-level entities in charge of establishing and maintaining the routes to its ever-growing number of destinations.

An Autonomous System, also referred to as a domain, is a network managed by an organisation through which traffic can transit. Figure 3.1 illustrates the structure of a domain where routers are distinguished based on whether they lie at the edge of the domain (border routers), or inside it (internal routers). A domain with a single border router is a customer which only requires it to connect with its provider while a domain with multiple border routers serves as a transit domain. The establishement of routes is performed at two levels, with the interplay of two processes giving rise to the routes that are used to forward traffic. At the intra-domain level, all routers are running an Interior Gateway Protocol (IGP) in order to construct and maintain routing tables describing the best paths towards all the other routers belonging to the same domain but also to the border routers from neighboring domains directly connected to its own. At the inter-domain level, the border routers are running an Exterior Gateway Protocol (EGP) through which they can be engaged in peering sessions with each other in order to exchange information that allows them to maintain forwarding tables describing all the best routes they know. In reaction to the route advertisements it receives, a border router can introduce or retract entries in its forwarding table to maintain connectivity in the context of a changing topology. Aside from the advertisements, border routers can be configured with static routes to originate routes to destinations. A distinction is made between internal and external peering. External peering involves border routers from distinct domains. In this case, various attributes of the routes such as the number of domains traversed to reach the destination are taken into consideration in the process by which the best routes are selected. In the case of internal peering, border routers which belong to the same domain are involved. As such, the attributes of the routes are complemented by the information contained within the routing tables to play a role in the selection of the best routes. For example, if a border router receives a route to the same destination from two different internal peers, the interior cost to reach these peers through the infrastructure may be used in the selection process.

This means that the network administrator can perform traffic engineering [3] within its domain and decide how the load is distributed across its infrastructure by choosing the policy to be applied on the links. Furthermore, the IGP used can incorporate those considerations to choose the paths based on dynamic aspects such as load, delay or reliability. To some extent, the administrator can also influence how traffic flows into the domain by the use of the `MULTI_EXIT_DISC` attribute or multi-homing.

Two pieces of information have to be obtained by the organisation managing a network to be able to act as an AS and participate in the global infrastructure; an AS number (ASN) and a network prefix. An ASN [47, 60] is a number which uniquely identifies an AS in the global infrastructure and is assigned by the Internet Assigned Numbers Authority (IANA)[30]. A network prefix is a set of addresses that are managed by an organisation who is allowed to either assign them to hosts within its network or allocate a subset to another organisation, typically one of its customers. At the root of this hierarchy, the IANA is in charge of managing the prefix `0.0.0.0/0` which contains all IPv4 addresses. The IANA allocates

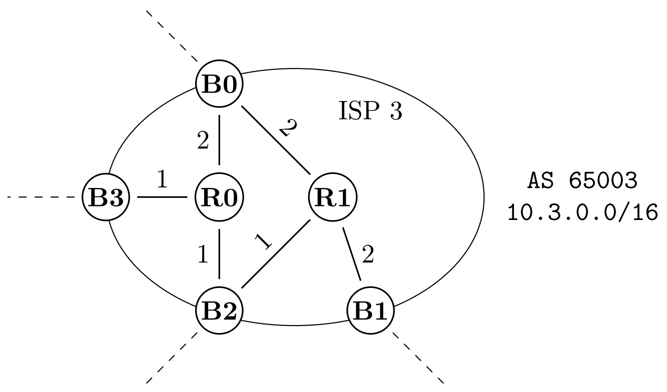


Figure 3.1: Internal structure of an Autonomous System.

subprefixes of important size (/8) to five Regional Internet Registries (RIR), each of which is tasked with the allocation for a vast geographical region[1]. Below the RIRs are Local Internet Registries (LIR) which are typically Internet Service Providers (ISPs) or other organisations who assign most of the block they obtain to their customers. The AS on Figure 3.1 has been allocated `ASN 65003` with the network prefix `10.3.0.0/16` which means that it can manage the IP addresses ranging from `10.3.0.0` through `10.3.255.255` as it sees fit.

The process by which a packet is handled by routers is a destination-based hop-by-hop forwarding [41]. Whenever a packet is received by a border router, a lookup is performed within its forwarding table to identify the most suitable matching entry based on the destination address. Conceptually, the next-hop represents the next domain along the way when the destination lies outside the current domain and is represented by the external peer of the next domain along the way. In the case where this peer is connected to another border router than the one who received the packet, the routing table provides the information to get the packet through the current domain to that border router. The repetition of this process by the border routers of the successive domains eventually brings the packet to its final destination. Similarly, within a domain, the routers perform a process that brings the packet from the border router where it entered to the one that is connected to the external peer. The routing tables inside the domains complement the forwarding tables at the AS level in order to fully implement the routes that packets are expected to follow towards their destination.

Instead of keeping track of a route for each destination network, routers can aggregate them together when they are compatible in some way. A domain that provides connectivity to multiple customers can choose to conceal the routes to their specific network prefixes behind the one it has been allocated and take care of forwarding any packet to any of them according to the effective routes that it knows. By keeping the details of how to reach several destination to itself, a domain can advertise a route to a more general destination that covers all of the child routes, thus reducing the amount of routes that it advertises to its neighbors. This mecanism was originally introduced as a means to slow down the growth rate of the global forwarding table by limiting the amount of information that needs to be advertised to other domains to a strict minimum. Another issue that route aggregation mitigates is known as route flapping [59], where changes to routes in a portion of the infrastructure that is being aggregated remain concealed thus preventing unecessary processing and communications in other routers.

In this chapter, we will describe how route aggregation works. We will start by presenting what destinations represent and how they relate to hosts or networks. We will be focusing on IPv4 [45] coupled with the Classless Inter-Domain Routing (CIDR) scheme [23]. We will briefly present how an IGP constructs routing tables within a domain. The routing tables play a role in completing the information that is discovered by an EGP in order to accept only routes that are usable through its domain. The *de facto* standard EGP in use today is known as the Border Gateway Protocol (BGP-4). We will describe the process of route advertisement, how the best routes are selected based on their attributes and how border routers update those attributes upon exchanging their routes with their peers. We will conclude this chapter by presenting the route aggregation mecanism. The aggregation can happen in multiple places and under various conditions. Our focus will be on the aggregation mecanism at the BGP-4

[1]As of 2011, all /8 prefixes have been allocated by the IANA to the RIRs

level between external peers and within a simplified IGP. The separation between the forwarding table and the advertised routes is the means by which it is implemented. Each border router maintains the effective routes that it uses for forwarding purposes and the set of routes that it advertises to its peers separate. By advertising aggregate routes but forwarding according to its effective routes, the border routers can maintain reachability towards destinations while helping its external peers reduce the size of their forwarding tables. The specifications of EGP and IGP leaves some freedom for the implementation with the consequence that routers from different manufacturers behave in different ways. From the perspective of BGP, the problem lies in how the attributes of an aggregate route are obtained from the child routes. In [36], Le & al. covered many subtle differences observed in hardware from different manufacturers. We will conclude this chapter by looking at how the default mecanism in Cisco routers can result in reachability issues.

3.1 Addressing schemes

An addressing scheme works by attaching some information to physical vertices that allows them to be reachable through the construction of routing and forwarding tables. In [45], IPv4 addresses are 32-bit identifiers that are used to label packets so that routers can use their tables to forward traffic towards the hosts using them. Other schemes have been proposed over the years to provide more features. IPv6 [15] was introduced to expand the size of the address space by using 64 bits identifiers. An ongoing effort is the Locator-Identifier Separation Protocol [50] which separates the location of a router from its identification, two functionalities currently conflated in IP addresses. Another one is the Compact Routing [34, 58] which studies how routing schemes capture a trade-off between the size of the routing tables produced with respect to the stretch of the routes they encode. The stretch refers to a factor that exists between the route found and the shortest one that exists.

In this work, we will be considering the coupling of IPv4 with the CIDR addressing scheme. A CIDR prefix is a concise representation of a range of addresses which can be expressed in a binary format. In the case of IPv4, both IP addresses and network masks are encoded as 32-bit words; for example the prefix `10.1.4.0/24` can be represented as follows

```
10.1.4.0    00001010000000010000010000000000    (address)
     /24    11111111111111111111111100000000    (mask)
```

The idea that a given CIDR prefix is a subprefix of another is central to the aggregation mecanism; whenever an AS is managing a prefix and receives a route advertisement for a prefix covered by its own, the covering prefix should be advertised instead. Consider the following two prefixes

```
10.1.0.0    00001010.00000001.00000000.00000000
     /16    11111111.11111111.00000000.00000000

10.1.5.0    00001010.00000001.00000101.00000000
     /24    11111111.11111111.11111111.00000000
```

The two prefixes `10.1.4.0/24` and `10.1.5.0/24` are subprefixes of `10.1.0.0/16` but are disjoint with each other. Given two prefixes $(\mathtt{a}_1, \mathtt{m}_1)$ and $(\mathtt{a}_2, \mathtt{m}_2)$, we can define the subprefix relation $\sqsubseteq$ by

$$(\mathtt{a}_1, \mathtt{m}_1) \sqsubseteq (\mathtt{a}_2, \mathtt{m}_2) \triangleq \mathtt{m}_2 \leqslant \mathtt{m}_1 \ \wedge \ \mathtt{a}_1 \,\&\, \mathtt{m}_2 = \mathtt{a}_2 \,\&\, \mathtt{m}_2 \tag{3.1}$$

where the & operator stands for the pointwise combination of the two operands by the binary `AND` operation (*a.k.a.* bitwise `AND` [31]). In other words, a prefix is a subprefix of another if-and-only-if the length of the mask of the latter is greater than that of the former and the bits covered by the mask of the latter are identical in both prefixes.

3.2 Interior Gateway Protocol

Despite the diversity of routing protocols designed to run within a domain, all share the common goal of establishing paths between the routers within its boundaries along with the external peers connected to its border routers. The inclusion of the external peers in this process is due to the fact that they are used

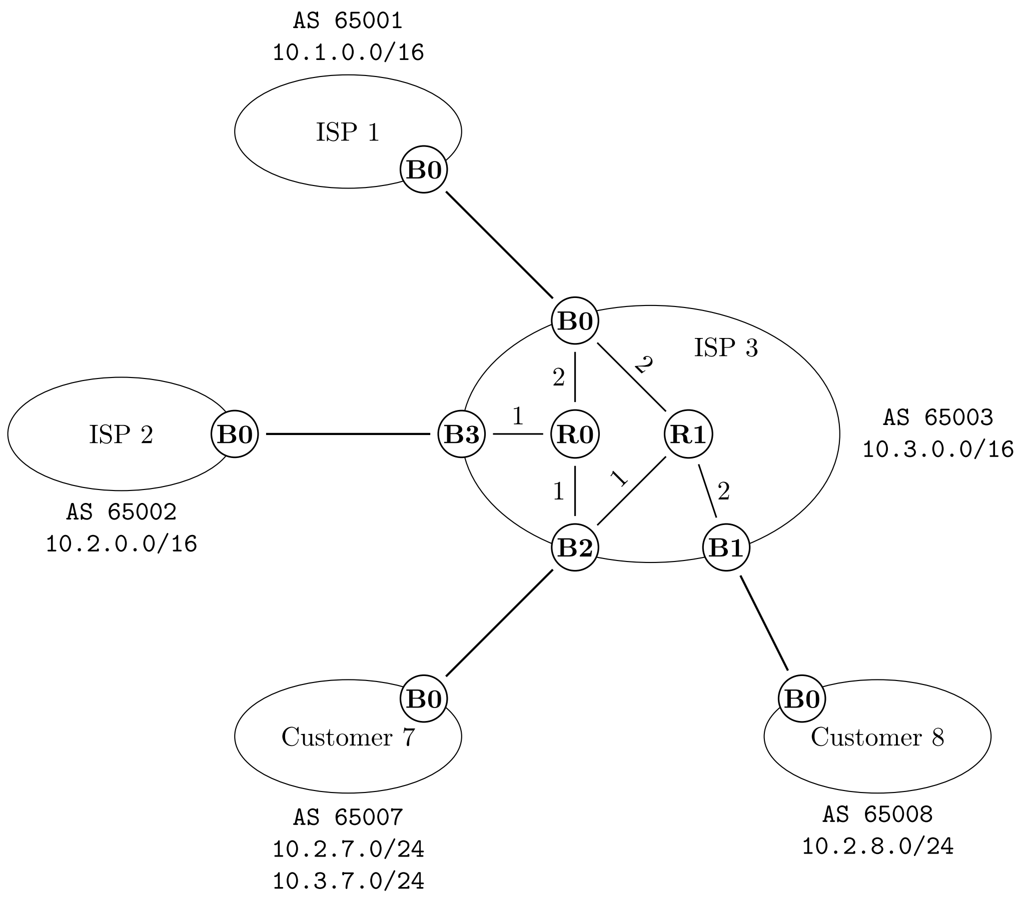


Figure 3.2: Internal structure of ISP 3 with the cost metric.

as the next-hops for route advertisements at the EGP level. Once the decision has to be made as to which route to accept for a destination, the external peer associated with it has to be reachable for the route to be usable by a border router. In practice there are interactions from EGP to IGP, with the possibility for route advertisements to result in the insertion of entries into the routing tables of internal routers regarding the associated destinations. For the sake of clarity, we choose to separate the two concerns but these interactions will be discussed once we have covered EGP. The purpose of this section is to present how an IGP works to produce and maintain the routing tables throughout its execution. We will start by discussing the algorithms underlying the two approaches to routing protocols; Dijkstra's Algorithm[16] for the link-state approach and Bellman-Ford's Algorithm[5] for the distance-vectoring method. We will illustrate a run of both algorithms in ISP 3 as shown on Figure 3.2. In a real-world network, the routers are assigned IP addresses but we omit them for conciseness. The problems that can arise from the use of one or the other will be exposed and discussed. The cost associated with the links are attached to them on the graph and each undirected link is bidirectional. We use undirected links as shorthand for two directed arcs going in both directions with identical costs. In some contexts, it is common for the label to be different for each direction, with complex metrics typically exhibiting an asymmetry in the labelling of the graph. The process by which the routing tables are derived from the solutions produced by those algorithms will be presented along with the forwarding mecanism. For more details about a specific IGP, we point the reader to the corresponding specifications and technical literature.

3.2.1 Link-State Routing Protocol (LSRP)

All the routing protocols falling in the *link-state* class somehow rely on Dijkstra's Algorithm [16, 11], presented here as Algorithm 1, for the computation of a shortest-path tree from which the routing table can be derived. The process that link-state protocols use is structured in three phases; link-state flooding to allow all routers to build a complete representation of the network's topology, execution of Dijkstra's Algorithm to produce a shortest-path tree and finally construction of the routing table based on that shortest-path tree. This algorithm can be easily modified to produce the contents of the routing table directly. In the remainder of this section, the words router and vertex are considered to be synonymous. A prerequisite for the execution of Dijkstra's Algorithm is a complete knowledge of the connectivity within the network by the source router running it. The link-state flooding phase is used to disseminate the topological information from every router to every other. Each entry of the resulting link-state database, $\mathbf{A}_{i,j}$, encodes the cost of the direct link that goes from vertex i to vertex j. A special value ∞ is used to denote the absence of a direct link between the corresponding vertices. At the beginning of the execution, all destinations $d \in \mathcal{V}$ are considered unreachable except the source, which is the router running the algorithm. The termination of the execution is guaranteed by the fact that a set of relaxed vertices $\mathcal{R}$ increases in size at every iteration by inclusion of a relaxation vertex selected at each iteration. Because the source router has a complete knowledge of the topology, it can compare its current knowledge of the cost (ω) towards the direct neighbors of the relaxation vertex q with the current knowledge of the cost to q before using the direct link from q to its direct neighbors. The strategy to choose the relaxation vertex is to pick the closest unrelaxed vertex, a greedy choice that does not prevent the final value of ω for all destinations to be optimal. It is possible to demonstrate that the cost associated with the relaxed vertices is minimal and will not be updated by any later iteration, which is why only unrelaxed destinations are considered on Line (9).

Algorithm 1 Dijkstra's Algorithm for the cost metric

```
 1: for all $d \in \mathcal{V}$ do
 2:     $\omega[d] \leftarrow \infty$
 3: end for
 4: $\omega[s] \leftarrow 0$
 5: $\mathcal{R} \leftarrow \varnothing$
 6: while $\mathcal{R} \neq \mathcal{V}$ do
 7:     pick $q \in \mathcal{V} - \mathcal{R}$ such that $\forall\, q' \in \mathcal{V} - \mathcal{R} : \omega[q] \leqslant \omega[q']$
 8:     $\mathcal{R} \leftarrow \mathcal{R} \cup \{q\}$
 9:     for all $d \in \mathcal{V} - \mathcal{R}$ do
10:         if $\omega[d] > \omega[q] + \mathbf{A}_{q,d}$ then
11:             $\omega[d] \leftarrow \omega[q] + \mathbf{A}_{q,d}$
12:         end if
13:     end for
14: end while
```

The link-state database for Figure 3.2 is given as Table 3.3 with all the internal links. As a complement, the links towards external peers are included with a cost of 0 associated with them. The reason we omit the direction coming from the external peers of the other domains is simply because they do not participate in the IGP within ISP 3 and as such do not provide link-states for their connectivity. In this example, we associate the cost 0 to the external links. This means that the cost to reach an external peer will be given directly by the cost of traversing the domain to the border router it is connected to. When considering the interaction between IGP and EGP, the metric for the external link is not the same as the internal metric. In the presence of multiple independent connections between two domains, the network administrator might decide either to favour the internal cost over the external metric, an approach known as hot-potato routing [56, 57], in order to get the packets faster out of its network. In contrast, cold-potato routing attempts to find the most interesting external route with the potential consequence of keeping packets longer within the network.

A run of Algorithm 1 at router **B2** is given on Figure 3.4. Each line gives the iteration number of the main loop at Line (6), the costs to reach all other routers produced by that iteration along with the relaxation vertex. Once a vertex has been relaxed, its distance is underlined. The first line describes the state after initialization is performed. Due to the fact that external peers are not connected to any further vertices, the iterations that consider them as relaxation vertices do not change the costs known towards other routers. Because of the initialization, the first relaxation vertex is always the source since it has the lowest value of 0. The third iteration involves a non-deterministic situation where both **R0** and **R1** are candidates for relaxation.

We will prove in Chapter 5 that the choice of either candidate results in the same final answer in general. In this case the relaxation of **R0** followed by **R1** brings us to a makes vertex **B0** the only

Source	Destination	Cost
R0	**B0**	2
R0	**B3**	1
R0	**B2**	1
R1	**B0**	2
R1	**B2**	1
R1	**B1**	2
B0	**R0**	2
B0	**R1**	2
B3	**R0**	1
B2	**R0**	1
B2	**R1**	1
B1	**R1**	2
B0	**I1.B0**	0
B3	**I2.B0**	0
B2	**C7.B0**	0
B1	**C8.B0**	0

Figure 3.3: Link-state database for ISP 3.

Iter.	**B0**	**B1**	**B2**	**B3**	**R0**	**R1**	**I1.B0**	**I2.B0**	**C7.B0**	**C8.B0**	Relax. vertex
0	∞	∞	0	∞	∞	∞	∞	∞	∞	∞	*n/a*
1	∞	∞	$\underline{0}$	∞	1	1	∞	∞	0	∞	**B2**
2	∞	∞	$\underline{0}$	∞	1	1	∞	∞	$\underline{0}$	∞	**C7.B0**
3	3	∞	$\underline{0}$	2	$\underline{1}$	1	∞	∞	$\underline{0}$	∞	**R0**
4	3	3	$\underline{0}$	2	$\underline{1}$	$\underline{1}$	∞	∞	$\underline{0}$	∞	**R1**
5	3	3	$\underline{0}$	$\underline{2}$	$\underline{1}$	$\underline{1}$	∞	2	$\underline{0}$	∞	**B3**
6	3	3	$\underline{0}$	$\underline{2}$	$\underline{1}$	$\underline{1}$	∞	$\underline{2}$	$\underline{0}$	∞	**I2.B0**
7	$\underline{3}$	3	$\underline{0}$	$\underline{2}$	$\underline{1}$	$\underline{1}$	3	$\underline{2}$	$\underline{0}$	∞	**B0**
8	$\underline{3}$	$\underline{3}$	$\underline{0}$	$\underline{2}$	$\underline{1}$	$\underline{1}$	3	$\underline{2}$	$\underline{0}$	3	**B1**
9	$\underline{3}$	$\underline{3}$	$\underline{0}$	$\underline{2}$	$\underline{1}$	$\underline{1}$	$\underline{3}$	$\underline{2}$	$\underline{0}$	3	**I1.B0**
10	$\underline{3}$	$\underline{3}$	$\underline{0}$	$\underline{2}$	$\underline{1}$	$\underline{1}$	$\underline{3}$	$\underline{2}$	$\underline{0}$	$\underline{3}$	**C8.B0**

Figure 3.4: Run of Algorithm 1 with router **B2** as the source.

candidate for relaxation for Iteration 6. Iteration 7 through 9 involve several candidates for relaxation, no matter the order in which they are selected, the costs obtained after the final iteration will be the same with the only difference being the successive sets of candidates possible. The complete shortest-path tree resulting from running Algorithm 1 at router **B2** is given on Figure 3.5. The use of oriented links comes from the fact that those indicate the direction in which the path is going.

The greedy strategy to pick the closest unrelaxed vertex for relaxation implies that only n iterations of the main loop from Line (6) are required. The overal complexity of the algorithm depends on how the selection of the relaxation vertex at Line (7) is performed. The use of Fibonacci Heaps [22] to manage the set of unrelaxed vertices results in a complexity on the order of $(m + n\log(n))$ where n is the number of vertices and m the number of arcs.

In order to be useful for the construction of a routing table which associates a next-hop to every destination, the algorithm requires a minor modification to the solution it produces by storing structural information about the shortest-path tree itself. One way this can be achieved is by keeping track of the relaxation vertex [11] through which the last value of the cost was updated by Line (11). In this case the relaxation vertex represents the predecessor of the destination along the way. The inclusion of the predecessor vertex means that the output of the algorithm is no longer unique. There are two paths with the best cost 3 going from **B2** to **B0**, **B2** $\rightarrow$ **R0** $\rightarrow$ **B0** and **B2** $\rightarrow$ **R1** $\rightarrow$ **B0**. Depending on the order in which these two vertices are relaxed, the resulting shortest-path tree will be different. The resulting vectors, ω for the costs and π for the predecessors are given below and can be matched with the tree presented on Figure 3.5. A hyphen $-$ is a generic way to represent that the predecessor is the source router. Another way to phrase this is that all destinations from the π vector with a hyphen are direct neighbors of the source router that executed the algorithm.

The identification of the next-hop towards some destination router by means of the π vector can be

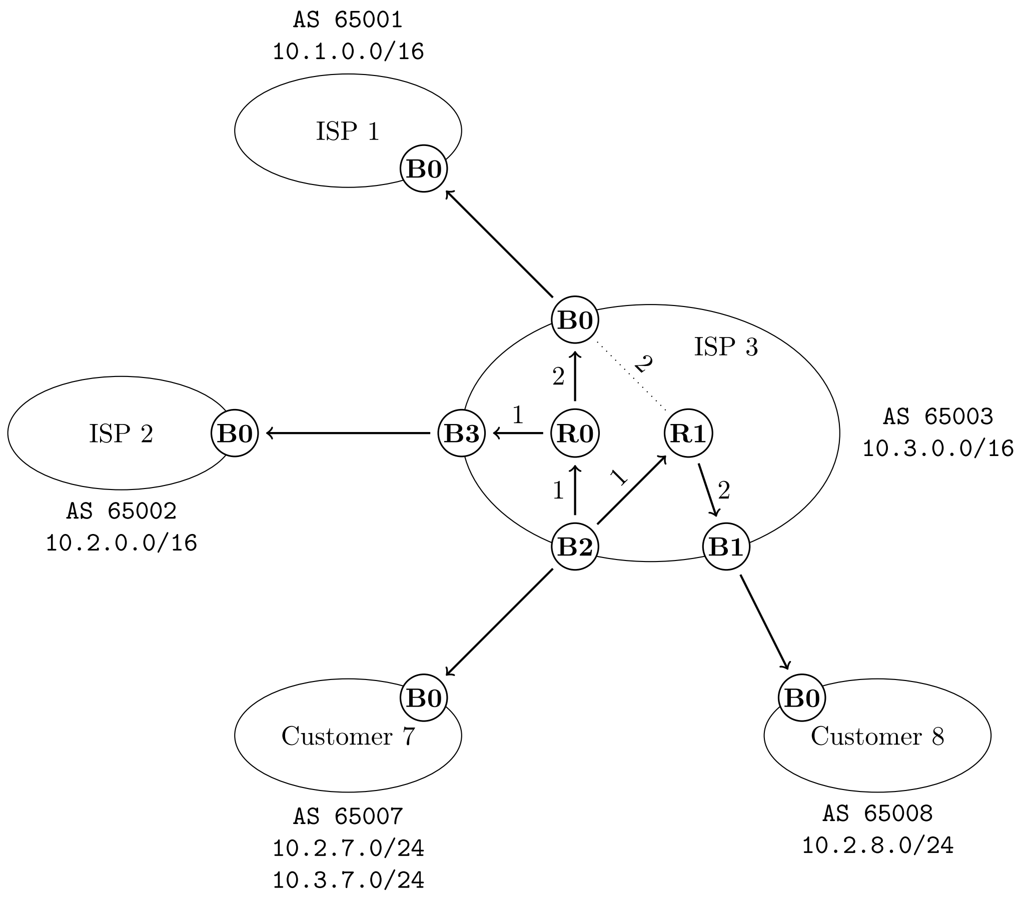


Figure 3.5: The shortest-path tree rooted at vertex **B2**.

	B0	**B1**	**B2**	**B3**	**R0**	**R1**	**I1.B0**	**I2.B0**	**C7.B0**	**C8.B0**
ω	3	3	0	2	1	1	3	2	0	3
π	**R0**	**R1**	–	**R0**	–	–	**B0**	**B3**	–	**B1**

Figure 3.6: Costs and predecessors obtained at the source router **B2**. A hyphen – denotes the fact that the corresponding destinations are directly connected to the source.

Destination	Cost	Next-hop
B0	2	**R0**
B1	3	**R1**
B2	0	–
B3	3	**R0**
R0	1	**R0**
R1	1	**R1**
I1.B0	2	**R0**
I2.B0	3	**R0**
C7.B0	0	**C7.B0**
C8.B0	3	**R1**

Figure 3.7: Routing table of **B2** obtained from the output of Algorithm 1 augmented with predecessors.

	B0	**B1**	**B2**	**B3**	**R0**	**R1**	**I1.B0**	**I2.B0**	**C7.B0**	**C8.B0**
ω	2	4	1	1	0	2	2	4	1	1
π	–	**R1**	–	–	–	**B2**	**B0**	**B3**	**B2**	**B1**

Figure 3.8: Costs and predecessors obtained at the source router **R0**.

done by a recursive lookup of the predecessors until a hyphen is reached. For example, to identify the next-hop for destination **I1.B0**, the sequence of predecessors will be **B0**, **R0**, –, which indicates that **R0** is the next-hop to be used when attempting to reach **I1.B0**. By applying this method for all destinations, the routing table for **B2** can be obtained with the corresponding costs associated with the destinations.

Each router within the network using a LSRP will produce its own routing table independently of the other routers. In the context of next-hop forwarding, this brings the question of consistency across routers. Figure 3.8 presents the vectors ω and π obtained by router **R0** running Algorithm 1. Given that **B2** computes its shortest-path tree independently of **R0** without any form of coordination, one could wonder how the path identified by **R0** (**R0** $\rightarrow$ **B0** $\rightarrow$ **I1.B0**) is consistent with the one computed by **B2** (**B2** $\rightarrow$ **R0** $\rightarrow$ **B0** $\rightarrow$ **I1.B0**). The possibility for inconsistency was identified by Sobrinho & Griffin in [54] where they narrowed down the root of the problem to an algebraic property of the metric being used. They showed that when a complex metric involving bandwidth with a tie-breaking on the distance for the selection of best paths was used, routers could disagree on the best path from their perspective towards a destination. In the worst case, this can result in routing loops that are distributed across the routing tables of several routers. Dijkstra's Algorithm has already been the object of several generalizations and applications [33, 43].

3.2.2 Distance-Vectoring Routing Protocol (DVRP)

The second class of routing protocols is based on a distance-vectoring method. In contrast with the link-state approach, the routers running a DVRP work by exchanging information about the paths they identified towards destinations. The internal behavior of DVRP can be summarized as 3 phases; reception of path information provided by a direct neighbor with a recomputation of its associated cost, selection between the path by comparing its updated cost with the one currently known towards the corresponding destination and sending of updates to the direct neighbors. We delay the presentation of any pseudo-code for the Bellman-Ford's Algorithm [5] until Chapter 5.

The only topological information that a router needs to maintain is the cost associated with the outbound links towards its direct neighbors. This adjacency information corresponds to the row of the link-state database with the source set to the current router. In the initial state, only the distances to the direct neighbors are known from the adjacency information. This provides the router with its initial routing table. The path information is represented as tuples that associate a destination with a cost. Upon receiving path information that associates a destination d with a cost c from a neighbor q, the cost is updated to $\mathbf{A}_{i,q} + c$. The resulting cost is compared with the one for the best path currently know to reach d and the one with the lowest value is kept. In the event where the neighbor which provided the new path is identical to the one that provided the path currently in the routing table, the new path replaces it in order to keep the most recent information. If the best path changes due to this operation, the router must announce this update to its direct neighbors as a means to inform them its perspective

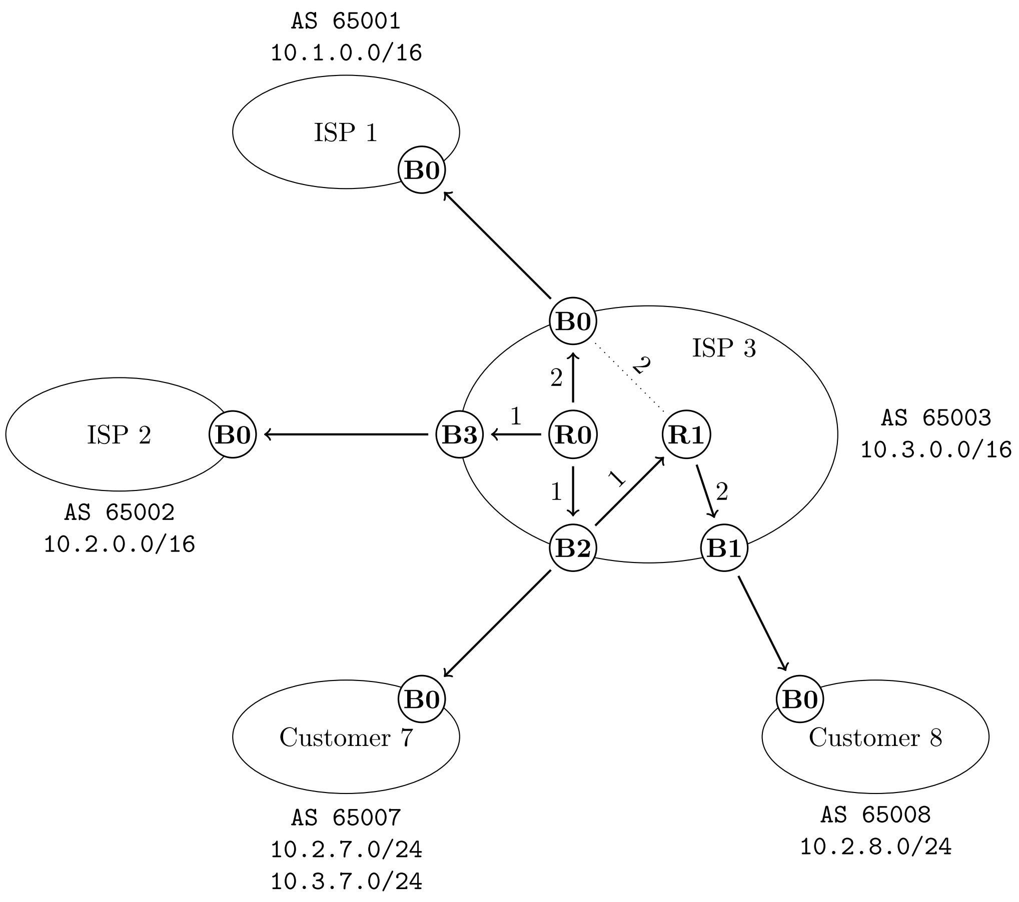


Figure 3.9: The shortest-path tree rooted at vertex **R0**.

	Destination	Distance	Next-hop
B0	**R0**	2	**R0**
	R1	2	**R1**
	I1.B0	0	**I1.B0**

	Destination	Distance	Next-hop
B3	**R0**	1	**R0**
	I2.B0	0	**I2.B0**

	Destination	Distance	Next-hop
B1	**R1**	2	**R1**
	C8.B0	0	**C8.B0**

	Destination	Distance	Next-hop
R0	**B0**	2	**B0**
	B2	1	**B2**
	B3	1	**B3**

	Destination	Distance	Next-hop
B2	**R0**	1	**R0**
	R1	1	**R1**
	C7.B0	0	**C7.B0**

	Destination	Distance	Next-hop
R1	**B0**	2	**B0**
	B1	2	**B1**
	B2	1	**B2**

Figure 3.10: Initial routing tables corresponding to the adjacency information for each router within the boundaries of ISP 3.

on how to reach *d* has changed. Note that a router need not perform this update everytime for every new path information received. It can accumulate the path information over a certain interval of time before running the process over the resulting collection.

Any changes to the topology affect the local routing table whose information is then diffused through the infrastructure without the need to flood any link-state change to all routers. This has the consequence that a DVRP can adapt faster to topological changes. In contrast with a LSRP, the paths used throughout the infrastructure are constructed backward, starting from the destinations and flowing back to other routers. The next-hop consistency question that was raised at the end of the previous section is solved by the fact that the selection process is based solely on the connectivity that neighbors are announcing and not an assumption on the choices made by other routers along the path. This also allows the routing tables to be directly constructed without the use of an intermediary shortest-path tree. Whenever the selection process picks the best path towards a destination, the next-hop is already known as the neighbor who originally provided that path.

The adjacency information for the entire network is provided in the form of the initial routing tables on Figure 3.10 where the distance is given for each router and destination. We assume that the external peers connected to the border routers have an associated distance of 0. Unlike LSRP, it is not possible to focus on the computation being performed by one vertex since the path information is progressively constructed and distributed by all the vertices. For this reason, we will illustrate a run of the algorithm by showing the routing tables of each router of ISP 3. The interleaving in which the routing table updates are exchanged allows for a consequent number of possible executions. For this reason, we choose to illustrate the execution in lockstep by considering that all the routing tables are computed in a syncronized way. The routing tables at one iteration are obtained independently from the routing tables of the previous iteration. The initial routing tables at each router are listed on Figure 3.10 where for each destination, the distance and next-top are given.

During the first iteration, router **B2** will receive the routing tables of routers **R0** and **R1**. The collection of paths with updated costs are listed on Figure 3.11. They are obtained by taking the entries of the routing table of **R0** (resp. **R1**) and adding the cost from **B2** to **R0** (resp. **R1**) to the cost of each entry.

Figure 3.12 gives the new routing table for router **B2**, which is obtained by picking for each destination the path with the least cost among those from the collection of updated paths on Figure 3.11 and the previous routing table on Figure 3.10. The paths for destination **B2** are discarded due to the implicit path with cost 0 that always exists between a router and itself. For destination **B0**, the paths through **R0** and **R1** both have the same cost so either of them could serve as the next-hop. The origin of each path information is used as the next-hop in the routing table.

The reachability has increased for all routers after one iteration, with destinations **B0**, **B3** and **B1** now reachable through **R0** and **R1** from **B2**. Through the progressive exchange of updates to their

Destination	Cost	From
B0	3	**R0**
B3	2	**R0**
B2	2	**R0**
B0	3	**R1**
B2	2	**R1**
B1	3	**R1**

Figure 3.11: Collection of paths at router **B2** during the first iteration.

	Destination	Distance	Next-hop
B0	**B1**	4	**R1**
	B2	3	**R0**
	B3	3	**R0**
	R0	2	**R0**
	R1	2	**R1**
	I1.B0	0	**I1.B0**

	Destination	Distance	Next-hop
B1	**B0**	4	**R1**
	B2	3	**R1**
	R1	2	**R1**
	C8.B0	0	**C8.B0**

	Destination	Distance	Next-hop
B3	**B0**	3	**R0**
	B2	2	**R0**
	R0	1	**R0**
	I2.B0	0	**I2.B0**

	Destination	Distance	Next-hop
R0	**B0**	2	**B0**
	B2	1	**B2**
	B3	1	**B3**
	R1	2	**B2**
	I1.B0	2	**B0**
	I2.B0	1	**B3**
	C7.B0	1	**B2**

	Destination	Distance	Next-hop
B2	**B0**	3	**R0**
	B1	3	**R1**
	B3	2	**R0**
	R0	1	**R0**
	R1	1	**R1**
	C7.B0	0	**C7.B0**

	Destination	Distance	Next-hop
R1	**B0**	2	**B0**
	B2	1	**B2**
	R0	2	**B2**
	I1.B0	2	**B0**
	C7.B0	1	**B2**
	C8.B0	2	**B1**

Figure 3.12: Routing tables after the first lockstep iteration.

	Destination	Distance	Next-hop
	I1.B0	0	**I1.B0**
	I2.B0	3	**R0**
	B3	3	**R0**
	B2	3	**R0**
B0	**B1**	4	**R1**
	C7.B0	3	**R0**
	C8.B0	4	**R1**
	R0	2	**R0**
	R1	2	**R1**

	Destination	Distance	Next-hop
	I1.B0	4	**R1**
	I2.B0	5	**R1**
	B0	4	**R1**
	B3	5	**R1**
B1	**B2**	3	**R1**
	C7.B0	3	**R1**
	C8.B0	0	**C8.B0**
	R0	4	**R1**
	R1	2	**R1**

	Destination	Distance	Next-hop
	I1.B0	3	**R0**
	I2.B0	0	**B1**
	B0	3	**R0**
	B2	2	**R0**
B3	**B1**	5	**R0**
	C7.B0	2	**R0**
	C8.B0	5	**R0**
	R0	1	**R0**
	R1	3	**R0**

	Destination	Distance	Next-hop
	I1.B0	2	**B0**
	I2.B0	1	**B3**
	B0	2	**B0**
	B3	1	**B3**
R0	**B2**	1	**B2**
	B1	4	**B2**
	C7.B0	1	**B2**
	C8.B0	4	**B2**
	R1	2	**B2**

	Destination	Distance	Next-hop
	I1.B0	3	**R0**
	I2.B0	2	**R0**
	B0	3	**R0**
	B3	2	**R0**
B2	**B1**	3	**R1**
	C7.B0	0	**C7.B0**
	C8.B0	3	**R1**
	R0	1	**R0**
	R1	1	**R1**

	Destination	Distance	Next-hop
	I1.B0	2	**B0**
	I2.B0	3	**B2**
	B0	2	**B0**
	B3	3	**B2**
R1	**B2**	1	**B2**
	B1	2	**B1**
	C7.B0	1	**B2**
	C8.B0	2	**B1**
	R0	2	**B2**

Figure 3.13: Routing tables after convergence.

routing tables, the routers eventually converge to a set of routing tables which is given on Figure 3.13.

The distance-vectoring approach suffers from problems of its own that a DVRP has to address to guarantee proper working. Due to the dynamic aspects involved in a running network, links are subject to failure. A possible consequence of such a failure is the *count-to-infinity* problem [35].

It is commonplace to discuss LSRP and DVRP in terms of communication overhead, convergence speed and adaptability to dynamic changes of the topology. In [19], we demonstrated that the two approaches can disagree on which are the best paths in a given network labelled by means of a given metric. Equal-Cost Multi-Path (ECMP) is a method by which a routing protocol confronted with multiple equally good paths towards a destination keeps them and applies a load-balancing algorithm when forwarding occurs. By modelling ECMP within an algebraic formalism [25, 27], we were able to identify a sufficient condition for both classes of routing protocols to agree on all best paths towards all destinations. For example, under ECMP, two paths connecting **B2** to **B0** with weight 3 would end up in the routing table, **B2** → **R0** → **B0** and **B2** → **R1** → **B0**, and would be usable for forwarding.

Various techniques exist to allow several IGP to run side by side within the same infrastructure, contributing to the routing tables that are obtained (Administrative Distance) or exchanging the routes they identified among each other (Route Redistribution).

3.2.3 Open Shortest Path First (OSPF)

OSPF [40, 10] is a link-state protocol that uses a cost for making its decisions. This cost is defined as a dimensionless, lower-is-better metric. Whenever relaxation occurs, the cost of the path to the relaxation vertex is added to the one of the link to its neighbor to produce the total cost. The routing tables support CIDR notation and use the LMP rule for forwarding. This enables the route information learned at the edge of the AS to be injected into the routing tables as a means to implement the routes passing through the AS. In order to reduce the convergence time, communication overhead and increase stability of the routing tables across the domain, OSPF introduces a mecanism to divide the internal network of the AS into areas all connected to a special backbone area. There are four types of routers in OSPF terminology. An AS Boundary Router (ASBR) is a router that is communicating with routers belonging to another AS, exchanging the route information and injecting the route information into the AS. An Area Border Router (ABR) lies at the boundaries of multiple areas and runs an instance of the link-state algorithm for each area it has an interface in. It also takes care of summarizing the path information pertaining to the destinations that are reachable through the area it sits in before sending them to the backbone for redistribution into the other areas. A Backbone Router is a router with an interface connected to the backbone and an Internal Router has all its interface in the same area. In the event where multiple paths with identical least cost are identified, OSPF is designed to use Equal-Cost Multi-Path (ECMP).

3.2.4 Routing Information Protocol Version 2 (RIPv2)

RIPv2 [37] is a simple distance-vectoring protocol that relies on a classical concept of distance to reach destinations. The routing table support CIDR notation and LMP, allowing for the route information learned by border routers to be injected into the AS to insert additionnal forwarding entries. The path information stored in the routing table is managed by a garbage collector that cleans up the entries that are not refreshed by the neighbor who originally announced them. Due to the typical network size envisionned and as a means to reduce the time routers spend counting-to-infinity, the distance 16 is used to denote an unreachable destination. A route can also be retracted from the routing table by having the neighbor who originated it send an update for that destination with a distance of 16. Routing loops are prevented by the use of two mecanisms; split horizon and poisoned reverse. Under the split horizon, the routes discovered through a neighbor are not announced back to it. When poisoned reverse is used, the router is allowed to announce the route back to the neighbor who advertised it after setting its associated distance to 16. Whenever a routing loop forms between two neighbors, poisoned reverse forces the removal of both routes.

3.2.5 Enhanced Interior Gateway Routing Protocol (EIGRP)

EIGRP [46] is a hybrid distance-vectoring protocol that was originally developed by Cisco but later turned into an open standard. It uses a custom algorithm known as the Diffused Update Algorithm (DUAL) and a composite metric that includes both static and dynamic aspects; the minimal bandwidth along the path, the load describing how saturated the path is, the delay from the router to the destination and the reliability of the path. The four components are combined according to

$$\left[K_1 \times \text{bandwidth} + \frac{K_2 \times \text{bandwidth}}{(256 - \text{load})} + K_3 \times \text{delay}\right] \times \frac{K_5}{\text{reliability} + K_4}$$

where each factor K_1 through K_5 can be set to alter the metric in subtle ways. It relies on a sufficient condition to guarantee only loop-free paths are selected. The freedom provided by the use of these factors enables network administrators to design their own metric from a restricted space of possibilities.

3.3 Exterior Gateway Protocol

The Border Gateway Protocol (BGP-4) has become the *de facto* standard for managing the exchange of connectivity information between ASes. RFC4271[47] specifies, among other things, the structure of an advertisement, the way in which it they are exchanged, the rules for selecting the best routes that end up in the forwarding table (the import policy) and to which peers they are advertised (the export policy). Concisely, BGP-4 can be described as a path-vectoring protocol for disseminating connectivity information towards destinations represented by IPv4 network prefixes with the goal of constructing destination-based forwarding tables. A path-vectoring protocol constructs routes in a similar way to the way a DVRP does in the sense that participating routers learn routes from their peers and select the

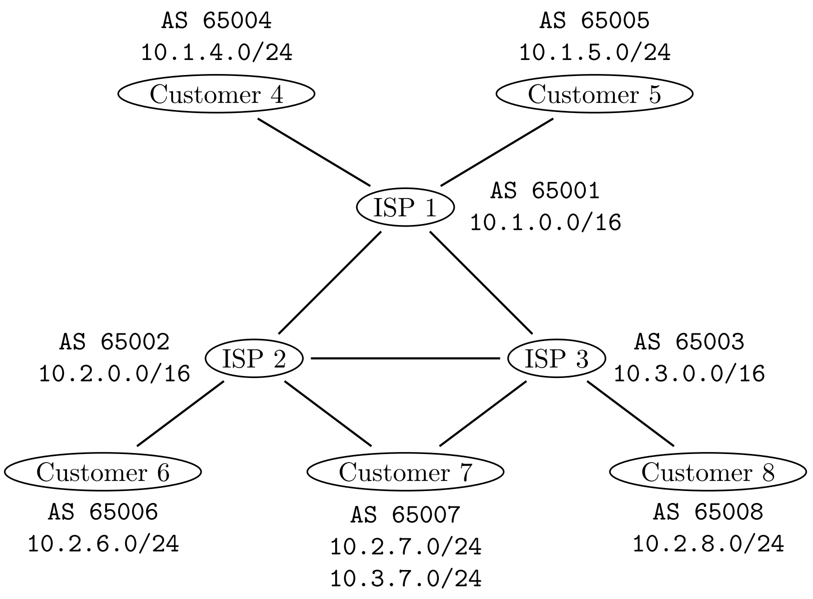


Figure 3.14: High-level view of a networking infrastructure describing prefix allocation.

best ones among those after updating their attributes. In this section, we will give a detailed description of how BGP-4 constructs the forwarding tables based on its configuration. In order to connect with other Autonomous Systems, an AS uses special routers at the edge of its network called BGP speakers. These routers take care of exchanging route information with each other so that forwarding tables can be maintained up to date across ASes. The BGP speakers at the edge of two different ASes can be connected through the BGP-4 protocol (external peering) but it also allows BGP speakers from one AS to communicate with each other (internal peering). Depending on the type of peering, the rules for handling route advertisement differ slightly. We will start by describing the attributes of routes manipulated by BGP speakers. We will then present the structure of the Decision Process that picks the best routes out of an input set along with the conditions that those routes must satisfy. We will end this section by presenting examples where forwarding loops can arise due to misconfiguration. The infrastructure on Figure 3.14 will be used throughout this section to illustrate the main cases where aggregation occurs. It involves three interconnected transit ISPs along with five Customers. The network prefixes allocated to each AS are given in CIDR notation and the AS Numbers are included as well. The example is designed to include three typical scenarios that we wish to cover. We will illustrate throughout this section how these scenarios work under the regime of route advertisement handling followed by BGP-4.

Customers 4 and 5 are both using subprefixes of their provider ISP 1. As such, they fall under the typical case where route aggregation occurs as a means to merge routes advertisements when they pertain to destinations that are covered by the prefix allocated to the AS.

Customer 7 will allow us to illustrate a case of multi-homing, whereby a physical network is reachable through two independent prefixes provided by two distinct providers. This enables the customer to remain reachable when one of its provider is experiencing technical issues with its connectivity. In the event where ISP 2 encounters problems with its infrastructure, all traffic bound for `10.2.6.0/24` will be dropped, effectively rendering Customer 6 completely unreachable. Similarly, the traffic bound for `10.2.7.0/24` will meet the same fate, unlike the one for `10.3.7.0/24` which will be properly delivered. While this situation is undesirable, its occurence is beyond the reach of customers who can still remain partially reachable if they use multi-homing.

Finally, Customer 8 represents a re-homed network. The relationships between providers and customers are commercially-driven. As such, it is possible that a customer decides to resign from a provider to subscribe to another one. In this case, Customer 8 used to obtains its connectivity through ISP 2 but decided to switch to ISP 3. In order to avoid the need to change its network prefix and renumber all its hosts, Customer 8 was allowed to keep the prefix that was allocated by ISP 2.

Attribute	Category
`ORIGIN`	mandatory
`AS_PATH`	mandatory
`NEXT_HOP`	mandatory
`MULTI_EXIT_DISC`	discretionary
`LOCAL_PREF`	only internal
`ATOMIC_AGGREGATE`	discretionary
`AGGREGATOR`	discretionary

Figure 3.15: List of attributes by category and scope.

Note that an alternative exists for the implementation of multi-homing and re-homing in terms of the type of addresses used [1]. Provider-Independent addresses have the specificity that they do not belong to any prefix allocated to domains and as such cannot undergo any aggregation at all unlike Provider-Aggregatable addresses. Provider-Independent addresses are scarce in number due to their inability to be aggregated. If their number was to grow, it would impact the size of forwarding tables across the global infrastructure and hinder the ability of route aggregation to maintain them small. We assume here the use of Provider-Aggregatable addresses.

3.3.1 Route information

The main object manipulated by BGP-4 are routes. A route describes the association of a destination, represented by a network prefix called the Network Reachability Layer Information (NLRI), together with a set of attributes. RFC4271 defines seven attributes that are categorized as either mandatory or discretionary. Mandatory attributes must always be present in route advertisements while discretionary attributes can be omitted. Table 3.15 gives the list of those attributes along with their category. The attribute `LOCAL_PREF` is particular in the sense that it only has meaning in the context of one domain. Whenever a route advertisement is received at a border router of a domain, a preference for that route is computed. This preference is then carried over in subsequent advertisements of that route to the internal peers of the border router who do not recompute it. We will now detail the meaning of the mandatory attributes and go succinctly over the discretionary ones.

The `ORIGIN` is an attribute that describes the means through which the route information was produced. The information can either be generated from within an AS (`IGP`) which injects the route into BGP or discovered through the Exterior Gateway Protocol (`EGP`). This value literally refers to a protocol described in [49]. Another possibility is that the route was produced by the aggregation of several routes or obtained through redistribution from another protocol in which case its value is `INCOMPLETE`. Under route aggregation, if any child route has a value of `INCOMPLETE` for this attribute, the `ORIGIN` of the aggregate route becomes `INCOMPLETE`. Otherwise, if any child route has a value of `EGP`, the `ORIGIN` for the aggregate route becomes `EGP`. Otherwise it has the value `IGP`.

The `AS_PATH` is used to encode the domains that were traversed by the route advertisement. A route is originated by an AS through advertisement to its neighbors. Those neighbors run the selection process and potentially extend and advertise the route further, carrying over the AS Numbers that were traversed by the route information. This attribute is made of path segments which can be either an `AS_SET` or an `AS_SEQUENCE`. An `AS_SET` is an unordered list of ASN that appears in the context of aggregation and will be discussed later. An `AS_SEQUENCE` is an ordered list of ASN that were traversed by the route information. Whenever a border router advertises a route to an external peer, it must prepend the ASN of its domain at the beginning of the first `AS_SEQUENCE` of the `AS_PATH`.

The `NEXT_HOP` field encodes the IPv4 address of the peer through which the destination is reachable. When the route information is exchanged between external peers, this attribute should be set to the address that is used to establish the BGP connection. For internal peering, the border router should not alter this attribute. The forwarding process requires the introduction of a distinction between the `NEXT_HOP` and the immediate next-hop. The immediate next-hop describes the internal router that should be used as a next-hop to reach the `NEXT_HOP`. Consider the example we gave on Figure 3.5 when discussing LSRP and suppose that router **B2** receives a packet from **C7.B0** that is bound towards an address in the network `10.1.0.0/16`. At the EGP level, the only possible route towards that destination has **I1.B0** as its NEXT_HOP. By looking up its routing table, **B2** can identify the immediate next-hop as **R0** and forward the packet to it. When the NEXT_HOP of multiple routes to be aggregated are identical, it is used for the aggregate route. Otherwise the NEXT_HOP is set to the aggregating router address.

MULTI_EXIT_DISC is used whenever multiple external peering connections exist between two ASes. This metric allows to put emphasis on which external link should be favoured when several routes for inbound traffic exist through the same neighboring AS. This attribute enables a domain with multiple entry points from another domain to indicate which one should be favoured with respect to a given destination. In other word, this provides the network administrator with a limited ability to influence how neighboring domains send their traffic into its domain. Routes with different MULTI_EXIT_DISC values cannot be aggregated.

The LOCAL_PREF is an important metric that plays a role in the first stage of the selection of routes within the network of an AS. The first phase of the Decision Process that selects the best routes computes the preference that is given to a route based on its attributes. A typical way this is achieved is by having a configured value set for each neighboring AS which is then used as the preference for the routes provided by each AS. As the route is advertised to internal peers, the LOCAL_PREF is included in this attribute and used directly by the Decision Process of those peers.

ATOMIC_AGGREGATE is used to mark a route that was obtained by aggregating several routes. It denotes the fact that some information was removed from the AS_PATH attribute that describes the aggregated routes.

AGGREGATOR is an attribute that encodes the ASN and the IP address of the border router that performed the aggregation resulting in the route it is associated to. Upon aggregation, the value of this attribute must be replaced by the information identifying the aggregating router.

In the remainder of this section, we will only focus on the three mandatory attributes along with LOCAL_PREF. The rules according to which those attributes are changed throughout the Decision Process will be presented as we describe its inner workings.

3.3.2 Anatomy of a BGP Speaker

From a high-level perspective, a BGP Speaker uses three sets internally as part of its execution; an input set, a set of effective routes and a set of advertised routes. The input set of routes is used to collect the route advertisements received from peers. This set is subject to a unicity constraint on the NLRI attribute and the peer that advertised the route. Whenever a new route for an existing NLRI is received from a peer, the old route from that peer is replaced. The route advertisements are implemented by means of UPDATE messages which encode path attributes together with a set of NLRI. This allows the grouping within one UPDATE of several routes that share all their attributes. Alternatively, those messages can contain information regarding routes which were withdrawn by the peer from its forwarding table. In the event of a disconnection of a BGP session with a peer, the routes received from that peer are removed from the input set to reflect the fact that its state is no longer known and the routes that it used should not be considered for the selection phase.

Any change to the input set triggers a run of the Decision Process that performs the selection of the best routes and their subsequent advertisement. This process is structured in three phases that run in sequence. Phase I is a preprocessing applied to the contents of the input set to establish the degree of preference for each route. Phase II constitutes the core of the selection process by which the BGP speaker establishes the changes that must be applied to its set of effective routes. Phase III takes care of populating the set of advertised routes to generate the new route advertisements for dissemination to the peers.

The calculation of the degree of preference assigns a preference to each route in the input set. The routes that are received from internal peers already carry the LOCAL_PREF attribute which removes the need to perform this computation. While the specifics of how this computation produces the preference are left to the implementation, the only restriction is that it cannot rely on the existence or non-existence of other routes or on the attributes of other routes. It is possible that the outcome of this calculation is a value that renders the route ineligible. For example, a network administrator could decide to blacklist an AS, so that whenever a route carries the ASN of that AS in its AS_PATH, the route is marked as ineligible. Ineligible routes are not taken into account in the remaining phases of the Decision Process

NLRI	LOCAL_PREF	AS_PATH	Source	NEXT_HOP
10.1.0.0/16	100	65001	iBGP	**I1.B0**
10.1.0.0/16	100	65002, 65001	eBGP	**I2.B0**
10.2.0.0/16	100	65001, 65002	iBGP	**I1.B0**
10.2.0.0/16	100	65002	eBGP	**I2.B0**
10.3.7.0/24	100	65007	iBGP	**C7.B0**
10.2.8.0/24	100	65008	iBGP	**C8.B0**

Figure 3.16: Example of contents of the input set for router **B3** with the degree of preference already computed.

but are nevertheless kept in the input set. One way it can be computed is by assigning a value which ranks the neighboring domains to represent the preference of using the routes they provide. The routes received from an AS with a high preference are preferred over those obtained from an AS with a lower preference. A BGP speaker will then pick as the preference for the routes received from a neighbor AS the value which was assigned to it.

Table 3.16 gives an example of input set for router **B4** of Figure 3.2. We assume in this case that all neighboring domains were given a preference of 100 which in turn gives us the value of the `LOCAL_PREF` for all of them. The source denotes whether the route was received through an internal or external peering.

The core of the selection process is performed on the routes from the input set that were marked as eligible by the previous phase. Two additional conditions filter out some of those routes; resolvability and loop-freedom. Given that routes can be obtained through external and internal peering, the latter case requires the ability to resolve the `NEXT_HOP` to an immediate next-hop. In other words, the BGP speaker must know of an internal path through the AS network to reach the external peer captured by the `NEXT_HOP` attribute. This ability is provided by the routing tables that the IGP running within the AS maintains. The BGP speaker refers to its routing table to identify whether it knows a path to reach the `NEXT_HOP`. Another subtle aspect of the resolvability condition is that the installation of the route cannot cause the `NEXT_HOP` to become unresolvable.

Every AS that is traversed by route information must include itself in the `AS_PATH` as part of the handling of route advertisements so that the AS-level route is completely described by this attribute. A BGP speaker from an AS can scan the `AS_PATH` of a route and consider that the presence of its ASN denotes the existence of a forwarding loop, thus rendering the route invalid.

A sequence of tie-breaking rules can then be applied on the set of routes which are eligible, resolvable and loop-free to identify the best ones towards each distinct `NLRI`. The design of these rules guarantees that only one route at most is selected for each destination. The order in which they are applied matches that of the columns listed on Figure 3.16. If multiple routes for a destination have an identical `LOCAL_PREF` which is higher than that of all the other routes, the selection process looks at the `AS_PATH` to determine the route with the least number of ASN among them. A segment of type `AS_SEQUENCE` contributes its length to this number whereas an `AS_SET` counts as 1. The selection process then falls back to the `ORIGIN` attribute and prefers routes that have originated within the AS (`IGP`) over those learned from a neighbor (`EGP`) with the least preferred being the routes obtained by aggregation or redistribution (`INCOMPLETE`). At this stage, in the case where two distinct ASes with equal preference advertised two routes containing an identical number of hops, the internal cost associated with the `NEXT_HOP` is extracted from the routing table and used to pick the best path. As a last resort, the route received from the peer with the lowest address is picked. The contents of the set of effective routes is updated by replacing the previous routes with the ones selected during Phase II, ensuring that at most one route exists for each distinct destination. The updates made to the set of effective routes result in potential changes that have to be made to the routing table to reflect the new forwarding table.

Consider the two routes for the network `10.2.0.0/16` in the input set on Figure 3.16, one received from the external peer **I2.B0** and the other from the internal peer **B0**. By inspection of the attributes of each route according to the tie-breaking rules, the route with the shortest `AS_PATH` is the second one. As a result, the first route bound for `10.2.0.0/16` is discarded. The same rule eliminates the second route for `10.1.0.0/16`.

Suppose now that routers **B0** and **B3** advertise two routes for the destination `10.4.0.0/16` that they learned from their respective external peers. Upon receiving these two advertisements, router **B5** would run the Decision Process in which the tie-breaking rule for the internal cost would result in the selection of the second route. This requires to go back to the routing tables (Figure 3.13) established by IGP where **B3** has a higher internal cost (2) than **B0** (3). On the other hand, router **B1** would favour the first route because from its perspective, **B0** (4) has a lower cost than **B3** (5). Figure 3.18 gives the effective routes that are picked at the end of the selection process. This represents the forwarding table that router **B3** would be using.

NLRI	LOCAL_PREF	AS_PATH	Source	NEXT_HOP
10.4.0.0/16	100	65001,65004	eBGP	**I1.B0**
10.4.0.0/16	100	65002,65004	eBGP	**I2.B0**

Figure 3.17: Two route advertisements for destination `10.4.0.0/16` generated by routers **B0** and **B3** respectively.

NLRI	LOCAL_PREF	AS_PATH	Source	NEXT_HOP
10.1.0.0/16	100	65001	iBGP	**I1.B0**
10.2.0.0/16	100	65002	eBGP	**I2.B0**
10.3.7.0/24	100	65007	iBGP	**C7.B0**
10.2.8.0/24	100	65008	iBGP	**C8.B0**

Figure 3.18: Effective routes used by router **B3**

The final phase of the Decision Process is performed after the best routes have been selected. The set of effective routes is transferred into the set of advertised routes. When a given route is to be announced to an external peer, the `NEXT_HOP` is updated to the address of the current router. Any route withdrawn from the set of effective routes after the first two phases must also be removed from the set of advertised routes. This might happen either because the peer that announced the route has issued an `UPDATE` message to withdraw it or the resolvability condition no longer holds. After the set of advertised routes has been established, the BGP speaker can perform aggregation on its contents.

3.3.3 Aggregation of BGP routes

We will now present how the mecanism of route aggregation works on the set of advertised routes. We will focus on ISP 1 (see Figure 3.19) which sub-allocated the network prefixes `10.1.4.0/24` and `10.1.5.0/24` to two of its customers. Those customers have no other connection to the global infrastructure than through ISP 1.

We assume that the cost associated with the internal links is equal to 1 and give on Figure 3.20 the routing tables that would be produced by an ECMP-enabled IGP. The route advertisements that will be received at routers **I1.B2** and **ISP1.B3** will be identical, see Figure 3.21. Without any form of aggregation, routers **I1.B2** and **I1.B3** would advertise the two routes to their respective external peers, **I2.B0** and **I3.B0**.

Route aggregation leverages the fact that both customers are located behind an AS that manages the network prefix that covers their own. Given that routers **I1.B2** and **I1.B3** have the effective routes towards the specific prefixes, they have the ability to handle any traffic bound for the more general prefix `10.1.0.0/16`. Since ISP 1 is the one in charge of allocating portions of `10.1.0.0/16`, it can find itself with the routes bound for any of its subprefixes allocated to a customer. In general, the BGP speakers of ISP 1 would be configured to aggregate all the child routes falling under the network prefix that it manages. An aggregation rule can be described by a single piece of information; the `NLRI` under which routes can be aggregated. However, it is necessary to define clearly what the resulting values of each attributes are based on the values of the aggregated routes. Only the three mandatory attributes are relevant here, since the `LOCAL_PREF` has no meaning outside the AS that computed it.

The rule for the `ORIGIN` attribute is to set it to `IGP` if all the aggregated routes have it set to `IGP`. If any route has `INCOMPLETE` as its `ORIGIN` attribute, the `ORIGIN` of the aggregate route is set to `INCOMPLETE`.

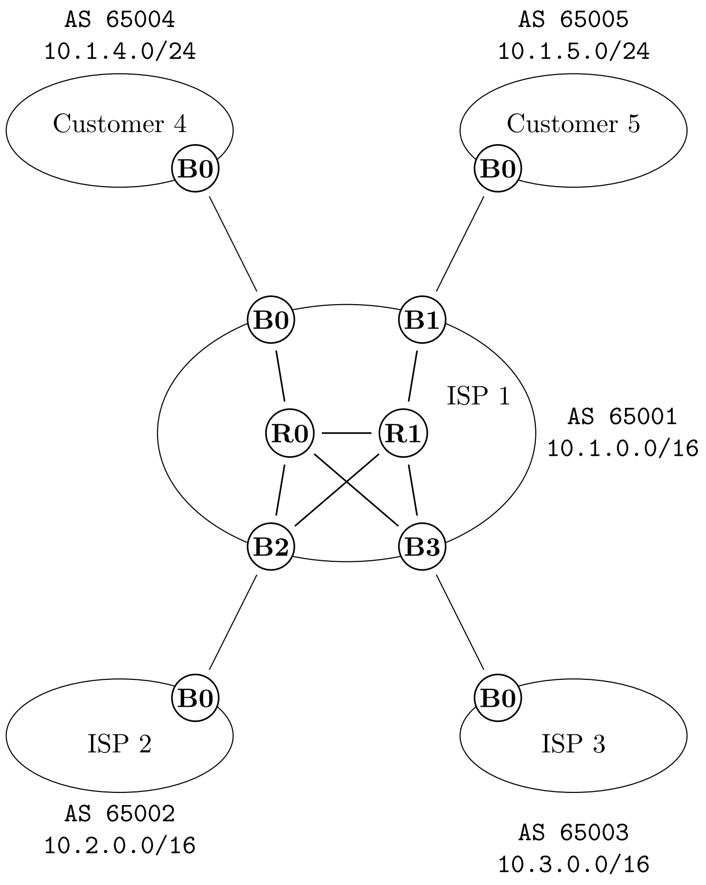


Figure 3.19: Internal structure of ISP 1. The cost of the internal links are assumed to be all equal to 1.

	Destination	Distance	Next-hop
B0	**B1**	3	**R0**
	B2	2	**R0**
	B3	2	**R0**
	R0	1	**R0**
	R1	2	**R0**
	I2.B0	2	**R0**
	I3.B0	2	**R0**
	C4.B0	0	**C4.B0**
	C5.B0	3	**R0**

	Destination	Distance	Next-hop
B1	**B0**	3	**R1**
	B2	2	**R1**
	B3	2	**R1**
	R0	2	**R1**
	R1	1	**R1**
	I2.B0	2	**R1**
	I3.B0	2	**R1**
	C4.B0	3	**R1**
	C5.B0	0	**C5.B0**

	Destination	Distance	Next-hop
B2	**B0**	2	**R0**
	B1	2	**R1**
	B3	2	**R0**, **R1**
	R0	1	**R0**
	R1	1	**R1**
	I2.B0	0	**I2.B0**
	I3.B0	2	**R0**, **R1**
	C4.B0	2	**R0**
	C5.B0	2	**R1**

	Destination	Distance	Next-hop
B3	**B0**	2	**R0**
	B1	2	**R1**
	B2	2	**R0**, **R1**
	R0	1	**R0**
	R1	1	**R1**
	I2.B0	2	**R0**, **R1**
	I3.B0	0	**I3.B0**
	C4.B0	2	**R0**
	C5.B0	2	**R1**

	Destination	Distance	Next-hop
R0	**B0**	1	**B0**
	B1	2	**R1**
	B2	1	**B2**
	B3	1	**B3**
	R1	1	**R1**
	I2.B0	1	**B2**
	I3.B0	1	**B3**
	C4.B0	1	**B0**
	C5.B0	2	**R1**

	Destination	Distance	Next-hop
R1	**B0**	2	**R0**
	B1	1	**B1**
	B2	1	**B2**
	B3	1	**B3**
	R0	1	**R0**
	I2.B0	1	**B2**
	I3.B0	1	**B3**
	C4.B0	2	**R0**
	C5.B0	1	**B1**

Figure 3.20: Routing tables produced by an ECMP-enabled IGP running within ISP 1.

NLRI	LOCAL_PREF	AS_PATH	ORIGIN	NEXT_HOP
10.1.4.0/24	100	65004	eBGP	**B4**
10.1.5.0/24	100	65005	eBGP	**B5**

Figure 3.21: Route advertisements received by **B2** and **B3** from their internal peers **B0** and **B1**. These are also the effective routes that will be selected for forwarding and the ones placed in the set of advertised routes.

In all other cases, the `ORIGIN` is set to `EGP`.

The standard way by which the `AS_PATH` of multiple aggregated routes is produced is by taking the longest leading sequence common to all aggregated routes and appending an `AS_SET` constructed of all ASN found in the remaining parts at the end. In [36], Le & al. present two behaviors for Cisco routers with regards to aggregation; a default behavior which replaces the `AS_PATH` by the ASN of the aggregating network and an optional behavior that additionally adds an `AS_SET` composed of the ASN appearing in the child routes. This optional behavior seems consistent with the standard method described in [47] for the simple cases presented.

When the `NEXT_HOP` is identical for all the aggregated route, it is used as the `NEXT_HOP` for the aggregate route. Otherwise it is set to the address of the BGP speaker performing the aggregation. If an aggregation rule for `10.1.0.0/16` is set at router **B3**, the two routes from Figure 3.21 would be merged into a unique route advertisement that is given on Figure 3.22. This implies that router **I3.B0** will store exactly one route in its set of effective routes that will cover all the possible subprefixes of `10.1.0.0/16`.

NLRI	AS_PATH	ORIGIN	NEXT_HOP
10.1.0.0/16	65001	eBGP	**I1.B3**
10.1.0.0/16	65001, {65004, 65005}	eBGP	**I1.B3**

Figure 3.22: Route advertisement emitted by router **I1.B3** to peer **I3.B0** under the default Cisco behavior (upper) and the optional `AS_SET` inclusion (lower).

3.3.4 Intra-domain aggregation

We conclude this chapter by showing how certain configuration choices can easily result in the disruption of connectivity. In [36], Le & al. studied the differences between multiple implementations of router behavior. In particular, they were looking at routing protocols that provide route aggregation features. At the EGP level, the aggregation is performed at the scope of the Autonomous System. Each AS performs aggregation before advertising routes to their external peers. At the IGP level, depending on which routing protocol is used, the behavior of route aggregation differs. The central issue when route aggregation is performed within a domain is how the interior cost of the aggregate route is computed based on the child routes. In the case of RIPv2, the rule is to set the interior cost to the minimal value of all the child routes. This choice can result in multiple connectivity issues as illustrated on the following example which is based on those appearing in [36] with different prefix allocations.

Figure 3.23 presents the internal structure of ISP 2 on which we will focus. As we mentioned earlier, the route discovery process of BGP can result in the introduction of entries into the routing tables of internal routers in order to fully implement the routes that were found. For example, under no aggregation by the IGP of ISP 2, routes for the destinations obtained from neighboring peers would result in entries for `10.1.0.0/16` (interior cost 4, next-hop **B0**) and `10.3.0.0/16` (interior cost 3 and next-hop **B1**) to be introduced into the routing table of **R0**. The traffic bound for either destination could flow unmodified through the infrastructure and properly reach its intended `NEXT_HOP`.

RIPv2 supports route aggregation which is configured per interface. We will consider here a distance-vectoring protocol which uses a normal distance metric unlike the hop-count bounded to 16 that RIPv2 relies on. We assume that **B0** aggregates on the `10.0.0.0/8` prefix for all routes that it sends through its link to **R0**. Another router **B1** aggregates on the same prefix for all routes sent to **R1**. Strictly speaking, RIPv2 is based on a cost metric that is bounded by the value 16 which counts the number of hops. Le &

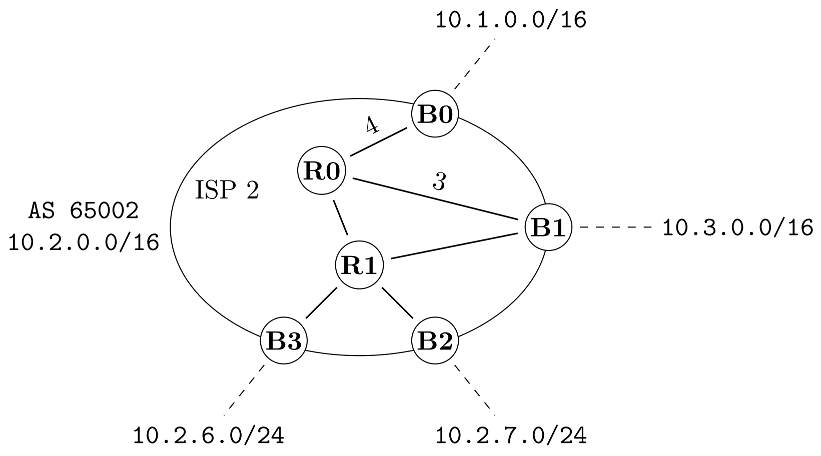


Figure 3.23: Detailed infrastructure of ISP 2 giving rise to reachability issues. All the links have a cost of 1 associated with them except the one between **B0** and **R0**.

Router	Destination	Interior cost	`NEXT_HOP`
B0	`10.1.0.0/16`	0	**I1.B2**
	R0	4	**R0**
B1	`10.3.0.0/16`	0	**I3.B3**
	R0	3	**R0**
	R1	1	**R1**
B2	`10.2.7.0/24`	0	**C7.B1**
	R1	1	**R1**
B3	`10.2.6.0/24`	0	**C6.B0**
	R1	1	**R1**
R0	**B0**	4	**B0**
	B1	3	**B1**
	R1	1	**R1**
R1	**R0**	1	**R0**
	B1	1	**B1**
	B2	1	**B2**
	B3	1	**B3**

Figure 3.24: Initial routing tables.

al. discovered that configuration of route aggregation rules on one interface influences the advertisements that are sent out of other interfaces. A consequence is that **B1** will not announce any route falling in the `10.0.0.0/8` prefix to router **R1**.

As the information is shared as part of the distance-vectoring method, the router will increase their connectivity by discovering new destinations. At some stage, **R0** will compare the route for destination `10.0.0.0/8` received from **B0** with a distance of 4 against the route for the same destination received from **B1** with a distance of 3 and install the latter in its routing table. This route will then be re-advertised to **R1** who will further advertise it to **B1**. As a consequence, the routes marked in Figure 3.24 form a forwarding loop for any traffic bound for the general prefix `10.0.0.0/8` but not for the specific prefix `10.3.0.0/16` which can exit the loop. Suppose that a traffic originating from the lower left portion of the network (Customer 6) is bound for `10.3.0.0/16`. Using the routing tables of Figure 3.24, one can find under the Longest-Match Prefix rule that the traffic will follow the path **B3** → **R1** → **B1** → **I3**.**B3**. However, if a traffic is bound for `10.1.0.0/16`, the path containing a loop will be **B3** → **R1** → **B1** → **R0** → **R1** → .... A common practice to avoid forwarding loops is the installation of *sink-routes* in routing tables; for each aggregation rule a router uses, a route should be present in the routing table with the same `NLRI` where the corresponding action is to drop any traffic matching that route. By virtue of the Longest-Match Prefix rule, any traffic matching the general route but no child route has to be dropped to prevent the traffic from endlessly consuming resources in terms of processing power and bandwidth. This consideration however does not solve the issue that destination `10.1.0.0/16` is unreachable under those forwarding tables, nor does it explain in any way why the forwarding loop formed in the first place.

Router	Destination	Interior cost	NEXT_HOP	Router	Destination	Interior cost	NEXT_HOP
	10.0.0.0/8	6	**R0**		10.0.0.0/8	5	**R1**
	10.2.6.0/24	6	**R0**		10.3.0.0/16	0	**I3.B3**
	10.2.7.0/24	6	**R0**		10.2.6.0/24	2	**R1**
	10.1.0.0/16	0	**I1.B2**		10.2.7.0/24	2	**R1**
B0	**B1**	6	**R0**	**B1**	**B0**	6	**R1**
	B2	6	**R0**		**B2**	2	**R1**
	B3	6	**R0**		**B3**	2	**R1**
	R0	4	**R0**		**R0**	2	**R1**
	R1	5	**R0**		**R1**	1	**R1**
	10.0.0.0/8	5	**R1**		10.0.0.0/8	5	**R1**
	10.2.6.0/24	2	**R1**		10.2.6.0/24	0	**C6.B0**
	10.2.7.0/24	0	**C7.B0**		10.2.7.0/24	2	**R1**
B2	**B0**	6	**R1**	**B3**	**B0**	6	**R1**
	B1	2	**R1**		**B1**	2	**R1**
	B3	2	**R1**		**B2**	2	**R1**
	R0	2	**R1**		**R0**	2	**R1**
	R1	1	**R1**		**R1**	1	**R1**
	10.0.0.0/8	1	**B1**		10.0.0.0/8	4	**R0**
	10.2.6.0/24	2	**R1**		10.2.6.0/24	1	**B3**
	10.2.7.0/24	2	**R1**		10.2.7.0/24	1	**B2**
R0	**B0**	4	**B0**	**R1**	**B0**	5	**R0**
	B1	2	**R1**		**B1**	1	**B1**
	B2	2	**R1**		**B2**	1	**B2**
	B3	2	**R1**		**B3**	1	**B3**
	R1	1	**R1**		**R0**	1	**R0**

Figure 3.25: Routing tables after stabilisation of the distance-vectoring method.

Chapter 4

Algebraic Theory of Routing

"Young man, in mathematics you don't understand things. You just get used to them."

John Von Neumann.

We will start this section by laying out the theoretical foundations on which the remainder of the thesis will rest. This chapter essentially constitutes a review of the results found in [25][54]. The central objects are linear recurrence equations of the form

$$r = (r \otimes a) \oplus b$$
$$l = (a \otimes l) \oplus b$$

where r and l are the unknowns, a and b are constant elements (typically, b is a unit) with binary operations $\oplus$ and $\otimes$ satisfying a set of elementary properties. The classical litterature on the application to path problems [9, 24, 38] considers only the case where the least solutions to those equations coincide with the global optimum of the path problem associated with their resolution; a concern related to optimisation problems where such solutions are desirable. As was pointed out in recent work [54], the problem being solved in routing is merely related to stabilisation of a computation; finding a fixpoint takes precedence over it coinciding with the best possible answer for the path problem being solved. Under such loose conditions, global optima are unobtainable by means of local-search methods. These findings have pointed out that meaning can be found even when some axioms are dropped; the theory can provide sensible answers to real-world problems even in an impoverished form. This chapter serves mainly as a summary of various results pertaining to the path-finding problem.

We will start by giving the relevant definitions and results in Section 4.1. The central specification used for path-problem algorithms takes the form of linear recurrence equations and their respective sets of fixpoints will be characterized. We provide partial result concerning the structure of those sets in the sense that they are closed under the additive law. We will further describe the lift operation to square matrices in Section 4.2, a construct which enables the representation of graphs and solutions over them in a concise but complete way. We will move on to relate those matrices with the path-problem to be solved and give a simple iterative method to do so. In the last section, we shall provide various results related to preorders, orders and sets of minimal elements.

4.1 Elementary definitions and results

The central structure of the algebraic theory of routing is called an IDEMPOTENT semiring [25] which is a set together with two binary laws restricted by simple properties. We will relate progressively the algebraic aspects to the path problem expressed over a labelled graph $\mathcal{G} = (\mathcal{V}, \mathcal{A}, \omega)$ where vertices from a set $\mathcal{V}$ are inter-connected by arcs of $\mathcal{A}$ each of which is given a weight by means of a total function $\omega : \mathcal{A} \to \mathcal{S}$.

Definition 4.1.1. *Let $\mathcal{S}$ be a set endowed with two binary internal laws $\oplus$ and $\otimes$. A semiring is a triplet*

$(\mathcal{S}, \oplus, \otimes)$ *in which the following properties hold;*

$$\begin{array}{rl}
\oplus-\text{COMMUTATIVITY}: & \forall\, a,b \in \mathcal{S} : a \oplus b = b \oplus a \\
\oplus-\text{ASSOCIATIVITY}: & \forall\, a,b,c \in \mathcal{S} : a \oplus (b \oplus c) = (a \oplus b) \oplus c \\
\otimes-\text{ASSOCIATIVITY}: & \forall\, a,b,c \in \mathcal{S} : a \otimes (b \otimes c) = (a \otimes b) \otimes c \\
\oplus-\text{ZERO}: & \exists\, \overline{0} \in \mathcal{S} : \forall\, a \in \mathcal{S} : a = a \oplus \overline{0} = \overline{0} \oplus a \\
\otimes-\text{UNIT}: & \exists\, \overline{1} \in \mathcal{S} : \forall\, a \in \mathcal{S} : a = a \otimes \overline{1} = \overline{1} \otimes a \\
\otimes-\text{ANNIHILATOR}: & \forall\, a \in \mathcal{S} : \overline{0} = a \otimes \overline{0} = \overline{0} \otimes a \\
\text{LEFT-DISTRIBUTIVITY}: & \forall\, a,b,c \in \mathcal{S} : a \otimes (b \oplus c) = (a \otimes b) \oplus (a \otimes c) \\
\text{RIGHT-DISTRIBUTIVITY}: & \forall\, a,b,c \in \mathcal{S} : (a \oplus b) \otimes c = (a \otimes c) \oplus (b \otimes c)
\end{array}$$

Notation 4.1.1. *When no ambiguity can arise, the $\otimes$ symbol will be omitted. Thus if a and b are elements drawn from the ground set $\mathcal{S}$ of a semiring, ab and $a \otimes b$ refer to one and the same element which is called their product.*

This simple structure is useful to model elementary and compound metrics but also other types of information such as next-hops or paths. However, the semiring axioms can turn out to be too restrictive for the variety of metrics that can be imagined. In our model, we will use structures called prebimonoids [54] which satisfy only a subset of those properties. In essence, prebimonoids are composed of a commutative monoid $(\mathcal{S}, \oplus, \overline{0})$, where the additive law satisfies COMMUTATIVITY, ASSOCIATIVITY and admits an IDENTITY, combined with a a non-associative monoid $(\mathcal{S}, \otimes, \overline{1})$ where the multiplicative law is only required to admit an IDENTITY and have the $\overline{0}$ as its ANNIHILATOR. In particular, any semiring is a prebimonoid but the converse does not hold.

Definition 4.1.2. *A prebimonoid is a triplet $(\mathcal{S}, \oplus, \otimes)$ in which the following properties hold;*

$$\begin{array}{rl}
\oplus-\text{COMMUTATIVITY}: & \forall\, a,b \in \mathcal{S} : a \oplus b = b \oplus a \\
\oplus-\text{ASSOCIATIVITY}: & \forall\, a,b,c \in \mathcal{S} : a \oplus (b \oplus c) = (a \oplus b) \oplus c \\
\oplus-\text{ZERO}: & \exists\, \overline{0} \in \mathcal{S} : \forall\, a \in \mathcal{S} : a = a \oplus \overline{0} = \overline{0} \oplus a \\
\otimes-\text{UNIT}: & \exists\, \overline{1} \in \mathcal{S} : \forall\, a \in \mathcal{S} : a = a \otimes \overline{1} = \overline{1} \otimes a \\
\otimes-\text{ANNIHILATOR}: & \forall\, a \in \mathcal{S} : \overline{0} = a \otimes \overline{0} = \overline{0} \otimes a
\end{array}$$

As illustrations, we now describe three classical examples of semirings; the natural dioid, the tropical semiring and the bottleneck algebra, each of which has an interpretation in terms of path problems.

Example 4.1.1. *The structure $(\mathbb{N}, +, \times)$, known as the natural dioid, forms a semiring. It is common knowledge that both operations are* COMMUTATIVE *and* ASSOCIATIVE. *The standard addition admits 0 as its* ZERO *which also doubles as an* ANNIHILATOR *for the standard multiplication. The value 1 serves as the* UNIT *for multiplication and* DISTRIBUTIVITY *between the two operations holds from both sides. In the context of the algebraic path problem, this structure is related to the problem of counting the number of paths connecting two vertices.*

Example 4.1.2. *The structure $(\mathbb{N}^{\infty}, \min, +)$, known as the tropical semiring, forms a semiring. The* min *operation is* ASSOCIATIVE *and* COMMUTATIVE. *Given that ∞ is greater than any other natural integer, this element acts as its* ZERO. *The addition $+$ is extended to work with ∞ which becomes its* ANNIHILATOR. *The value 0 is the* UNIT *for addition and also happens to be the least element of the set, making it the* ANNIHILATOR *for* min. DISTRIBUTIVITY *can be shown to hold from both sides. In the context of the algebraic path problem, this structure is the one underlying the classical shortest-path problem.*

Example 4.1.3. *The structure $(\mathbb{N}^{\infty}, \max, \min)$, known as the bottleneck algebra, forms a semiring. The* max *and* min *operations are both* ASSOCIATIVE *and* COMMUTATIVE. *Given that no element is greater than ∞, that element cannot be the minimum with respect to any other element but has to be the maximum with respect to any other element. This makes ∞ the* UNIT *for* min *and the* ANNIHILATOR *for* max. *Conversely, 0 is the minimum with respect to any other element and the maximum with respect to no other elements, making it the* ZERO *for* max. DISTRIBUTIVITY *holds from both sides. This structure is used to model path problems involving bandwidth.*

It is commonplace to restrict the additive law considered when modelling path problems in particular by considering the SELECTIVE case. While this assumption is suitable when studying single path problems, it appears to be far too restrictive when looking at multipath problems. We start by stating a more general version in terms of IDEMPOTENCY.

Definition 4.1.3. *A prebimonoid* $(\mathcal{S}, \oplus, \otimes)$ *is said to be* IDEMPOTENT *when its addition satisfies*

$$\oplus - \text{IDEMPOTENCY}: \quad \forall\, a \in \mathcal{S} \;:\; a = a \oplus a$$

When dealing with optimisation problems, it is necessary to have a concept of ordering to express the idea of better or best solution. Any IDEMPOTENT prebimonoid is endowed with a natural order relation which is a lower-is-better ordering. The objective for the path-problem is to identify the paths with the most interesting value of the metric, which happens to coincide with the lowest value.

Lemma 4.1.1. *Every* IDEMPOTENT *prebimonoid comes with a partial order called the natural order defined as*

$$a \leqslant_{\oplus} b \;\Leftrightarrow\; a = a \oplus b \tag{4.1}$$

Proof. Each of the three properties of a partial order is proved separately.

- REFLEXIVITY. $a \leqslant_{\oplus} a \;\Leftrightarrow\; a = a \oplus a$ is equivalent to IDEMPOTENCY by definition.
- TRANSITIVITY. $a \leqslant_{\oplus} b \;\wedge\; b \leqslant_{\oplus} c \;\Rightarrow\; a \leqslant_{\oplus} c$.

$$\begin{aligned} a \leqslant_{\oplus} b \;\Leftrightarrow\; a = a \oplus b \;&\Leftrightarrow\; a = a \oplus (b \oplus c) && (b \leqslant_{\oplus} c)\\ &\Leftrightarrow\; a = (a \oplus b) \oplus c && (\text{ASSOCIATIVITY})\\ &\Leftrightarrow\; a = a \oplus c && (a \leqslant_{\oplus} b)\\ &\Leftrightarrow\; a \leqslant_{\oplus} c \end{aligned}$$

- ANTI-SYMMETRY. $a \leqslant_{\oplus} b \;\wedge\; b \leqslant_{\oplus} a \;\Rightarrow\; a = b$.

$$\begin{aligned} a \leqslant_{\oplus} b \;\Leftrightarrow\; a = a \oplus b \;&\Leftrightarrow\; a = b \oplus a && (\text{COMMUTATIVITY})\\ &\Leftrightarrow\; a = b && (b \leqslant_{\oplus} a) \end{aligned}$$

□

The natural order for $(\mathbb{N}^{\infty}, \max, \min)$ coincides with the greater than order on natural integers. Within the context of the path problem, the structures are often restricted to SELECTIVITY, given that metrics in routing problems are used to select the best path among several alternatives available.

Definition 4.1.4. *A prebimonoid* $(\mathcal{S}, \oplus, \otimes)$ *is said to be* SELECTIVE *when its addition satisfies;*

$$\oplus - \text{SELECTIVITY}: \quad \forall\, a, b \in \mathcal{S} \;:\; a = a \oplus b \;\vee\; b = a \oplus b$$

A direct consequence of SELECTIVITY is the TOTALITY of the natural order, a key property for the applicability of Dijkstra's Algorithm. In the case of the tropical semiring $(\mathbb{N}^{\infty}, \min, +)$, the natural order coincides with the order of natural integers, with the exception that ∞ is part of it. Any two integers are related in one way $(n_1 \leqslant n_2)$ or the other $(n_2 \leqslant n_1)$.

Lemma 4.1.2. *The natural order of a* SELECTIVE *prebimonoid* $(\mathcal{S}, \oplus, \otimes)$ *is* TOTAL

$$\forall\, a, b \in \mathcal{S} \;:\; a \leqslant_{\oplus} b \;\vee\; b \leqslant_{\oplus} a$$

Proof. Given two elements $a, b \in \mathcal{S}$, we can rewrite SELECTIVITY by definition of the natural order

$$\begin{aligned} a = a \oplus b \;\vee\; b = a \oplus b \;&\Leftrightarrow\; a = a \oplus b \;\vee\; b = b \oplus a && (\oplus - \text{COMMUTATIVITY})\\ &\Leftrightarrow\; a \leqslant_{\oplus} b \;\vee\; b \leqslant_{\oplus} a && (\text{Definition (4.1)}) \end{aligned}$$

□

Consider the simple directed graph on Figure 4.1 over which we will illustrate the $(\mathbb{N}^{\infty}, \min, +)$ structure. The only two possible paths going from source **2** towards destination **5** are $\mathbf{2} \to \mathbf{1} \to \mathbf{5}$ and $\mathbf{2} \to \mathbf{3} \to \mathbf{1} \to \mathbf{5}$. Their respective weights can be obtained as follows

$$\omega(\mathbf{2} \to \mathbf{1} \to \mathbf{5}) = \omega(\mathbf{2} \to \mathbf{1}) + \omega(\mathbf{1} \to \mathbf{5}) = 1 + 2 = 3$$
$$\omega(\mathbf{2} \to \mathbf{3} \to \mathbf{1} \to \mathbf{5}) = \omega(\mathbf{2} \to \mathbf{3}) + \omega(\mathbf{3} \to \mathbf{1}) + \omega(\mathbf{1} \to \mathbf{5}) = 1 + 1 + 2 = 4$$

This illustrates how the multiplicative law of the prebimonoid is used; as the operation that relates concatenation of arcs to combination of their weights

$$\omega(v_1 \to v_2 \to v_3) = \omega(v_1 \to v_2) \otimes \omega(v_2 \to v_3)$$

For an arbitrary number of arcs we have the general rule

$$\omega(v_1 \to v_2 \to \cdots \to v_{l-1} \to v_l) = \omega(v_1 \to v_2) \otimes \ldots \otimes \omega(v_{l-1} \to v_l)$$

Note that $\otimes -$ ASSOCIATIVITY would guarantee that the weight of a path is well-defined no matter how we parenthesise the expression above. Given the lack of this property in prebimonoids, we assume that the parentheses associate on the right. Given the weights of two paths connecting the same source and destination as was the case for $\mathbf{2} \to \mathbf{1} \to \mathbf{5}$ and $\mathbf{2} \to \mathbf{3} \to \mathbf{1} \to \mathbf{5}$, the min operation is used to select the best path based on their distance. The additive law provides an implicit selection of paths based on their weights.

$$[\omega(\mathbf{2} \to \mathbf{1} \to \mathbf{5})\ \min\ \omega(\mathbf{2} \to \mathbf{3} \to \mathbf{1} \to \mathbf{5})] = [3\ \min\ 4] = 3 = \omega(\mathbf{2} \to \mathbf{1} \to \mathbf{5})$$

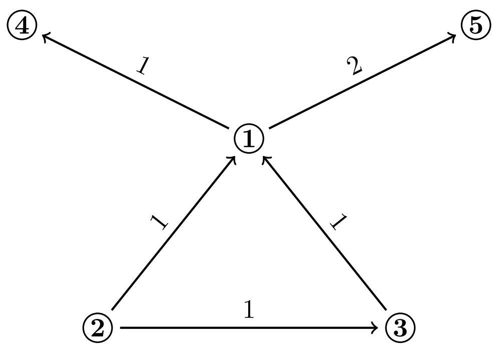


Figure 4.1: A simple graph in the context of the distance metric $(\mathbb{N}^{\infty}, \min, +)$.

The ZERO serves as a special element for the natural order by bounding it from above; a prebimonoid always contains a least interesting element with respect to all the others. The ZERO is typically used to label invalid paths so that any valid path wins over it. In the case of $(\mathbb{N}^{\infty}, \min, +)$, the absence of arcs is denoted by ∞; whenever a sequence of arcs includes one or more arcs missing from the graph, the resulting weight becomes ∞, which is the TOP of the tropical semiring. The path $\mathbf{3} \to \mathbf{2} \to \mathbf{1} \to \mathbf{5}$ on Figure 4.1 has the following weight

$$\omega(\mathbf{3} \to \mathbf{2} \to \mathbf{1} \to \mathbf{5}) = \omega(\mathbf{3} \to \mathbf{2}) + \omega(\mathbf{2} \to \mathbf{1}) + \omega(\mathbf{1} \to \mathbf{5}) = \infty + 1 + 2 = \infty$$

In comparison to the path $\mathbf{3} \to \mathbf{1} \to \mathbf{5}$ which has a weight of $1 + 2 = 3$, this path is less interesting due to its weight coinciding with the TOP of the tropical semiring.

Lemma 4.1.3. *Let $(\mathcal{S}, \oplus, \otimes)$ be an* IDEMPOTENT *prebimonoid. Its* ZERO *is the* TOP *element with respect to the natural order.*

$$\forall\, a \in \mathcal{S}\ :\ a \leqslant_{\oplus} \overline{0} \qquad \text{(TOP)}$$

Proof. By definition of the ZERO, $\forall\, a \in \mathcal{S}\ :\ a = a \oplus \overline{0} \Leftrightarrow a \leqslant_{\oplus} \overline{0}$. □

In its basic form, a prebimonoid does not need to admit the dual concept; a BOTTOM element smaller than any other element. For example, we have already shown that the tropical semiring admits 0 as its additive ANNIHILATOR. This also has the consequence that 0 is the BOTTOM element with respect to to the natural order of the tropical semiring. Conversely, the natural dioid does not admit an ANNIHILATOR for $+$; ∞ would act as such an ANNIHILATOR if it were part of the underlying ground set.

Definition 4.1.5. *Let $(\mathcal{S}, \oplus, \otimes)$ be an* IDEMPOTENT *prebimonoid. An element β which is smaller than any element with respect to the natural order relation is called the* BOTTOM *element.*

$$\forall\, a \in \mathcal{S} \;:\; \beta \leqslant_{\oplus} a \tag{BOTTOM}$$

In some prebimonoids, the UNIT element acts as a BOTTOM element as well; as is the case for the element 0 in the tropical semiring but sometimes not; for example 1 in the natural dioid $(\mathbb{N}, +, \times)$. It is easy to incorporate a UNIT element that doubles as a BOTTOM element to a prebimonoid $(\mathcal{S}, \oplus, \otimes)$ in the event it is missing by defining $(\mathcal{S} \cup \{\overline{1}\}, \oplus_{\overline{1}}, \otimes_{\overline{1}})$ where the two new operations are given by

$$a \oplus_{\overline{1}} b = \begin{cases} a \oplus b & a \neq \overline{1} \neq b \\ \overline{1} & \text{otherwise} \end{cases}$$

$$a \otimes_{\overline{1}} b = \begin{cases} b & a = \overline{1} \\ a & b = \overline{1} \\ a \otimes b & \text{otherwise} \end{cases}$$

This constitutes one way of including a UNIT element into a structure to form a prebimonoid. In practice, the weakest restriction imposed on such element would be to act as a multiplicative IDENTITY. A more general approach to include a UNIT to produce a prebimonoid is to define the multiplicative law as above and the additive law as producing any function of a and b when either is the unit. Whenever a concept of convergence is involved, a special property is used in the process of proving that it effectively occurs. As we will show in the section on matrices, INFLATIONARITY is used to show that the least fixpoint to an equation is well-defined and can be reached by iterative approaches or local search algorithms.

Definition 4.1.6. *An* IDEMPOTENT *prebimonoid $(\mathcal{S}, \oplus, \otimes)$ is said to be* LEFT-INFLATIONARY *if its natural order satisfies*

$$\leqslant_{\oplus} -\text{INFLATIONARITY}: \quad \forall\, a, b \in \mathcal{S} \;:\; a \leqslant_{\oplus} a \otimes b \;\wedge\; b \leqslant_{\oplus} a \otimes b$$

It is easy to illustrate why INFLATIONARITY is important for path problems. Consider the graph from Figure 4.2 where the underlying distance structure is no longer grounded in $\mathbb{N}^{\infty}$ but in $\mathbb{Z}^{\infty}$; negative distances are included in the problem. It is quite straightforward to see how this affects INFLATIONARITY; we can find two weights such that their sum is lower than either of them, e.g. $2 + (-3) \leqslant 2$, whereas INFLATIONARITY requires that their sum is greater than either of them. This allows graphs in which there is no shortest path between some vertices, as is the case between vertex **2** and **5** on Figure 4.2. The direct path $2 \to 1 \to 5$ has a weight of **3**, but the path $\mathbf{2} \to \mathbf{1} \to \mathbf{3} \to \mathbf{2} \to \mathbf{1} \to \mathbf{5}$ has a weight of 1. It is possible to generate an infinite set of paths by successively traversing one more time the loop $\mathbf{1} \to \mathbf{3} \to \mathbf{2} \to \mathbf{1}$ which will decrease the weight of the resulting path further. One can traverse this loop an arbitrary number of time to shorten the distance between the source and the destination. A property closely related to INFLATIONARITY is known as the q-stability of elements.

Definition 4.1.7. *Given a prebimonoid $(\mathcal{S}, \oplus, \otimes)$, an element $a \in \mathcal{S}$ is said to be q-stable if*

$$\sum_{0 \leqslant k \leqslant q}^{\oplus} a^k = \sum_{0 \leqslant k \leqslant q+1}^{\oplus} a^k$$

The cycle that appears in Figure 4.2 has the particularity that its weight (-1) is that there is no q for which it is q-stable. For any given value of q, the $q + 1^{th}$ term will be obtained by decreasing the value of the q^{th} term by 1. In terms of paths, this simply means that traversing the cycle one more time will further reduce the length of any path towards any destination by 1, thus improving upon the current solution. This is what Gondran & Minoux call a 0-absorbing cycle [25] which is a cycle whose weight is not 0-stable. Under the assumption that there are no 0-absorbing cycles in a graph, the shortest paths are well-defined and can be obtained by an iterated approach. In particular, any graph over which the path problem is solved in the context of an INFLATIONARY prebimonoid is exempt of 0-absorbing cycles.

The concept of INFIMUM is very useful in the context of a multipath problem; whenever two paths are known towards some destination d with each of them having the same lowest value for the metric used, both should be kept instead of discarding one or the other.

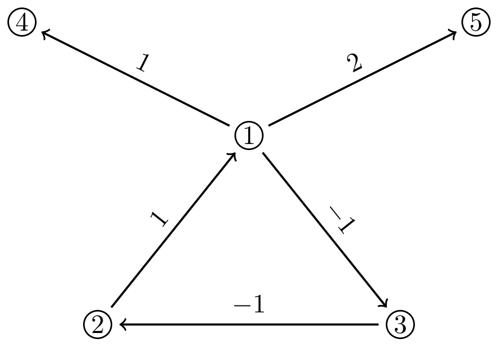


Figure 4.2: A simple graph in the context of the distance metric with negative values.

Lemma 4.1.4. *Let $(\mathcal{S},\oplus,\otimes)$ be an* IDEMPOTENT *prebimonoid. The* INFIMUM *(or greatest lower bound) of two elements a and b with respect to the natural order relation is given by their sum;*

$$\begin{array}{rl} \text{LOWER BOUND} : & a\oplus b \leqslant_\oplus a \ \wedge\ a\oplus b \leqslant_\oplus b \\ \text{GREATEST} : & \forall\, l \in \mathcal{S} : l \leqslant_\oplus a \ \wedge\ l \leqslant_\oplus b \ \Rightarrow\ l \leqslant_\oplus a\oplus b \end{array}$$

Proof. Deriving that addition gives a lower bound is straightforward. In the first case, $a\oplus b \leqslant_\oplus a$ is by Definition (4.1) the same as $a\oplus b = (a\oplus b)\oplus a$, which is the equation we must manage to derive

$$\begin{aligned} a\oplus b &= (a\oplus a)\oplus b && (\text{IDEMPOTENCY}) \\ &= a\oplus(a\oplus b) && (\text{ASSOCIATIVITY}) \\ &= (a\oplus b)\oplus a && (\text{COMMUTATIVITY}) \end{aligned}$$

The second case, $a\oplus b \leqslant_\oplus b$ is the same as $a\oplus b = (a\oplus b)\oplus b$, which we derive as follows

$$\begin{aligned} a\oplus b &= a\oplus(b\oplus b) && (\text{IDEMPOTENCY}) \\ &= (a\oplus b)\oplus b && (\text{ASSOCIATIVITY}) \end{aligned}$$

The fact that it is greater than any other lower bound l follows easily

$$\begin{aligned} l\leqslant_\oplus a \ \wedge\ l\leqslant_\oplus b &\Leftrightarrow\ l = l\oplus a \ \wedge\ l = l\oplus b \\ &\Rightarrow\ l\oplus l = (l\oplus a)\oplus(l\oplus b) \\ &\Leftrightarrow\ l\oplus l = l\oplus(a\oplus l)\oplus b && (\text{ASSOCIATIVITY}) \\ &\Leftrightarrow\ l\oplus l = l\oplus(l\oplus a)\oplus b && (\text{COMMUTATIVITY}) \\ &\Leftrightarrow\ l\oplus l = (l\oplus l)\oplus(a\oplus b) && (\text{ASSOCIATIVITY}) \\ &\Leftrightarrow\ l = l\oplus(a\oplus b) && (\text{IDEMPOTENCY}) \\ &\Leftrightarrow\ l\leqslant_\oplus a\oplus b \end{aligned}$$

which concludes the proof that $a\oplus b$ is the INFIMUM of a and b. □

The INFIMUM is different from either operand whenever they are mutually incomparable. The INFLATIONARITY and INFIMUM within IDEMPOTENT prebimonoids tell us that multiplication can only bring us higher in terms of metric values while addition brings us lower with respect to the natural order relation; the two laws progress in opposite directions.

$$a\oplus b \leqslant_\oplus a \leqslant_\oplus a\otimes b$$

Figure 4.3 summarizes the various concepts described so far. The implication relation between all of them are represented with directed arrows.

Due to the potential lack of totality of its natural order, an IDEMPOTENT prebimonoid may admit incomparable elements. In order to approach the sets of fixpoints to recurrence equations, we need to understand how these elements fit into the picture and how they work together with ordered elements. To this end, we define an incomparability relation related to the natural order. This definition allows us to contemplate a complete coverage of the ground set of an IDEMPOTENT prebimonoid. Formally, a binary relation on the set $\mathcal{S}$ is simply a subset of its cartesian product.

$$a \parallel_\oplus b \ \Leftrightarrow\ \neg(a\leqslant_\oplus b) \ \wedge\ \neg(b\leqslant_\oplus a)$$

$$\begin{tikzcd}
 & \boxed{\oplus - \text{ASSOCIATIVITY}} \arrow[r] \arrow[dl] & \boxed{\leqslant_{\oplus} - \text{TRANSITIVITY}} \\
\boxed{\begin{array}{l} a \oplus b \leqslant_{\oplus} a \\ a \oplus b \leqslant_{\oplus} b \end{array}} \arrow[dd] & & \\
 & \boxed{\oplus - \text{COMMUTATIVITY}} \arrow[r] \arrow[ul] & \boxed{\leqslant_{\oplus} - \text{ANTI-SYMMETRY}} \\
\boxed{a \oplus b = \text{INFIMUM}(a, b)} & & \\
 & \boxed{\oplus - \text{IDEMPOTENCY}} \arrow[r] \arrow[ul] & \boxed{\leqslant_{\oplus} - \text{REFLEXIVITY}} \\
 & \boxed{\oplus - \text{ZERO}} \arrow[r] & \boxed{\leqslant_{\oplus} - \text{TOP}} \\
 & \boxed{\oplus - \text{ANNIHILATOR}} \arrow[r] & \boxed{\leqslant_{\oplus} - \text{BOTTOM}}
\end{tikzcd}$$

Figure 4.3: Map of the properties of prebimonoids

The central object in the algebraic theory of routing are the least fixpoints to linear recurrence equations which are relating the elements a, b, l, r drawn from an IDEMPOTENT prebimonoid

$$r = ra \oplus b \tag{4.2}$$
$$l = al \oplus b \tag{4.3}$$

These two equations serve as specifications to what routing protocols are computing. As we will illustrate in the next chapter, they capture the two different approaches to solve routing problems; either by constructing a solution rooted at the vertex running an algorithm (link-state) or by combining the partial solutions from the neighbors (distance-vector). We now define the least fixpoints of the left equation in any IDEMPOTENT prebimonoid.

Definition 4.1.8. *Let $(\mathcal{S}, \oplus, \otimes)$ be an* IDEMPOTENT *prebimonoid satisfying* LEFT-DISTRIBUTIVITY. *Given a linear recurrence equation with $a, b, l \in \mathcal{S}$*

$$l = al \oplus b$$

We define the set of fixpoints

$$\varphi = \{l \in \mathcal{S} \mid l = al \oplus b\}$$

and the set of least fixpoints in two equivalent ways

$$\begin{aligned} \lambda &= \{l \in \varphi \mid \forall\, l' \in \varphi : l \leqslant_{\oplus} l' \lor l \parallel_{\oplus} l'\} \\ &= \{l \in \varphi \mid \forall\, l' \in \varphi : \neg(l' < l)\} \end{aligned}$$

We provide the following result which guarantees that whenever we have two fixpoints of a linear recurrence equation over an IDEMPOTENT prebimonoid, their INFIMUM is a fixpoint as well.

Theorem 4.1.1. *Let $(\mathcal{S}, \oplus, \otimes)$ be an* IDEMPOTENT *prebimonoid. Given a linear recurrence equation of the form*

$$l = (a \otimes l) \oplus b$$

and two fixpoints $l_1, l_2 \in \varphi$, their INFIMUM *is a fixpoint as well; $l_1 \oplus l_2 \in \varphi$.*

Proof. Suppose we have two fixpoints;

$$\begin{aligned} l_1 &= (a \otimes l_1) \oplus b \\ l_2 &= (a \otimes l_2) \oplus b \end{aligned}$$

We can sum both equations to obtain

$$\begin{aligned}
l_1 \oplus l_2 &= (a \otimes l_1) \oplus b \oplus (a \otimes l_2) \oplus b \\
&= (a \otimes l_1) \oplus (a \otimes l_2) \oplus b \oplus b && (\textsc{commutativity, associativity}) \\
&= (a \otimes l_1) \oplus (a \otimes l_2) \oplus b && (\textsc{idempotency}) \\
&= a \otimes (l_1 \oplus l_2) \oplus b && (\textsc{left-distributivity})
\end{aligned}$$

Therefore, $l_1 \oplus l_2 \in \lambda$. □

Theorem 4.1.1 can be complemented to claim the existence of a unique fixpoint to the left linear recurrence equation. A sufficient condition is that there is no infinite number of mutually incomparable elements. Assuming there is at most a finite number i of mutually incomparable elements, this implies that there are at most i mutually incomparable fixpoints. As a consequence, the sum of those mutually incomparable fixpoints is well-defined and coincides with their INFIMUM. Note that Definition (4.1.8) and Theorem 4.1.1 can also be described in the context of Equation (4.2).

4.2 Square matrices over idempotent prebimonoids

The IDEMPOTENT prebimonoid structure allows to represent the elementary metrics manipulated by routing protocols between two physical vertices. Consider the distance structure $(\mathbb{N}^{\infty}, \min, +)$ where the min operation is used to discriminate between two or more paths connecting the same source and destination. The identification of a solution to the problem of the shortest path will require a generalisation on the sources and destinations, which requires a further extension to express the interplay between n interconnected vertices forming a network, each having their own local connectivity and configurations with respect to the other vertices. The standard approach used for this transition is to lift a prebimonoid to square matrices over it; an operation that involves constructing a new ground set and defining an associated addition and multiplication.

Definition 4.2.1. *Let $(\mathcal{S}, \oplus, \otimes)$ be a prebimonoid. The set $\mathbb{M}_n(\mathcal{S})$ of square matrices of order n with elements drawn from $\mathcal{S}$ can be endowed with new additive and multiplicative laws, where $i, j \in \mathcal{V}$*

$$[\mathbf{X} \boxplus \mathbf{Y}]_{i,j} = \mathbf{X}_{i,j} \oplus \mathbf{Y}_{i,j} \tag{4.4}$$

$$[\mathbf{X} \boxtimes \mathbf{Y}]_{i,j} = \sum_{1 \leqslant q \leqslant n}^{\oplus} \mathbf{X}_{i,q} \otimes \mathbf{Y}_{q,j} \tag{4.5}$$

The ZERO *and* UNIT *matrices are defined by*

$$\mathbf{O}_{i,j} = \overline{0} \tag{4.6}$$

$$\mathbf{I}_{i,j} = \begin{cases} \overline{1} & i = j \\ \overline{0} & i \neq j \end{cases} \tag{4.7}$$

The graph interpretation of matrix multiplication is illustrated on Figure 4.4. A matrix encodes the weights of the paths connecting any pair vertices; their multiplication concatenates the paths between two vertices i and j by considering every possible intermediate vertex q. The weight of the path going through vertex q will be obtained by $\mathbf{X}_{sq} \otimes \mathbf{Y}_{qd}$ and the sum will be taken across all those possible paths.

Lemma 4.2.1. *If $(\mathcal{S}, \oplus, \otimes)$ is an* IDEMPOTENT *prebimonoid then so is $(\mathbb{M}_n(\mathcal{S}), \boxplus, \boxtimes)$.*

Proof. The properties for the additive law derive directly from the underlying addition due to the pointwise nature of the definition

- $\boxplus$ − COMMUTATIVITY

$$\begin{aligned}
[\mathbf{X} \boxplus \mathbf{Y}]_{i,j} &= \mathbf{X}_{i,j} \oplus \mathbf{Y}_{i,j} && (\text{Definition (4.4)}) \\
&= \mathbf{Y}_{i,j} \oplus \mathbf{X}_{i,j} && (\oplus - \textsc{commutativity}) \\
&= [\mathbf{Y} \boxplus \mathbf{X}]_{i,j} && (\text{Definition (4.4)})
\end{aligned}$$

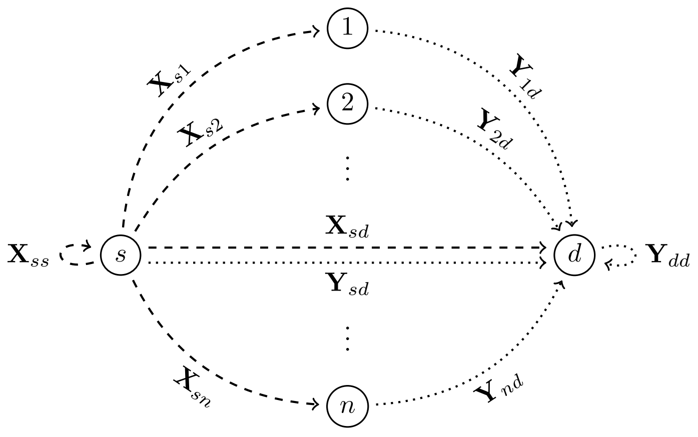


Figure 4.4: Graphical interpretation of $\mathbf{X} \boxtimes \mathbf{Y}$. The considered paths for every q are always the concatenation of a dashed path (corresponding to $\mathbf{X}_{sq}$) followed by a dotted path (corresponding to $\mathbf{Y}_{qd}$).

- $\boxplus$ – ASSOCIATIVITY

$$\begin{aligned}
[\mathbf{X} \boxplus (\mathbf{Y} \boxplus \mathbf{Z})]_{i,j} &= \mathbf{X}_{i,j} \oplus (\mathbf{Y}_{i,j} \oplus \mathbf{Z}_{i,j}) && \text{(Definition (4.4))}\\
&= (\mathbf{X}_{i,j} \oplus \mathbf{Y}_{i,j}) \oplus \mathbf{Z}_{i,j} && (\oplus - \text{ASSOCIATIVITY})\\
&= [(\mathbf{X} \boxplus \mathbf{Y}) \boxplus \mathbf{Z}]_{i,j} && \text{(Definition (4.4))}
\end{aligned}$$

- $\boxplus$ – ZERO

$$\begin{aligned}
[\mathbf{X} \boxplus \mathbf{O}]_{i,j} &= \mathbf{X}_{i,j} \oplus \bar{0} && \text{(Definition (4.4), Definition (4.6))}\\
&= \mathbf{X}_{i,j} && \text{(ZERO)}\\
&= \bar{0} \oplus \mathbf{X}_{i,j} && \text{(ZERO)}\\
&= [\mathbf{O} \boxplus \mathbf{X}]_{i,j} && \text{(Definition (4.4), Definition (4.6))}
\end{aligned}$$

The proofs for the properties of the multiplicative law are slightly more complex. The sum symbols always implicitly have $1 \leqslant q, q' \leqslant n$

- $\boxtimes$ – UNIT

$$[\mathbf{XI}]_{i,j} = \sum_{q}^{\oplus} \mathbf{X}_{i,q}\mathbf{I}_{q,j} = \sum_{q \neq j}^{\oplus} \mathbf{X}_{i,q}\bar{0} \oplus \sum_{q=j}^{\oplus} \mathbf{X}_{i,q}\bar{1} = \mathbf{X}_{i,j}$$

$$[\mathbf{IX}]_{i,j} = \sum_{q}^{\oplus} \mathbf{I}_{i,q}\mathbf{X}_{q,j} = \sum_{i \neq q}^{\oplus} \bar{0}\mathbf{X}_{q,j} \oplus \sum_{i=q}^{\oplus} \bar{1}\mathbf{X}_{q,j} = \mathbf{X}_{i,j}$$

- $\boxtimes$ – ANNIHILATOR

$$[\mathbf{XO}]_{i,j} = \sum_{q}^{\oplus} \mathbf{X}_{i,q}\bar{0} = \bar{0} = \sum_{q}^{\oplus} \bar{0}\mathbf{X}_{q,j} = [\mathbf{OX}]_{i,j}$$

□

4.2.1 Global optimum to a path problem

We now have all the elements to describe the problem of finding the optimal paths in a graph. In this section, we will relate the structure of the problem in the context of a labelled graph to two constructs from the algebraic path problem: the global optimum $\mathbf{A}^*$ and an iterative method rooted in a recurrence equation. In particular, we will assume that the underlying structure is a semiring. The main contributions that will be exposed in Chapter 6 will be focused on distance-vectoring methods, which is why we will look at these two constructs from the perspective of Equation (4.3).

Consider the context of an arbitrary labelled graph $\mathcal{G} = (\mathcal{V}, \mathcal{A}, \omega)$. The set $\mathcal{V}$ represents vertices which are connected together by means of arcs listed in $\mathcal{A}$, each of which is assigned a weight by means of a total function $\omega : \mathcal{A} \to \mathcal{S}$. It is possible to define the sets of paths between any two vertices in an inductive way; the set of paths connecting i to j made of exactly k arcs, $\mathcal{P}^k(i,j)$, is given by

$$\mathcal{P}^0(i,j) = \begin{cases} \varnothing & i \neq j \\ \{\epsilon\} & i = j \end{cases}$$
$$\mathcal{P}^k(i,j) = \{(i,q) \diamond \pi \mid (i,q) \in \mathcal{A} \ \wedge \ \pi \in \mathcal{P}^{k-1}(q,j)\} \qquad k > 0$$

where $\diamond$ concatenates a link to the left of a path. This definition gives us a basis to express the potentially infinite set of paths between two vertices;

$$\mathcal{P}^{\langle m \rangle}(i,j) = \bigcup_{k=0}^{m} \mathcal{P}^k(i,j) \tag{4.8}$$
$$\mathcal{P}(i,j) = \lim_{m\to\infty} \mathcal{P}^{\langle m \rangle}(i,j) \tag{4.9}$$

The weight function of the labelled graph, $\omega : \mathcal{A} \mapsto \mathcal{S}$, must be lifted from arcs to paths

$$\omega_L(\pi) = \begin{cases} \overline{1} & \pi = \epsilon \\ \omega(i,q) \otimes \omega_L(\pi') & \pi = (i,q) \diamond \pi' \end{cases} \tag{4.10}$$

Note that the product of the weight of successive links, which this function computes, is well defined if $\otimes$ is ASSOCIATIVE. Given that it is not case in a prebimonoid, we assume that the expressions associate on the right. In other words, the following equality holds for any path π containing $k \geqslant 1$ arcs.

$$\omega_L(\pi) = \omega(i,q_1) \otimes (\omega(q_1,q_2) \otimes \ldots (\omega(q_{k-2},q_{k-1}) \otimes \omega(q_{k-1},j)))$$

Given a labelled graph $\mathcal{G} = (\mathcal{V}, \mathcal{A}, \omega)$ and the corresponding underlying semiring $(\mathcal{S}, \oplus, \otimes)$, the objective of the path problem is to identify a matrix $\mathbf{A}^*$ such that

$$\mathbf{A}^*_{i,j} = \sum_{\pi \in \mathcal{P}(i,j)}^{\oplus} \omega_L(\pi) \tag{4.11}$$

It is clear from this definition that we are contemplating a formal definition of the problem to identify the globally optimal weight among all paths between any two vertices i and j since the sum is taken over all possible paths which can connect i to j. This definition can be related to a decomposition in terms of matrix powers. It is possible to approach from the algebraic perspective by considering the adjacency matrix as the starting point;

$$\mathbf{A}_{i,j} = \begin{cases} \omega(i,j) & (i,j) \in \mathcal{A} \\ \overline{0} & (i,j) \notin \mathcal{A} \end{cases}$$

We can define the power of a matrix and relate it to the terms appearing in Equation (4.9).

$$\mathbf{A}^0 = \mathbf{I} \tag{4.12}$$
$$\mathbf{A}^k = \mathbf{A} \boxtimes \mathbf{A}^{k-1} \quad , \quad k > 0 \tag{4.13}$$

Lemma 4.2.2. *The k-th power of an adjacency matrix $\mathbf{A}$ contains, for every pair i,j, the weight of a path containing k arcs which is globally optimal among all paths made of k arcs.*

$$\mathbf{A}^k(i,j) = \sum_{\pi \in \mathcal{P}^k(i,j)}^{\oplus} \omega_L(\pi)$$

Proof. By induction.

- $k = 0$.

$$\mathbf{A}^0(i,j) = \mathbf{I}(i,j) = \left\{ \begin{array}{ll} \overline{0} = \sum^{\oplus}_{\pi\in\varnothing} \omega_L(\pi) & i \neq j \\ \overline{1} = \sum^{\oplus}_{\pi\in\{\epsilon\}} \omega_L(\pi) & i = j \end{array} \right\} = \sum_{\pi \in \mathcal{P}^0(i,j)}^{\oplus} \omega_L(\pi)$$

- $k > 0$. The induction hypothesis is given for $k-1$

$$\mathbf{A}^{k-1}(i,j) = \sum_{\pi \in \mathcal{P}^{k-1}(i,j)}^{\oplus} \omega_L(\pi) \tag{IH}$$

The inductive case follows quite easily

$$\begin{aligned}
\mathbf{A}^{k}(i,j) &= [\mathbf{A} \boxtimes \mathbf{A}^{k-1}](i,j) && (4.13)\\
&= \sum_{1 \leqslant q \leqslant n}^{\oplus} \mathbf{A}(i,q) \otimes \mathbf{A}^{k-1}(q,j) \\
&= \sum_{1 \leqslant q \leqslant n}^{\oplus} \mathbf{A}(i,q) \otimes \sum_{\pi \in \mathcal{P}^{k-1}(q,j)}^{\oplus} \omega_L(\pi) && \text{(IH)}\\
&= \sum_{1 \leqslant q \leqslant n}^{\oplus} \sum_{\pi \in \mathcal{P}^{k-1}(q,j)}^{\oplus} \mathbf{A}(i,q) \otimes \omega_L(\pi) && (\text{LEFT-DISTRIBUTIVITY})\\
&= \sum_{\pi \in \mathcal{P}^{k}(i,j)}^{\oplus} \omega_L(\pi)
\end{aligned}$$

□

By virtue of this Lemma, taken together with (4.9) and (4.11), we can give an alternative definition of $\mathbf{A}^*$

$$\mathbf{A}^* = \sum_{0 \leqslant k}^{\boxplus} \mathbf{A}^k \tag{4.14}$$

Note that the sum on the right hand side of Definition (4.11) or Definition (4.14) could be not well-defined due to three possible reasons. The lack of $\oplus -$ ASSOCIATIVITY would mean that given a certain order in which the sum is resolved, the resulting value could be different. Given a sum of n terms, $\oplus -$ ASSOCIATIVITY guarantees that parentheses can be used in any way and the sum would yield always the same value. The lack of $\oplus -$ COMMUTATIVITY would require specifying the order in which the terms appear. Finally, the absence of INFLATIONARITY (negative weights in the classical shortest-path problem are inverses for standard addition) could have as a consequence that the successive powers of $\mathbf{A}$ are decreasing without a lower bound on them, thus always changing as k increases. Other alternatives to INFLATIONARITY have been studied in the litterature [9, 24] and revolved around the absence of cycles in the graphs whose weight would improve the current solution between two vertices at each additional traversal of the cycle.

The question of whether $\mathbf{A}^*$ is well-defined or degenerate is related to the question of the existence of a globally optimal solution to the problem at hand. Various conditions for a positive answer to this question can be found in the classical litterature (see [25]). We only give a sketch of the argument for the q-stability condition.

Lemma 4.2.3. *Every square matrix of order n over a prebimonoid $(\mathcal{S}, \oplus, \otimes)$ is $n-1$-stable if every element from the prebimonoid is $0-$ stable.*

$$\forall\, a \in \mathcal{S} \;:\; a^0 = a^0 \oplus a^1 \;\Rightarrow\; \forall\, \mathbf{A} \in \mathbb{M}_n(\mathcal{S}) \;:\; \sum_{0 \leqslant k \leqslant n-1}^{\boxplus} \mathbf{A}^k = \sum_{0 \leqslant k \leqslant n}^{\boxplus} \mathbf{A}^k$$

The 0-stability is a more general property than INFLATIONARITY which we assumed earlier. In particular we have $\overline{1} \leqslant_{\oplus} \overline{1} \otimes a \;\Leftrightarrow\; \overline{1} \leqslant_{\oplus} a$ which by definition gives us $\overline{1} = \overline{1} \oplus a$. The idea is that the paths containing more than n arcs necessarily contain loops and removing those loops results in paths with a more interesting value of the metric in terms of the natural order as well as a lower number of arcs. In essence, for any path with $k \geqslant n$ arcs, we can decompose it to a path with $k < n$ arcs to which a loop is attached. If l represents the weight of the loop and p_1, p_2 are the weights of the two subpaths before and after the vertex at which the loop is attached, we can deduce from INFLATIONARITY that the weight of path with the loop has a higher weight that the path alone

$$\begin{aligned} p_2 \leqslant_\oplus l \otimes p_2 &\Leftrightarrow p_2 = p_2 \oplus (l \otimes p_2) \\ &\Rightarrow p_1 \otimes p_2 = p_1 \otimes (p_2 \oplus (l \otimes p_2)) \\ &\Leftrightarrow p_1 \otimes p_2 = (p_1 \otimes p_2) \oplus (p_1 \otimes l \otimes p_2) && (\textsc{left-distributivity}) \\ &\Leftrightarrow p_1 \otimes p_2 \leqslant_\oplus p_1 \otimes l \otimes p_2 \end{aligned}$$

This means that any term $(p_1 \otimes l \otimes p_2)$ from Definition (4.11) incorporated by a weight with more than n arcs will have a corresponding term $(p_1 \otimes p_2)$ with a more interesting weight. If we consider Definition (4.14) as a method to compute $\mathbf{A}^*$ by successively adding the powers of the adjacency matrix until the cumulative sum stabilizes, once we reach $n-1$, we are certain that no further power of $\mathbf{A}$ will contain for any i, j a value more interesting that the one currently known. Considering the case where $\mathbf{A}^*$ is well-defined due to $n-1$-stability,

$$\mathbf{A}^* = \sum_{0 \leqslant k \leqslant n-1}^{\boxplus} \mathbf{A}^k \tag{4.15}$$

It is easy to show that this matrix is a solution to Equation (4.3)

$$\begin{aligned} \mathbf{A}^* &= \mathbf{A}\mathbf{A}^* \boxplus \mathbf{I} \\ \sum_{0 \leqslant k \leqslant n-1}^{\boxplus} \mathbf{A}^k &= \mathbf{A} \boxtimes \sum_{0 \leqslant k \leqslant n-1}^{\boxplus} \mathbf{A}^k \boxplus \mathbf{I} && (\text{Equation (4.15)}) \\ \sum_{0 \leqslant k \leqslant n-1}^{\boxplus} \mathbf{A}^k &= \sum_{0 \leqslant k \leqslant n-1}^{\boxplus} \left(\mathbf{A} \boxtimes \mathbf{A}^k\right) \boxplus \mathbf{I} && (\textsc{left-distributivity}) \\ \sum_{0 \leqslant k \leqslant n-1}^{\boxplus} \mathbf{A}^k &= \sum_{0 \leqslant k \leqslant n-1}^{\boxplus} \mathbf{A}^{k+1} \boxplus \mathbf{I} && (\text{By Definition (4.13)}) \\ \sum_{0 \leqslant k \leqslant n-1}^{\boxplus} \mathbf{A}^k &= \sum_{0 \leqslant k \leqslant n}^{\boxplus} \mathbf{A}^k && (\mathbf{I} = \mathbf{A}^0) \end{aligned}$$

where the last equality is $n-1$-stability.

4.2.2 A simple iterative method

It is possible to derive a simple algorithm from the recurrence equation we are trying to solve. The method proceeds from an initial matrix $\mathbf{L}^{\langle 0 \rangle}$ and works by successive applications of the equation until a fixpoint is reached; that is, until we reach a k such that $\mathbf{L}^{\langle k \rangle} = \mathbf{L}^{\langle k-1 \rangle}$.

$$\begin{aligned} \mathbf{L}^{\langle 0 \rangle} &= \mathbf{I} \\ \mathbf{L}^{\langle k \rangle} &= \mathbf{A}\mathbf{L}^{\langle k-1 \rangle} \oplus \mathbf{I} && (k \geqslant 1) \end{aligned}$$

It is possible to relate the sequence of $\mathbf{L}^{\langle k \rangle}$ to the sequence of terms appearing in (4.9). The argument works by induction, starting with $k = 0$

$$\mathbf{L}_{i,j}^{\langle 0 \rangle} = \sum_{\pi \in \mathcal{P}^{\langle 0 \rangle}(i,j)}^{\oplus} \omega_L(\pi)$$

When $i \neq j$, there are no paths connecting i to j made of 0 links. The value of the entire sum is equal to the zero. On the other hand, when $i = j$, the one and only path connecting i to itself made of 0 links is the self-loop, for which the weight is the unit value. In either case, $\mathbf{L}_{i,j}^{\langle 0 \rangle} = \mathbf{I}_{i,j}$. The induction hypothesis is given by

$$\mathbf{L}_{i,j}^{\langle k-1 \rangle} = \sum_{\pi \in \mathcal{P}^{\langle k-1 \rangle}(i,j)}^{\oplus} \omega_L(\pi)$$

and the inductive case follows naturally

$$\begin{aligned}
\mathbf{L}^{\langle k\rangle}_{i,j} &= \sum^{\oplus}_{q} \mathbf{A}_{i,q}\mathbf{L}^{\langle k-1\rangle}_{q,j} \oplus \mathbf{I}_{i,j} \\
&= \sum^{\oplus}_{q} \mathbf{A}_{i,q} \otimes \sum^{\oplus}_{\pi\in\mathcal{P}^{\langle k-1\rangle}(q,j)} \omega_L(\pi) \oplus \mathbf{I}_{i,j} && \text{(IH)} \\
&= \sum^{\oplus}_{\pi\in\mathcal{P}^{\langle k\rangle}(i,j)} \omega_L(\pi) && \text{(LEFT-DISTRIBUTIVITY)}
\end{aligned}$$

Each application of the equation to the working solution adds another term, thus bringing us closer to the value of $\mathbf{A}^*$. Gurney showed [28] that this iterative method converges to a unique fixpoint under extremely loose assumptions, parting with LEFT-DISTRIBUTIVITY but requiring INFLATIONARITY.

The existence of a 0-absorbing cycle in the graph would mean that there is an infinite number of paths with decreasing weight present in P(i,j) preventing the convergence of the sequence of $\mathbf{L}^{\langle k\rangle}$.

4.3 Order relations and minimum operation

We give in this small section some properties related to constructs involving order relations and the minimum operation which can be applied on sets. A more thorough treatment can be found in [12] and [48]. Two important types of relations are preorders and orders.

Definition 4.3.1. *A preorder relation $\lesssim$ on the elements of a set X is a relation satisfying the properties;*

$$\begin{aligned}
\text{REFLEXIVITY}: &\quad \forall\, x \in X : x \lesssim x \\
\text{TRANSITIVITY}: &\quad \forall\, x,y,z \in X : x \lesssim y \wedge y \lesssim z \Rightarrow x \lesssim z
\end{aligned}$$

Definition 4.3.2. *An order relation $\leqslant$ on the elements of a set X is a preorder relation satisfying*

$$\text{ANTI-SYMMETRY}: \quad \forall\, x,y \in X : x \leqslant y \wedge y \leqslant x \Rightarrow x = y$$

Note that any preorder or order relation contains a strict version which can be obtained by considering the case where $x \lesssim y \wedge x \neq y$ and $x \leqslant y \wedge x \neq y$ respectively. Preorder and order relations can be total, relating all pairs of elements of X in one way or the other, and can admit special elements called bottom and top.

Definition 4.3.3. *A total preorder (resp. order) relation $\lesssim$ on the elements of a set X is a preorder (resp. order) which satisfies*

$$\text{TOTALITY}: \quad \forall\, x,y \in X : x \lesssim y \vee y \lesssim x$$

Definition 4.3.4. *A preorder relation $\lesssim$ on the elements of a set X can admit a bottom element or a top element*

$$\begin{aligned}
\text{BOTTOM}: &\quad \exists\, \bot \in X : \forall\, x \in X : \bot \lesssim x \\
\text{TOP}: &\quad \exists\, \top \in X : \forall\, x \in X : x \lesssim \top
\end{aligned}$$

Note that TOTALITY and the existence of bottom and top elements can be used for order relations as well. Consider the set of lists of elements drawn from a set, list(X). It is possible to construct a total preorder based on the number of elements appearing in lists

$$l_1 \lesssim_{\text{length}} l_2 \Leftrightarrow \text{length}(l_1) \leqslant \text{length}(l_2)$$

The relation $\lesssim_{\text{length}}$ does not satisfy ANTI-SYMMETRY as can be shown from the following example for lists of natural numbers, list$(\mathbb{N})$

$$[1,2,4] \lesssim_{\text{length}} [1,3,4] \wedge [1,3,4] \lesssim_{\text{length}} [1,2,4] \wedge [1,2,4] \neq [1,3,4]$$

On the other hand, the relation $\leqslant$ on natural numbers is an order relation given that

$$\forall\, n_1, n_2 \in \mathbb{N} : n_1 \leqslant n_2 \wedge n_2 \leqslant n_1 \Rightarrow n_1 = n_2$$

Given a preorder relation, we can apply an operation on sets of elements which provides the minimal elements it contains.

$$\min_{\lesssim}(P) \triangleq \{x \in P \mid \forall\, x' \in P : \neg(x' < x)\} \tag{4.16}$$

Some of the useful properties that operation 4.16 satisfies are

$$\min_{\lesssim}(P) = \min_{\lesssim}\left(\min_{\lesssim}(P)\right) \tag{4.17}$$

$$P = \min_{\lesssim}(P) \Rightarrow \min_{\lesssim}\left(\min_{\lesssim}(P) \cup Q\right) = \min_{\lesssim}(P \cup Q) \tag{4.18}$$

$$x \in \min_{\lesssim}(P) \Leftrightarrow (\exists i : x \in X_i) \wedge \big(\forall i : \forall\, x' \in X_i : \neg(x' < x)\big) \tag{4.19}$$

$$x \notin \min_{\lesssim}(P) \Leftrightarrow (\forall i : x \notin X_i) \vee \big(\exists i : \exists\, x' \in X_i : x' < x\big) \tag{4.20}$$

Given two preorders $\lesssim_x$ and $\lesssim_y$ respectively on the sets X and Y, it is possible to define their lexicographic product, denoted by $\lesssim_x \vec{\times} \lesssim_y$ which is a preorder on the set $X_1 \times X_2$ given by

$$(x_1, y_1)\ [\lesssim_x \vec{\times} \lesssim_y]\ (x_2, y_2) \equiv x_1 <_x x_2 \vee (x_1 \sim_x x_2 \wedge y_1 \lesssim_y y_2)$$

If the two preorders admit top elements, $\top_x$ and $\top_y$, their lexicographic product admits $(\top_x, \top_y)$ as its top element. In the same way, if the preorders admit bottom elements $\bot_x$ and $\bot_y$, then $(\bot_x, \bot_y)$ is the bottom element of their lexicographic product.

Chapter 5

Routing Algorithms

Perfection is achieved, not when there is nothing more to add, but when there is nothing left to take away.

Antoine de Saint-Exupéry.

Ever since it was taken over by industrial and commercial actors, the Internet has undergone an incredible growth. Much effort in this context was put on inter-operability; connecting together networks which used independently developed technologies to attain the size that we know today. While this emphasis on compatibility has allowed it to increase connectivity and reachability between customers, it has done so at the cost of bringing a wide diversity of problems upon the shoulders of network administrators. The main objective can be stated as that of keeping everyone connected together at all costs. In the light of ongoing research on the issues related to stabilisation of routes, forwarding loops and a need for greater modularity, the community has been working on solving these problems by considering them from a purely algorithmic perspective.

Metarouting [27] has parted with this approach by embracing Dijkstra's separation of concerns. While acknowledging the importance of understanding the problems originating in the algorithms and their implementations running on actual routers, it has plucked the metrics out of the picture and seeks to study them as a separate concern. This choice has allowed to shed light on how metrics play a role in inducing pathological situations that can cripple interconnected domains by reasoning in the context of a simple algebraic framework. In this context, the recurrence equations presented in Chapter 4 serve as specifications to the algorithms in use today and simplify greatly their analysis.

Further efforts strived to study other aspects with the consequences of enriching the corpus of constructs available to model real-world protocols. In [27], Griffin and Sobrinho used routing algebras [53] as the basis to model routing metrics along with various composition products, enabling the expression of complex metrics in terms of simpler ones. Among those products was the lexicographic product, which enables to rely on a secondary metric when a primary metric leads to a tie between two paths. This construct was studied in depth by Gurney [29] who gave necessary and sufficient conditions for the preservation of distributivity of a lexicographic product. This property turned out to be central for the optimality of the produced solution.

It is quite common in the structure of the Internet to separate internal from the external protocols; a domain can be running an IGP to establish routes traversing its infrastructure while running an EGP at its border routers to interact with other domains. The resulting routes are the composition of successive internal and external portions, both being constructed through the use of different metrics and protocols. The scoped product attempts to capture this idea and Gurney showed how it is in fact expressible in terms of a lexicographic product. In [8], Billings and Griffin explored the interplay of routing and forwarding by looking at how destinations are attached to an infrastructure network. By using semi-modules [25], they illustrate mecanisms involved in forwarding such as route redistribution or administrative distance.

An aspect which had been left out of the picture were the paths themselves. The focus of the algebraic approach until recently was solely on the computation of the weight of the shortest-path as opposed to that of the shortest-path. In [19], the paths were made explicit by the use of the Algorithm-to-Algebra method. This enabled to make explicit the conditions under which an Equal-cost Multi-path (ECMP) can be computed exhaustively and when both a link-state approach and distance-vectoring method agree on the resulting paths.

In this chapter, we relate two classes of algorithms to the linear recurrence equations described in the previous chapter by proving that two instances of algorithms are computing fixpoints to the equations under an appropriate set of restrictions on the underlying algebraic structure. We provide a new, shorter and more general proof that Dijkstra's Algorithm is computing a fixpoint to a right equation requiring only for the additive law of the underlying structure to be SELECTIVE, ASSOCIATIVE, COMMUTATIVE and an input matrix which is INFLATIONARY. Secondly, we will prove the convergence of Bellman-Ford's Algorithm under the assumption that the additive law is IDEMPOTENT, ASSOCIATIVE, COMMUTATIVE and an input matrix which is LEFT-INFLATIONARY.

We will start by looking at two of the standard algorithms used in routers nowadays; Dijkstra's Algorithm for the link-state class [40, 42] and Distributed Bellman-Ford for distance/path vectoring methods [37, 47]. While the literature on the topic is quite rich in terms of other algorithms, the choice to limit ourselves to these two stems from two considerations. First, these are the most commonly used in actual implementations running in the real world. As such, not only are they the ones anyone who had an elementary contact with routing in networking is familiar with but also they are the easiest to understand. Secondly, they illustrate quite well the elementary mechanism that enables an algorithm to compute a routing solution given adjacency information. The emphasis is mostly put on their mathematical specification which capture the way the problem is decomposed in order to solve it. The key aspect being studied here is the contrast between the *link-state* approach; building a solution rooted at each vertex of the network and using this information in the forwarding process, with the *distance-vector* approach; importing solutions from one's neighbors and combining them together to produce the forwarding information. Our understanding of the problem at hand would not benefit much from considering more complex or optimized versions which would merely clutter the analysis with irrelevant aspects. The ideas presented in this chapter are mostly drawn from [27, 54] with the exception of our shorter proof of Dijkstra's Algorithm.

5.1 Link-state algorithms

The first class of routing algorithm is the link-state approach. It can be roughly decomposed into two separate phases; link-state flooding and effective computation. During the flooding, all vertices build their own copy of the adjacency information of the entire network in which the algorithm is being executed. This is achieved by having each vertex i broadcast its relevant row of the original adjacency matrix $\mathbf{A}$. This process enables every vertex to compute its solution independently of all the others. The one algorithm in use today is Dijkstra's Algorithm, which ranks among the most beautiful ever discovered. By relying on a total ordering of the vertices based on their distance from the source at one point during its execution, it allows to compute the shortest paths towards all destinations in the most efficient way known so far.

Over the decades, this algorithm has been the object of many application which range from integrated circuit layout optimisation [44] to Location-Based Services [14]. It has also been subject to several generalizations for the computation of the k-shortest paths [61][20], context-free grammar [33] or general metrics in routing [54].

The central operation on which Algorithm 2 is built is called the relaxation step of Line (10) and always involves three vertices; a source, a destination and a relaxation. In Dijkstra's Algorithm, the source is implicit given that it is the vertex running the algorithm which has its own local ω vector. The set $\mathcal{R}$ describes the vertices which have already been relaxed (picked as q in a previous iteration). The relaxation works by comparing two paths towards a destination d; the best one currently known ($\omega[d]$) against the one obtained by first going to the relaxation vertex ($\omega[q]$) followed by using the link connecting q to d ($\mathbf{A}_{q,d}$). The INFLATIONARITY of the metric involved implies that picking as a relaxation vertex one of the closest vertices will necessarily result in the stabilisation of at least one vertex on its final distance value, something which is not necessarily true in the case of the choice of a non-closest vertex. It can be shown that at one iteration, the minimal distance among the unrelaxed vertices is greater than

Algorithm 2 Dijkstra's Algorithm for the classical shortest-path problem

```
1: for all $d \in \mathcal{V}$ do
2:     $\omega[d] \leftarrow \infty$
3: end for
4: $\omega[s] \leftarrow 0$
5: $\mathcal{R} \leftarrow \varnothing$
6: while $\mathcal{R} \neq \mathcal{V}$ do
7:     pick $q \in \mathcal{V} - \mathcal{R}$ such that $\forall\, q' \in \mathcal{V} - \mathcal{R} : \omega[q] \leqslant \omega[q']$
8:     $\mathcal{R} \leftarrow \mathcal{R} \cup \{q\}$
9:     for all $d \in \mathcal{V} - \mathcal{R}$ do
10:        $\omega[d] \leftarrow \omega[d] \ \min\ (\omega[q] + \mathbf{A}_{q,d})$
11:    end for
12: end while
```

the maximal distance among the relaxed vertices, which by INFLATIONARITY implies that no extension of a path going through an unrelaxed vertex can surpass the known path to a relaxed vertex

$$\forall\, d \in \mathcal{R} : \forall\, q \in \mathcal{V} - \mathcal{R} : \omega[d] \leqslant \omega[q] \qquad \text{(see Lemma 5.1.2)}$$

a fact which justifies the reason for not considering relaxed destinations on Line (9) given that none of them would have their distance reduced by passing through any unrelaxed vertex $q \in \mathcal{V} - \mathcal{R}$. It is possible to partition the set $\mathcal{V}$ into three sets according to whether a vertex has already been relaxed and otherwise whether it is a candidate for relaxation or not. We consider the following partition of unrelaxed vertices

$$\mathcal{C} = \min_{\leqslant} (\mathcal{V} - \mathcal{R}) \qquad \text{(Candidates)}$$

$$\mathcal{U} = \mathcal{V} - \mathcal{R} - \mathcal{C} \qquad \text{(Non-candidates)}$$

There is an ordering leveraged by Algorithm 2 which relates all the vertices depending on their belonging to one of those sets as follows

$$\forall\, r \in \mathcal{R} : \forall\, c \in \mathcal{C} : \forall\, u \in \mathcal{U} : \omega[r] \leqslant \omega[c] \leqslant \omega[u] \tag{5.1}$$

Every iteration of Algorithm 2 has the consequence that the relaxation vertex q is transferred from $\mathcal{C}$ to $\mathcal{R}$, thus increasing the size of the set of relaxed vertices $\mathcal{R}$ by exactly one and reducing that of the set of unrelaxed vertices by one. This can also be represented in a graphical way (see Figure 5.1). The dotted lines represent the corresponding components from the working solution ω at one point for every vertex except the source and the one under relaxation. The component from ω is extended by the arc connecting to every possible d and matched against the d^{th} component of ω.

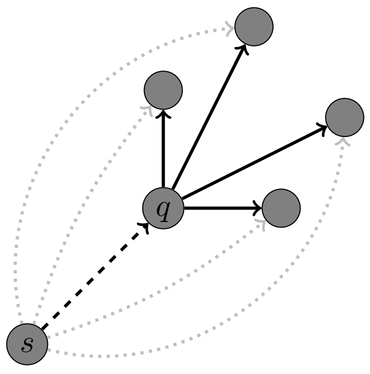


Figure 5.1: Graphical representation of the relaxation step. For a given destination d and weight vector ω residing at source s, relaxation by vertex q updates the weight vector with $\omega[d] := \omega[d] \ \min\ (\omega[q] + \mathbf{A}_{q,d})$.

The cleverness of Dijkstra's Algorithm resides in the use of the ordering of candidate vertices based on their current distance from the source. For a given destination, it is not necessary to apply a relaxation considering each and every intermediary vertex, picking vertices by increasing distance for relaxation will yield the correct solution. This is, in essence, the optimisation that allows to obtain Dijkstra's algorithm from Bellman-Ford's, the latter applying relaxation for every destination and intermediary vertex. In the next section, we will prove that Dijkstra's Algorithm is computing a fixpoint to Equation (5.2). From an algebraic perspective, what we will prove is that Algorithm 2 computes the s^{th} row of a fixpoint to the right equation

$$\mathbf{R} = \mathbf{R}\mathbf{A} \boxplus \mathbf{I} \tag{5.2}$$

over a structure subsuming the tropical semiring $(\mathbb{N}^{\infty}, \min, +)$. In this context, Equation 5.2 unfolds into a system of equations of the following form, where $d \in \mathcal{V}$

$$\mathbf{R}(s,d) = \min_{q \in \mathcal{V}}\{\mathbf{R}(s,q) + \mathbf{A}(q,d)\} \ \min \ \mathbf{I}(s,d)$$

which can be further rewritten by considering the definition of the unit matrix, given that 0 (resp. ∞) is the minimum (resp. maximum) distance

$$\mathbf{R}(s,d) = \begin{cases} 0 & s = d \\ \min_{q \in \mathcal{V}}\{\mathbf{R}(s,q) + \mathbf{A}(q,d)\} & s \neq d \end{cases}$$

This system is the dual of the classical Bellman Equations describing the shortest-paths solution that the Bellman-Ford algorithm produces given an input graph. In general, it is possible to prove that Dijkstra produces a least fixpoint over SELECTIVE INFLATIONARY semirings. The adjacency matrix, $\mathbf{A}$, encodes the graph with all elements on the diagonal being equal to ∞ (denoting the absence of self-loops) and no 0 anywhere else while $\mathbf{I}$ is the unit matrix. By laying out the block version of the right equation, we can relate it to two other aspects; firstly, every vertex s running the algorithm computes its relevant knowledge of $\mathbf{R}$ limited to the s^{th} row vector; only its working version, ω, needs to be stored. Secondly, the checking that one row vector of $\mathbf{R}$ is part of a fixpoint requires the entire matrix $\mathbf{A}$ to be known. In terms of routing, this requirement is fulfilled by the initial link-state flooding phase. The very same requirement applies for the relaxation step and appears clearly from Line (10) where q and d range over the entire set of vertices $\mathcal{V}$ for indexing $\mathbf{A}$ throughout the computation.

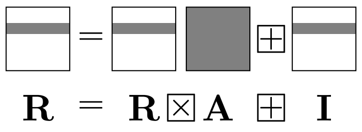


Figure 5.2: Block version of Equation (5.2).

One can use a weakened form of semirings to model metrics that can be used to label a graph and express the fixpoint problem that is being solved. At its basis, a metric is made up of three components. The first component, a ground set, contains all the possible values among which the computation is achieved; these are the ones used to configure the weights of links in a network and which also apply to paths. It is customary to have a special value corresponding to the weight of an invalid path. In the case of the `distance` metric, the value ∞ serves this purpose.

The second component, a summarization rule, combines together the weights of two paths from a particular source to a particular destination. This combination typically takes the form of a selection between one of the two paths, with the selection occurring based on the weight of the paths. Note that a special value $\overline{0}$ is used to denote the least interesting or invalid path. Line (2) of Algorithm 2 shows that this element denotes the initial value towards any destination but the source. This BOTTOM element represents the wisest choice given it denotes the most pessimistic view of the connectivity towards any destination. This can be relaxed in the event we are interested in bounding the value of the solution obtained by a least interesting weight between two vertices.

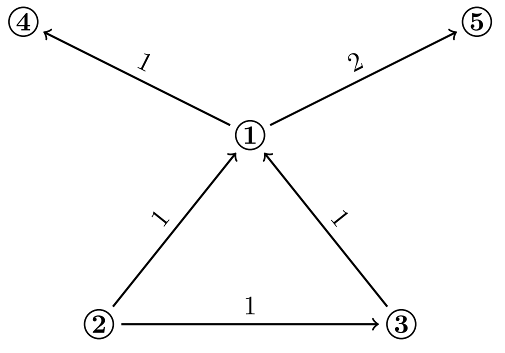
The third component, a concatenation rule, establishes the way by which the weight of a path is obtained from the weights of its constituent links. The weight of the individual links are multiplied together in the order in which they appear in the path they form. Consider the graph on Figure 5.4 where some arcs are missing, thus denoting the absence of direct connectivity between the corresponding vertices. In order to express this in the algebraic formalism, the element from the metric that acts as a $\otimes -$ ANNIHILATOR is used in the adjacency matrix. In the case of the `distance` metric, the element ∞ serves this purpose. Whenever a link is missing, an ∞ is placed in the matrix. Any sequence of vertices that involves traversing an arc with ∞ as its weight will produce a path with an infinite weight, due to its annihilating nature. Given that ∞ doubles as the BOTTOM element for the structure, it will turn the

name	$\mathcal{S}$	$\oplus$	$\otimes$	$\overline{0}$	$\overline{1}$
`distance`	$\mathbb{N}^\infty$	min	$+$	∞	0
`bandwidth`	$\mathbb{N}^\infty$	max	min	0	∞
`reliability`	[0, 1]	max	$\times$	0	1
`reachability`	$\{0,1\}$	$\vee$	$\wedge$	0	1

Figure 5.3: Simple metrics

weight of any invalid path (i.e. traversing a missing arc) into ∞. We list in Table 5.3 a few elementary metrics that can be used to label graphs in the format of semirings.

As an illustration, we present two graphs, labelled over `distance` and `bandwidth`, on Figure 5.4 and Figure 5.6.



$$\mathbf{A} = \begin{array}{c|ccccc} & \mathbf{1} & \mathbf{2} & \mathbf{3} & \mathbf{4} & \mathbf{5} \\ \hline \mathbf{1} & \infty & \infty & \infty & 1 & 2 \\ \mathbf{2} & 1 & \infty & 1 & \infty & \infty \\ \mathbf{3} & 1 & \infty & \infty & \infty & \infty \\ \mathbf{4} & \infty & \infty & \infty & \infty & \infty \\ \mathbf{5} & \infty & \infty & \infty & \infty & \infty \end{array}$$

Figure 5.4: Graph encoded over `distance`.

In the case of the `distance` metric, with vertex **2** considered as the source running Dijkstra's Algorithm, the successive ω vectors are given below along with the specific q that was picked for relaxation to obtain the next value of ω. In other words, each line gives ω_k the vector at the end of the k^{th} iteration and the vertex q that was selected at Line (7) during that same iteration. The initial ω vector, with the corresponding destinations labelled above it, for vertex **2** is the following

$$\omega_0 = \begin{array}{c|ccccc} & \mathbf{1} & \mathbf{2} & \mathbf{3} & \mathbf{4} & \mathbf{5} \\ \mathbf{2} & \infty & 0 & \infty & \infty & \infty \end{array}$$

and the sequence of values it takes as Algorithm 2 progresses

$$\begin{array}{lcl} \omega_0 & = & [\ \infty\ 0\ \infty\ \infty\ \infty\ ] \\ \omega_1 & = & [\ 1\ 0\ 1\ \infty\ \infty\ ]\ q = \mathbf{2} \\ \omega_2 & = & [\ 1\ 0\ 1\ 2\ 3\ ]\ q = \mathbf{1} \\ \omega_3 & = & [\ 1\ 0\ 1\ 2\ 3\ ]\ q = \mathbf{3} \\ \omega_4 & = & [\ 1\ 0\ 1\ 2\ 3\ ]\ q = \mathbf{4} \\ \omega_5 & = & [\ 1\ 0\ 1\ 2\ 3\ ]\ q = \mathbf{5} \end{array}$$

Note that at iteration 2, there are two equally distant candidates from the source; vertex **1** and **3**, both with a distance of 1. The alternative sequence of values of ω in the case where **3** is selected instead of **1** is

$$\begin{array}{lcl} \omega_0 & = & [\ \infty\ 0\ \infty\ \infty\ \infty\ ] \\ \omega_1 & = & [\ 1\ 0\ 1\ \infty\ \infty\ ]\ q = \mathbf{2} \\ \omega_{2'} & = & [\ 1\ 0\ 1\ \infty\ \infty\ ]\ q = \mathbf{3} \\ \omega_3 & = & [\ 1\ 0\ 1\ 2\ 3\ ]\ q = \mathbf{1} \\ \omega_4 & = & [\ 1\ 0\ 1\ 2\ 3\ ]\ q = \mathbf{4} \\ \omega_5 & = & [\ 1\ 0\ 1\ 2\ 3\ ]\ q = \mathbf{5} \end{array}$$

We can see that the solution is always decreasing; the current value of each components in the vector ω is always smaller than its counterpart in the previous vector. Note that those entries do not indicate anything about the sequence of vertices that produced it; another mecanism is needed to identify the shortest path.

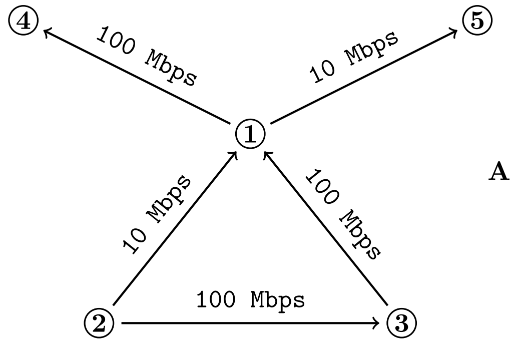
$$\begin{tikzcd}
\omega_0 \arrow[d, swap, "\mathbf{2}"] & & & \\
\omega_1 \arrow[r, "\mathbf{3}"] \arrow[d, swap, "\mathbf{1}"] & \omega_2' \arrow[d, swap, "\mathbf{1}"] & & \\
\omega_2 \arrow[r, "\mathbf{3}"] & \omega_3 \arrow[r, "\mathbf{4}"] & \omega_4 \arrow[r, "\mathbf{5}"] & \omega_5
\end{tikzcd}$$

Figure 5.5: Traces of all execution of Algorithm 2 on the example from Figure 5.4. An ω below another means that it is the smaller of the two with respect to the natural order, which means that all the components from the latter are smaller than the corresponding ones from the former with at least one of them strictly smaller. Two ω at the same level means that they are equal.

The addition of the predecessor vector [11] allows to keep track of the paths that are connecting the source to the various destination. From an algebraic perspective, this information can be coupled to the metric by means of the various composition operators that have been studied in the Metarouting approach [27][29]. The entire set of ω vectors obtained by each source in the network are given below, ω_k^v denotes the vector ω of vertex v at iteration k

$$\begin{array}{lcl}
\omega_5 & = & [\ 0 \quad \infty \quad \infty \quad 1 \quad 2\ ] \quad s = \mathbf{1} \\
\omega_5 & = & [\ 1 \quad 0 \quad 1 \quad 2 \quad 3\ ] \quad s = \mathbf{2} \\
\omega_5 & = & [\ 1 \quad \infty \quad 0 \quad 2 \quad 3\ ] \quad s = \mathbf{3} \\
\omega_5 & = & [\ \infty \quad \infty \quad \infty \quad 0 \quad \infty\ ] \quad s = \mathbf{4} \\
\omega_5 & = & [\ \infty \quad \infty \quad \infty \quad \infty \quad 0\ ] \quad s = \mathbf{5}
\end{array}$$

Concatenating all these rows together give us the fixpoint to the right equation corresponding to the adjacency matrix that was presented on Figure 5.4. Similarly, we illustrate the `bandwidth` structure in Figure 5.6, with the successive ω vectors computed by source **2** given below.



$$\mathbf{A} = \begin{array}{c|ccccc}
 & \mathbf{1} & \mathbf{2} & \mathbf{3} & \mathbf{4} & \mathbf{5} \\
\hline
\mathbf{1} & 0 & 0 & 0 & 100 & 10 \\
\mathbf{2} & 10 & 0 & 100 & 0 & 0 \\
\mathbf{3} & 100 & 0 & 0 & 0 & 0 \\
\mathbf{4} & 0 & 0 & 0 & 0 & 0 \\
\mathbf{5} & 0 & 0 & 0 & 0 & 0
\end{array}$$

Figure 5.6: Graph encoded over `bandwidth`.

In this case, the execution is deterministic at each stage. The best path obtained from **2** towards **4** is **2** $\rightarrow$ **3** $\rightarrow$ **1** $\rightarrow$ **4** while the one towards **5** is **2** $\rightarrow$ **3** $\rightarrow$ **1** $\rightarrow$ **5**

$$\begin{array}{lcllllllll}
\omega_0^2 & = & [& 0 & \infty & 0 & 0 & 0 &] & \\
\omega_1^2 & = & [& 10 & \infty & 100 & 0 & 0 &] & q = \mathbf{2} \\
\omega_2^2 & = & [& 100 & \infty & 100 & 0 & 0 &] & q = \mathbf{3} \\
\omega_3^2 & = & [& 100 & \infty & 100 & 100 & 10 &] & q = \mathbf{1} \\
\omega_4^2 & = & [& 100 & \infty & 100 & 100 & 10 &] & q = \mathbf{4} \\
\omega_5^2 & = & [& 100 & \infty & 100 & 100 & 10 &] & q = \mathbf{5}
\end{array}$$

We are going to prove in general that Dijkstra's Algorithm yields a fixpoint to the right equation when the algebraic structure considered has a SELECTIVE additive law and an INFLATIONARY natural order. First, we will annotate the basic algorithm with iteration counts on the variables manipulated. Note that the source $\mathbf{s}$ which is running the algorithm is implicit in this algorithm; each source computes its own solution independently from other vertices. In essence, with $\mathbf{s}$ being quantified universally, the following equality holds and relates the current knowledge ω in the algorithm with the corresponding row in the matrix being computed. By virtue of this fact, the complete fixpoint $\mathbf{R}$ is effectively computed in a distributed way, with every source $\mathbf{s}$ running Dijkstra's Algorithm computing its own row-vector.

$$\omega_k[d] = \mathbf{R}^k(s, d) \tag{5.3}$$

Algorithm 3 Dijkstra's Algorithm over a SELECTIVE INFLATIONARY semiring

```
1: for all $d \in \mathcal{V}$ do
2:     $\omega_0[d] \leftarrow \overline{0}$
3: end for
4: $\omega_0[s] \leftarrow \overline{1}$
5: $\mathcal{R}_0 \leftarrow \varnothing$
6: for all $k = 1 \ldots |\mathcal{V}|$ do
7:     pick $q_k \in \mathcal{V} - \mathcal{R}_{k-1}$ such that $\forall\, q \in \mathcal{V} - \mathcal{R}_{k-1} : \omega_{k-1}[q_k] \leqslant_\oplus \omega_{k-1}[q]$
8:     $\mathcal{R}_k \leftarrow \mathcal{R}_{k-1} \cup \{q_k\}$
9:     for all $d \in \mathcal{V} - \mathcal{R}_k$ do
10:        $\omega_k[d] \leftarrow \omega_{k-1}[d] \oplus (\omega_{k-1}[q_k] \otimes \mathbf{A}(q_k, d))$
11:    end for
12: end for
```

The annotation we use merely describes the value of the considered object at the end of iteration k. Given that each iteration of the main loop of Line (6) of Algorithm 2 is transferring exactly one vertex from $\mathcal{V} - \mathcal{R}$ to $\mathcal{R}$, the body of the loop Lines (7) – (11) will be executed exactly n times, at which point $\mathcal{R} = \mathcal{V}$.

Lemma 5.1.1. *For all k such that $1 \leqslant k \leqslant |\mathcal{V}|$, we have that:*

$$\forall\, d \in \mathcal{V} : \omega_0[d] = \mathbf{I}(s, d) \tag{5.4}$$

$$\forall\, d \in \mathcal{R}_k : \omega_k[d] = \omega_{k-1}[d] \tag{5.5}$$

$$\forall\, d \in \mathcal{V} - \mathcal{R}_k : \omega_k[d] = \omega_{k-1}[d] \oplus \omega_{k-1}[q_k] \otimes \mathbf{A}(q_k, d) \tag{5.6}$$

Proof. Inspection of Lines (1) – (4) and Lines (9) – (10) of the Algorithm demonstrate this. □

Statement 5.4 covers the initialisation of the algorithm; given that Lines (1) – (2) assign $\overline{0}$ to all entries of the vector ω and $\overline{1}$ for the source, this initialisation effectively assigns the $\mathbf{s}^{th}$ vector from the unit matrix to ω_0. Statement 5.5 states the fact that Line (9) does not apply the update from Line (10) to relaxed vertices $\mathcal{R}_k$; the corresponding values of ω_k are not be changed between the end of iteration $k-1$ and the end of iteration k. Finally, Statement 5.6 is concerned with the unrelaxed vertices for which the distance is updated accordingly. The next lemma revolves around the left relation of Statement 5.1; at any point during the computation, the candidates for relaxation all have a weight greater or equal than that of any previously relaxed vertex.

Lemma 5.1.2. *For all k such that $0 \leqslant k \leqslant |\mathcal{V}|$, we have that:*

$$\forall\, d \in \mathcal{R}_k : \forall\, q_{k+1} \in \min_{\leqslant_\oplus} (\mathcal{V} - \mathcal{R}_k) : \omega_k[d] \leqslant_\oplus \omega_k[q_{k+1}] \tag{5.7}$$

where q_k denotes the vertex selected at Line (7) during the k^{th} iteration.

Proof. By induction on k.

- $k = 0$. $\mathcal{R}_0 = \varnothing$. Vacuous.
- $k > 0$. We suppose the following Induction Hypothesis

$$\forall\, d \in \mathcal{R}_{k-1} \;:\; \forall\, q_k \in \min_{\leqslant_\oplus} (\mathcal{V} - \mathcal{R}_{k-1}) \;:\; \omega_{k-1}[d] \leqslant_\oplus \omega_{k-1}[q_k] \tag{IH}$$

The proof itself relies on the fact that $\mathcal{R}_k = \mathcal{R}_{k-1} \cup \{q_k\}$, which structures the proof into two parts, depending on whether the particular d is in $\mathcal{R}_{k-1}$ or is equal to q_k. We start from the following two facts, with $q_{k+1} \in \min_{\leqslant_\oplus} (\mathcal{V} - \mathcal{R}_k)$;

$$\omega_{k-1}[q_k] \leqslant_\oplus \omega_{k-1}[q_{k+1}] \tag{Line (7)}$$
$$\omega_{k-1}[q_k] \leqslant_\oplus \omega_{k-1}[q_k] \otimes \mathbf{A}(q_k, q_{k+1}) \tag{\textsc{inflationarity}}$$

The first inequality tells us that the weight of the path towards the candidate picked at iteration $k-1$ is more interesting (or equal) than that of the path to the candidate that will be picked during iteration k. The second one indicates that the path going to q_k has a lesser weight than when it is extended towards q_{k+1}. By Lemma 4.1.4, the infimum of the right hand sides of the two previous inequalities is greater than any of their common lower bounds

$$\omega_{k-1}[q_k] \leqslant_\oplus \omega_{k-1}[q_{k+1}] \oplus [\omega_{k-1}[q_k] \otimes \mathbf{A}(q_k, q_{k+1})]$$
$$\Rightarrow\; \omega_k[q_k] \leqslant_\oplus \omega_k[q_{k+1}] \tag{Lemma 5.1.1}$$

The right (resp. left) hand side is rewritten by virtue of Statement 5.6 (resp. $q_k \in \mathcal{R}_k$ and Statement 5.5). By using the Induction Hypothesis and TRANSITIVITY, we obtain

$$\forall\, d \in \mathcal{R}_{k-1} \;:\; \omega_k[d] \leqslant_\oplus \omega_k[q_{k+1}]$$

The conjunction of this with the previous inequality gives us the conclusion

$$\forall\, d \in \mathcal{R}_k \;:\; \omega_k[d] \leqslant_\oplus \omega_k[q_{k+1}]$$

□

The first part of the main claim about Dijkstra's Algorithm; that it is computing a fixpoint to Equation (5.2) can be stated as the following theorem. The fact that $\mathcal{R}_n = \mathcal{V}$ which is used in conjunction with this Theorem follows directly from Line (5) and Line (8).

Theorem 5.1.1. *For all k such that $0 \leqslant k \leqslant |\mathcal{V}|$, we have that:*

$$\forall\, d \in \mathcal{V} \;:\; \omega_k[d] = \sum_{q \in \mathcal{R}_k}^{\oplus} \big[\omega_k[q] \otimes \mathbf{A}(q, d)\big] \oplus \mathbf{I}(s, d) \tag{5.8}$$

Proof. By induction on k.

- $k = 0$. $\mathcal{R}_0 = \varnothing$.

$$\omega_0[d] = \sum_{q \in \varnothing}^{\oplus} \big[\omega_0[q] \otimes \mathbf{A}(q, d)\big] \oplus \mathbf{I}(s, d) = \overline{0} \oplus \mathbf{I}(s, d) = \mathbf{I}(s, d)$$

- $k > 0$. The Induction Hypothesis is the following

$$\omega_{k-1}[d] = \sum_{q \in \mathcal{R}_{k-1}}^{\oplus} \big[\omega_{k-1}[q] \otimes \mathbf{A}(q, d)\big] \oplus \mathbf{I}(s, d)$$

The weight of the path passing through q_k towards d can be added to both hand sides

$$\omega_{k-1}[d] \oplus \big[\omega_{k-1}[q_k] \otimes \mathbf{A}(q_k, d)\big] = \sum_{q \in \mathcal{R}_{k-1}}^{\oplus} \big[\omega_{k-1}[q] \otimes \mathbf{A}(q, d)\big] \oplus \mathbf{I}(s, d) \oplus \big[\omega_{k-1}[q_k] \otimes \mathbf{A}(q_k, d)\big]$$

in which the right hand side can be rewritten by ASSOCIATIVITY and COMMUTATIVITY of $\oplus$ to include the k^{th} term into the sum, with $\mathcal{R}_k = \mathcal{R}_{k-1} \cup \{q_k\}$

$$\omega_{k-1}[d] \oplus \big[\omega_{k-1}[q_k] \otimes \mathbf{A}(q_k, d)\big] = \sum_{q \in \mathcal{R}_k}^{\oplus} \big[\omega_{k-1}[q] \otimes \mathbf{A}(q, d)\big] \oplus \mathbf{I}(s, d)$$

Furthermore, we can apply Part (5.5) of Lemma 5.1.1 to replace the $\omega_{k-1}[q]$ in the right hand side

$$\omega_{k-1}[d] \oplus \big[\omega_{k-1}[q_k] \otimes \mathbf{A}(q_k, d)\big] = \sum_{q \in \mathcal{R}_k}^{\oplus} \big[\omega_{k}[q] \otimes \mathbf{A}(q, d)\big] \oplus \mathbf{I}(s, d) \tag{5.9}$$

Two different cases are covered separately;

– $d \in \mathcal{V} - \mathcal{R}_k$. By Lemma 5.1.1, Part (5.6) allows us to rewrite the left hand side of (5.9)

$$\omega_k[d] = \sum_{q \in \mathcal{R}_k}^{\oplus} \omega_k[q]\mathbf{A}(q, j) \oplus \mathbf{I}(s, d)$$

– $d \in \mathcal{R}_k$. By Part (5.5) of Lemma 5.1.1, it suffices to show the second equality holds in

$$\omega_k[d] = \omega_{k-1}[d] = \omega_{k-1}[d] \oplus \big[\omega_{k-1}[q_k] \otimes \mathbf{A}(q_k, j)\big]$$

Lemma 5.1.2 together with INFLATIONARITY give us a sufficient condition for that

$$\omega_{k-1}[d] \leqslant_{\oplus} \omega_{k-1}[q_k] \leqslant_{\oplus} \omega_{k-1}[q_k] \otimes \mathbf{A}(q_k, j)$$

□

This theorem shows that as Algorithm 2 is progressing through its computation, the distance towards every destination is obtained by combining the distances of relaxed vertices extended by the one arc from those vertices towards the destinations. Line (5) and Line (8) show us that the set $\mathcal{R}$ is increasing at every iteration until the main loop terminates and it covers the entire vertex set itself, at which point $k = n$ with the consequence that $\mathcal{R}_n = \mathcal{V}$ and Equation (5.8) will become

$$\forall\, d \in \mathcal{V} \;:\; \omega_k[d] = \sum_{q \in \mathcal{V}}^{\oplus} \big[\omega_k[q] \otimes \mathbf{A}(q, d)\big] \oplus \mathbf{I}(s, d)$$

By means of Equation (5.3), we can rewrite it into a form equivalent to Equation (5.2)

$$\forall\, d \in \mathcal{V} \;:\; \mathbf{R}^n(s, d) = \sum_{q \in \mathcal{V}}^{\oplus} \big[\mathbf{R}^n(s, q) \otimes \mathbf{A}(q, d)\big] \oplus \mathbf{I}(s, d)$$

The second part of the main claim about Dijkstra's Algorithm is concerned with the unicity of the fixpoint obtained. It allows one to show that whenever there are more than one candidates for Line (7), the order in which these candidates are relaxed will not change the outcome of the algorithm. We introduce an alternative definition of the annotated ω vector; instead of being indexed by the iteration number, we use the list of selected vertices for relaxation that lead to this state. This is a generalisation of the notation that was used on Figure 5.5.

$$\forall\, d \in \mathcal{V} \;:\; \omega_\epsilon[d] = \mathbf{I}(s, d) \tag{5.10}$$
$$\forall\, d \in \mathcal{R}_k \;:\; \omega_{\pi:q_k}[d] = \omega_\pi[d] \tag{5.11}$$
$$\forall\, d \in \mathcal{V} - \mathcal{R}_k \;:\; \omega_{\pi:q_k}[d] = \omega_\pi[d] \oplus \omega_\pi[q_k] \otimes \mathbf{A}(q_k, j) \tag{5.12}$$

where $\pi = q_1 : \cdots : q_{k-1}$. We define the set of candidates for relaxation at iteration k

$$\mathcal{C}_k = \{q_k \in \mathcal{V} - \mathcal{R}_{k-1} \mid \forall\, q \in \mathcal{V} - \mathcal{R}_{k-1} \;:\; \omega_{k-1}[q_k] \leqslant_{\oplus} \omega_{k-1}[q]\}$$

Lemma 5.1.3. *Given a list of previously selected vertices $\pi \in \mathcal{Perm}(\mathcal{R}_{k-1})$ and two vertices q and q' not in π such that $\omega_\pi[q] = \omega_\pi[q']$. Then we have*

$$\forall\, d \in \mathcal{V} \;:\; \omega_{\pi:q:q'}[d] = \omega_{\pi:q':q}[d]$$

Proof. We have to consider two cases;

- $d \in (\mathcal{R}_{k-1} \cup \{q, q'\})$. This is proven by the conjunction of

$$\forall\, d \in \mathcal{R}_{k-1} \;:\; \omega_{\pi:q:q'}[d] = \omega_{\pi:q}[d] = \omega_\pi[d] = \omega_{\pi:q'}[d] = \omega_{\pi:q':q}[d]$$

$$\begin{aligned} \omega_{\pi:q:q'}[q] = \omega_{\pi:q}[q] = \omega_\pi[q] &= \omega_\pi[q] \oplus \omega_\pi[q'] \otimes \mathbf{A}(q', q) = \omega_{\pi:q'}[q] = \omega_{\pi:q':q}[q] \\ \omega_{\pi:q:q'}[q'] = \omega_{\pi:q}[q'] = \omega_\pi[q'] \oplus \omega_\pi[q] \otimes \mathbf{A}(q, q') &= \omega_\pi[q'] = \omega_{\pi:q'}[q'] = \omega_{\pi:q':q}[q'] \end{aligned}$$

  The central equalities in the last two equations follow from INFLATIONARITY.

- $d \notin (\mathcal{R}_{k-1} \cup \{q, q'\})$. By Definition (5.12), together with INFLATIONARITY, we can derive

$$\begin{aligned} \omega_{\pi:q:q'}[d] &= \omega_{\pi:q}[d] \oplus \omega_{\pi:q}[q'] \otimes \mathbf{A}(q', j) \\ &= \omega_\pi[d] \oplus \omega_\pi[q] \otimes \mathbf{A}(q, j) \oplus [\omega_\pi[q'] \oplus \omega_\pi[q] \otimes \mathbf{A}(q, q')]\mathbf{A}(q', j) \\ &= \omega_\pi[d] \oplus \omega_\pi[q] \otimes \mathbf{A}(q, j) \oplus \omega_\pi[q'] \otimes \mathbf{A}(q', j) \\ \omega_{\pi:q':q}[d] &= \omega_{\pi:q'}[d] \oplus \omega_{\pi:q'}[q] \otimes \mathbf{A}(q, j) \\ &= \omega_\pi[d] \oplus \omega_\pi[q'] \otimes \mathbf{A}(q', j) \oplus [\omega_\pi[q] \oplus \omega_\pi[q'] \otimes \mathbf{A}(q', q)]\mathbf{A}(q, j) \\ &= \omega_\pi[d] \oplus \omega_\pi[q'] \otimes \mathbf{A}(q', j) \oplus \omega_\pi[q] \otimes \mathbf{A}(q, j) \end{aligned}$$

  where COMMUTATIVITY is a sufficient condition for the two terms to be equal.

□

Lemma 5.1.4. *Given the set of candidate vertices $\mathcal{C}_k$, such that $\forall\, q, q' \in \mathcal{C}_k \;:\; \omega_\pi[q] = \omega_\pi[q']$ and the permutation $\pi \in \mathcal{P}erm(\mathcal{R}_{k-1})$. Then we have*

$$\forall\, d \in \mathcal{V} \;:\; \forall\, \sigma, \sigma' \in \mathcal{P}erm(\mathcal{C}_k) \;:\; \omega_{\pi:\sigma}[d] = \omega_{\pi:\sigma'}[d]$$

Proof. By induction on the size of $\mathcal{C}_k$.

- $|\,\mathcal{C}_k\,| = 0$. $\mathcal{C}_k = \varnothing$. $\mathcal{P}erm(\varnothing) = \varnothing$. Vacuous.
- $|\,\mathcal{C}_k\,| > 0$. For every $\mathcal{C}_1, \mathcal{C}_2 \subset \mathcal{C}_k$ and $q_1, q_2 \in \mathcal{C}_k$ such that $\mathcal{C}_k = \mathcal{C}_1 \cup \{q_1\} = \mathcal{C}_2 \cup \{q_2\}$. Consider all permutations of the form $\pi : q_1 \in \mathcal{P}erm(\mathcal{C}_1)$ and $\pi : q_2 \in \mathcal{P}erm(\mathcal{C}_2)$. By Lemma (5.1.3), we have $\omega_{\pi:q_1:q_2}[d] = \omega_{\pi:q_2:q_1}[d]$. The universal quantification gives us the statement for all permutations in $\mathcal{P}erm(\mathcal{C}_k)$.

□

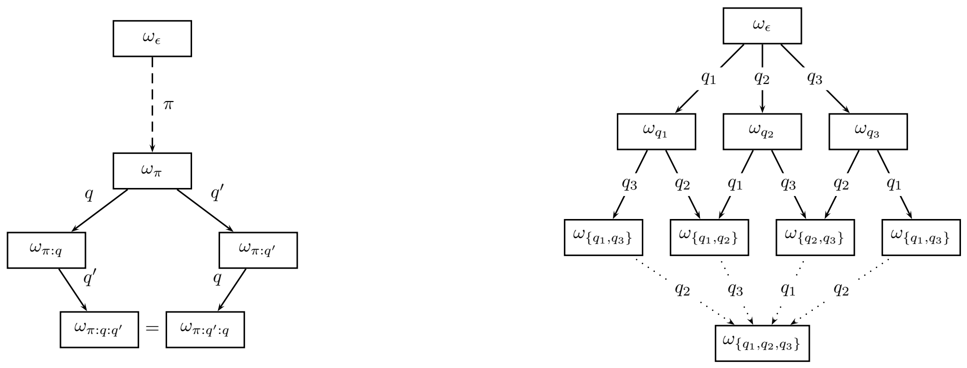


Figure 5.7: Left: Lemma 5.1.3 where every arrow represent one application of the body of Dijkstra's main loop considering the vertex labelling it as the candidate selected by Line (7). Right: Lemma 5.1.4 for $|\,\mathcal{C}_k\,| = 3$.

Theorem 5.1.2. *All the fixpoints that could be computed by Dijkstra's Algorithm coincide.*

Proof. Using Lemma (5.1.4) and the appropriate definitions extending $\mathcal{C}_k$, we can inductively go through the successive sets of $\leqslant_\oplus$-least candidates. Every step yields a unique answer. □

The consequence of this theorem is that despite the non-determinism introduced by Line (7); whenever several vertices satisfy the condition of being the closest any can be picked for relaxation, Dijkstra's Algorithm always reaches the same fixpoint to the right equation, even when DISTRIBUTIVITY does not hold. However it does not tell us whether there is a unique least fixpoint to the right equation. Assuming we are given another algorithm that also computes a least fixpoint to the same right equation, the question of whether Dijkstra's Algorithm computes the same fixpoint as this algorithm when DISTRIBUTIVITY does not hold is unanswered within the theory. A solution to this potential issue would be a thorough description of the structure of the set of fixpoints to recurrence equations of the form Equation (5.2).

5.2 Distance-vectoring algorithms

The second class of routing algorithm is based on a distance-vectoring method. In essence, the idea behind is for the participating routers to exchange information about the destinations they can reach. A router collects these announces from its directly connected neighbours to further expand its own connectivity by including these paths. The paths that a router discovers are always consistent with those of other routers given that bases its own routing table on those of its neighbors. The most widely known representative for this class of algorithms is the Bellman-Ford Algorithm [5]. In the context of routing, the first implementations of this algorithm were distributed [6][32]. Given that we are more interested in what is computed rather than all the possible executions involved in a distributed environment, we will consider a centralized version as a simplification. This does not limit the adequacy with reality given that the transition from a centralized computation to a distributed one has been studied in the past [7, 18] and is a separate concern.

$$\mathbf{L} = \mathbf{AL} \boxplus \mathbf{I} \tag{5.13}$$

In this section, we will prove that the iterative method described in Section 4.2.2 does converge to a fixpoint of such a left equation. The proof of Dijkstra's Algorithm convergence to a fixpoint relies on the explicit set $\mathcal{R}_k$ which stores the relaxed vertices. This set equivalently contains the vertices for which the weight will no longer change through the remainder of the computation. We argue that this set also exist in an implicit form in the computation performed by the Bellman-Ford algorithm and we can leverage an appropriate definition to prove that the vertices that end up in that set (1) have their weights unchanged for the rest of the computation and (2) remain in that set until the termination of the Algorithm.

We first define the *set of locally stabilized vertices at iteration* k which contains the vertices whose weight will not be changed by the next iteration:

$$\sigma_k = \{i \in V \mid \mathbf{L}^k(i,j) = \mathbf{L}^{k+1}(i,j)\} \tag{5.14}$$

For convenience, we will use the notation $\overline{\sigma}_k \equiv V - \sigma_k$. Note that it is not necessary for a vertex to stay in this set; changes to other vertice's weight can affect stable ones at a later iteration. In order to be able to prove convergence, we would require a set that grows throughout the computation until it coincides with the set V. The set of globally stabilized vertices at iteration k contains the vertices whose weight will not be surpassed by any locally non-stable vertex at the next iteration:

$$\Sigma_k = \{s \in \sigma_k \mid \forall\, v \in \overline{\sigma}_k : \mathbf{L}^k(s,j) \leqslant_\oplus \mathbf{L}^{k+1}(v,j)\} \tag{5.15}$$

We also define the *set of candidates at iteration* k which contains the vertices not globally stabilized which have the least $\mathbf{L}$-value.

$$\mathcal{C}_k = \{c \in \overline{\Sigma}_k \mid \forall\, v \in \overline{\Sigma}_k : \mathbf{L}^{k+1}(c,j) \leqslant_\oplus \mathbf{L}^{k+1}(v,j)\} \tag{5.16}$$

We assume an IDEMPOTENT bimagmoid $(\mathcal{S}, \oplus, \otimes)$ and a LEFT-INFLATIONARY matrix $\mathbf{A}$.

Lemma 5.2.1. $\forall\, a, b \in \mathcal{S} : a \oplus b \leqslant_\oplus b$

Proof. By ASSOCIATIVITY and IDEMPOTENCY. □

Lemma 5.2.2. $\forall\, a, b, c \in \mathcal{S} : c \leqslant_\oplus a \wedge c \leqslant_\oplus b \Leftrightarrow c \leqslant_\oplus (a \oplus b)$

Proof. ($\Rightarrow$) By ASSOCIATIVITY. ($\Leftarrow$) By Lemma 5.2.1 and TRANSITIVITY. □

Lemma 5.2.3. *For every* $s \in \Sigma_k$ *and* $v \in \overline{\Sigma}_k$ *we have* $\mathbf{L}^k(s,j) \leqslant_\oplus \mathbf{L}^{k+1}(v,j)$.

Proof. Note that $\overline{\Sigma}_k \equiv \overline{\sigma}_k \cup (\sigma_k - \Sigma_k)$. The property holds by definition when $v \in \overline{\sigma}_k$. Let $v \in (\sigma_k - \Sigma_k)$. $\exists\, o \in \overline{\sigma}_k : \mathbf{L}^{k+1}(o,j) < \mathbf{L}^k(v,j) = \mathbf{L}^{k+1}(v,j)$ and $\mathbf{L}^k(s,j) \leqslant_\oplus \mathbf{L}^{k+1}(o,j)$ which imply that $\mathbf{L}^k(s,j) \leqslant_\oplus \mathbf{L}^{k+1}(v,j)$. □

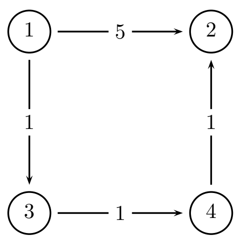
We make a simplification with respect to the ambiguous Bellman-Ford Algorithm as it appears in Cormen [11]; the order in which the set of arcs is visited is first defined by the targets of edges then by the origins; j varies in the outer loop and i in the inner loop.

Consequently the k^{th} value of $\mathbf{L}$ is determined columnwise in an independent way; the original algorithm updates the $\mathbf{L}$ vector along the way inside the innermost loop. Consider how the annotation we add in Algorithm 4 changes it; without it the order in which q is selected in the sum would have to be taken into account since the $\mathbf{L}(q,j)$ could have already been updated by a previous iteration of the inner loop so it could be $\mathbf{L}^k(q,j)$ and not $\mathbf{L}^{k-1}(q,j)$.

Algorithm 4 Centralized Bellman-Ford Algorithm

```
1: for all $i, j \in \mathcal{V}$ do
2:     $\mathbf{L}^0(i,j) \leftarrow \mathbf{I}(i,j)$
3: end for
4: for all $k = 1 \ldots |\mathcal{V}|$ do
5:     for all $j \in \mathcal{V}$ do
6:         for all $i \in \mathcal{V}$ do
7:             $\mathbf{L}^k(i,j) \leftarrow \mathbf{L}^{k-1}(i,j) \oplus \sum^{\oplus}_{q \in \mathcal{V}} \mathbf{A}(i,q)\mathbf{L}^{k-1}(q,j)$
8:         end for
9:     end for
10: end for
```

The proof is structured by first showing that convergence holds column-wise (Theorem 5.2.1). The focus is on the inner loop since the destination j is fixed. This result can be used to show that for all j, the columns will converge independently. The second part of the proof shows that Algorithm 4 is computing a fixpoint to Equation (5.13). The algorithm along with the sets σ_k and Σ_k are illustrated by the Example on Figure 5.8.



$$\mathbf{A} = \begin{array}{c} \\ ① \\ ② \\ ③ \\ ④ \end{array} \begin{array}{c} \begin{array}{cccc} ① & ② & ③ & ④ \end{array} \\ \left[\begin{array}{cccc} \infty & 5 & 1 & \infty \\ \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & 1 \\ \infty & 1 & \infty & \infty \end{array}\right] \end{array}$$

$$\begin{array}{lcl} \mathbf{L}^1_0 & = & [\ \boxed{\underline{0}}\ \ \infty\ \ \infty\ \ \boxed{\infty}\ ] \\ \mathbf{L}^1_1 & = & [\ \boxed{\underline{0}}\ \ \boxed{5}\ \ \boxed{\underline{1}}\ \ \infty\ ] \\ \mathbf{L}^1_2 & = & [\ \boxed{\underline{0}}\ \ 5\ \ \boxed{\underline{1}}\ \ \boxed{\underline{2}}\ ] \\ \mathbf{L}^1_3 & = & [\ \boxed{\underline{0}}\ \ \boxed{\underline{3}}\ \ \boxed{\underline{1}}\ \ \boxed{\underline{2}}\ ] \end{array}$$

Figure 5.8: Example of a simple graph weighted over SP and the successive values of the first row-vector of $\mathbf{L}^1_k$ (corresponding to source 1). The weights in a green box (resp. underlined) correspond to vertices that are in the set σ_k (resp. Σ_k). Note how $\mathbf{L}^1_3 \leqslant_\oplus \mathbf{L}^1_2 \leqslant_\oplus \mathbf{L}^1_1 \leqslant_\oplus \mathbf{L}^1_0$; each iteration improves the solution.

Lemma 5.2.4.

$$\forall\, i \in V : \mathbf{L}^k(i,j) = \mathbf{L}^{k-1}(i,j) \oplus \sum_{q \in V}^{\oplus} \mathbf{A}(i,q)\mathbf{L}^{k-1}(q,j)$$

Proof. By Lines 6-7 of Algorithm 4. □

Corollary 5.2.1. *For every $s \in \Sigma_k$ and $v \in \overline{\Sigma}_k$ we have*

$$\mathbf{L}^k(s,j) \leqslant_\oplus \mathbf{L}^k(v,j) \quad \textit{and} \quad \mathbf{L}^k(s,j) \leqslant_\oplus \sum_{q \in V}^{\oplus} \mathbf{A}(v,q)\mathbf{L}^k(q,j)$$

Proof. By Lemma 5.2.3, we have

$$\begin{aligned}\mathbf{L}^k(s,j) &\leqslant_{\oplus} \mathbf{L}^{k+1}(v,j) \\ &\leqslant_{\oplus} \mathbf{L}^k(v,j) \oplus \sum_{q\in V}^{\oplus} \mathbf{A}(v,q)\mathbf{L}^k(q,j) && \text{(Lemma 5.2.4)}\end{aligned}$$

By Lemma 5.2.2, we obtain the claim. □

The main theorem that has to be proved to show both convergence and the complexity is the following:

Theorem 5.2.1. $\Sigma_k \cup \mathcal{C}_k \subseteq \Sigma_{k+1}$.

Proof. Assume that $s \in \Sigma_k \cup \mathcal{C}_k$. We have for all $q \in \overline{\Sigma}_k$

$$\begin{aligned}\mathbf{L}^k(s,j) = \mathbf{L}^{k+1}(s,j) &\leqslant_{\oplus} \mathbf{L}^{k+1}(q,j) && (s \in \Sigma_k \subseteq \sigma_k\text{, Lemma 5.2.3}) \\ \mathbf{L}^{k+1}(s,j) &\leqslant_{\oplus} \mathbf{L}^{k+1}(q,j) && (s \in \mathcal{C}_k)\end{aligned}$$

In either case, we can deduce for all vertices $v \in V$

$$\begin{aligned}\mathbf{L}^{k+1}(s,j) &\leqslant_{\oplus} \mathbf{A}(v,q)\mathbf{L}^{k+1}(q,j) && (\textsc{left-inflationarity}) \\ &\leqslant_{\oplus} \sum_{q\in\overline{\Sigma}_k}^{\oplus} \mathbf{A}(v,q)\mathbf{L}^{k+1}(q,j) && \text{(Lemma 5.2.2)}\end{aligned}$$

On the other hand, for $o \in \overline{\Sigma}_k \cup \{s\}$

$$\begin{aligned}\mathbf{L}^{k+1}(s,j) &\leqslant_{\oplus} \mathbf{L}^{k+1}(o,j) \\ &\leqslant_{\oplus} \mathbf{L}^{k+1}(o,j) \oplus \sum_{q\in\Sigma_k}^{\oplus} \mathbf{A}(o,q)\mathbf{L}^{k}(q,j) && (\text{Lemma 5.2.4, } \textsc{idempotency}) \\ &\leqslant_{\oplus} \mathbf{L}^{k+1}(o,j) \oplus \sum_{q\in\Sigma_k}^{\oplus} \mathbf{A}(o,q)\mathbf{L}^{k+1}(q,j) && (\Sigma_k \subseteq \sigma_k)\end{aligned}$$

By Lemma 5.2.2, COMMUTATIVITY and ASSOCIATIVITY, the last two inequalities give

$$\mathbf{L}^{k+1}(s,j) \leqslant_{\oplus} \mathbf{L}^{k+1}(o,j) \oplus \sum_{q\in V}^{\oplus} \mathbf{A}(o,q)\mathbf{L}^{k+1}(q,j) = \mathbf{L}^{k+2}(o,j)$$

By Lemmas 5.2.1 and 5.2.4 we get $\mathbf{L}^{k+2}(s,j) \leqslant_{\oplus} \mathbf{L}^{k+1}(s,j)$. By ANTI-SYMMETRY, $s \in \sigma_{k+1}$. From this we can also deduce that $\overline{\sigma}_{k+1} \subseteq \overline{\Sigma}_k$ which concludes the proof. □

Corollary 5.2.2. $\Sigma_k = \Sigma_{k+1} \Rightarrow \Sigma_k = V$.

Proof. Assume that $\Sigma_k = \Sigma_{k+1} \wedge \Sigma_k \neq V$. Then $\overline{\Sigma}_k$ is not empty and it must contain some elements of least $\mathbf{L}$ value which implies that $\mathcal{C}_k$ is not empty. So $\Sigma_k \neq \Sigma_{k+1}$. *Reductio ad absurdum.* □

The longest chain of Σ_k we can get is whenever one set has exactly one more vertex than the previous one. The maximal k in this case is n with $\Sigma_n = V$. The outer loop can therefore be bound by n, so $\mathcal{O}(n^3)$ is the complexity of the Algorithm. In the end, $V = \Sigma_n \subseteq \sigma_n \subseteq V \Rightarrow \sigma_n = V$ so $\mathbf{L}$ stabilizes. As a consequence, Algorithm 4 is computing a fixpoint to Equation (5.13).

A well known property of Bellman-Ford's algorithm is to work even when negative weights are used as labels [11]. Under this loosened restriction, the algorithm can still compute the shortest-paths as long as negative weighted cycles do not occur in the graph considered [21][39]. Gondran & Minoux provided an algebraic proof [25] but require DISTRIBUTIVITY to hold. They studied the problem by assuming the absence of 0-absorbing cycles in the graph considered. This implies that the algorithm running within an algebraic structure for which all elements are 0-stable converges to a fixpoint. On the other hand, Gurney produced a proof for convergence [28] but requires INFLATIONARITY, which is a stronger restriction that the absence of q-absorbing cycles. The question of whether Algorithm 4 converges to a fixpoint for an algebraic structure for which all elements are 0-stable but does not satisfy DISTRIBUTIVITY is open at this stage.

Chapter 6

A model of Route Aggregation

I love humans. Always seeing patterns in things that aren't there.

The Eighth Doctor

The establishment of routes in the Internet is governed by the interaction of two independent processes. The global infrastructure is divided into Autonomous Systems which are interconnected together to exchange route information in order to discover the best routes towards destinations. We have described in Chapter 3 how domains use route advertisements to populate their forwarding tables. At the internal level, the routers that make up the network construct paths through the domain, enabling the border routers to identify how to reach the internal peers. At the external level, the border routers advertise their best routes towards destinations to each other. In this context, destinations are described by network prefixes which concisely represent ranges of addresses that are somehow reachable through a certain physical network. The route aggregation mechanism leverages the subprefix relation that exists between prefixes along with a separation between effective routes and advertised routes, enabling the merging of routes for destinations that are covered by a common network prefix. The attributes pertaining to the aggregated routes are combined together according to a set of rules resulting in those of the aggregate route.

In this chapter, we will describe an algebraic model of route aggregation. We will treat separately two processes that seem to be at work; aggregation and origination. We will approach the aggregation by expressing the computation of the exhaustive set of aggregation rules which exist between any two pair of vertices. Given that aggregation rules can be expressed by functions which reduce sets of effective routes, the composition of those functions can be used to express how all the effective routes at a given router j are transformed by all the sequences of routers which exist between another router i and router j. On the other hand, the routes originated by each router determine the best routes that are effectively used by all the others. This origination will be performed after the compound aggregation rules have been established to produce the effective routes at each router.

We will approach the problem in two separate steps. At first, we will look at route aggregation as it is performed in the context of inter-domain route exchanges. Our focus will be on the aspects of attributes that play a role limited to the external peerings that exist between border routers of different domains. We will reduce the problem by assuming that each domain is composed of a single border router. The route selection performed by BGP speakers will be revisited by keeping the attributes that play a role in external peerings. The attributes that are relevant on the exterior are all characterized by a preference on them, as defined by the tie-breaking rules of BGP, along with operations that transform them as the route information is carried between neighboring domains. While BGP is described as a path-vectoring protocol, it is strongly reminiscent of distance-vectoring in the sense that border routers base their decisions on the routes advertised by their neighbors. For this reason, the problem will be approached as that of solving a left equation over square matrices with elements taken in a prebimonoid.

Secondly, we will investigate how the route aggregation within an IGP can be modelled. In [36], Le & al. investigated the behaviour of Cisco and Juniper routers in order to identify the problems that arise from their implementation of route aggregation. They proposed a canonical router model which is an abstract representation of how various functional components of a router interact together. By using this model, they discovered multiple anomalies that can arise between routers configured to use route aggregation narrowing down the root cause behind most of them to the absence or improper use of sink routes. We will apply a similar construct as the one used for BGP to the interior of a domain and show how the forwarding loops are effectively captured in our approach.

Our motivation for this work is to explore how route aggregation can be expressed in the context of solving recurrence equations in a way that enables the study of the anomalies that can occur due to its use. While the problem has already been studied algorithmically [36], we seek to deconstruct the monolithic version of BGP in order to identify the specific component in which aggregation is encapsulated. On one hand, the path-vectoring method used by the peers is based on the exchange of route information in order to produce the sets of effective routes. In this format, the route information encodes the destination as a logical piece of information, namely the network prefix, in order to keep track of where the route leads to. On the other hand, the computation of a fixpoint to a recurrence equation maintains separate the physical destinations which are represented by the indices of the square matrices being computed. We choose to reflect the network prefixes into the information and define suitable operations which accurately express the modification to this information between any pair of vertices. In effect, the matrices include more structure into the set of effective routes than the path-vectoring method. However, we define a function that maps any square matrix to sets of effective routes and show that each successive matrix obtained during the computation of a fixpoint reduces to the sets of effective routes at the corresponding iteration. This consistency does not only cover the classical route aggregation scenario but also the particular behaviours that are multi-homing and re-homing. The theoretical considerations we presented in Chapter 4 and Chapter 5 include various properties that are sufficient to guarantee the convergence of Bellman-Ford's Algorithm to a solution along with the loop-freedom of the obtained routes. However, the resulting prebimonoid structure we obtain does not satisfy INFLATIONARITY which was assumed and we are unable to make any formal claims as to the convergence of the path-vectoring method or the loop-freedom of the routes obtained.

The model we describe may look overly simplistic to capture the complexity of BGP at the inter-domain and intra-domain levels. Despite this deliberate choice to restrict ourselves to the inter-domain level restricted to two route attributes, we are able to apply it to other cases with minor modifications. In particular, we illustrate at the end of this chapter how the route aggregation performed by a simple IGP can be expressed with a slight modification to the model of eBGP.

6.1 Anatomy of route information

Routes in BGP are used to describe how to reach physical networks to which a certain prefix is attached. This simple statement allows us to relate routes learned by border routers, for example **B0** of ISP 1 (Figure 6.3), to the physical network of ISP 2, Customer 6 and Customer 8 which happen to be reachable by use of the addresses contained within the range described by the `NLRI` field below.

`NLRI`	`LOCAL_PREF`	`AS_SEQUENCE`	Source	Int. Cost	`NEXT_HOP`	Imm. Next-Hop
`10.2.0.0/16`	120	`65002, 65006`	`iBGP`	2	**ISP2.B0**	**ISP1.R0**
`10.2.6.0/24`	120	`65002, 65006`	`iBGP`	2	**ISP2.B0**	**ISP1.R0**
`10.2.8.0/24`	120	`65003, 65008`	`iBGP`	2	**ISP3.B0**	**ISP1.R0**

Figure 6.1: Portion of the effective routes at **ISP1.B0** under no aggregation.

In order to reach any host within the physical network of some domain, an address within the range assigned to it must be used as the destination address. Suppose that Customer 4 originates a traffic bound for the address `10.2.6.42`. According to the forwarding tables it will produce, **C4.B0** will forward this traffic to **ISP1.B0**. By using the internal paths through the network of ISP 1, the packet will be reach **ISP1.B2** who will pass it to **ISP2.B0**. The internal process will be repeated until the packet reaches **C6.B0** who will be able to deliver the packet to its intended destination. The exchange of route information can be modelled by considering the network prefix as a logical information that describes the means by which a physical network can be reached. This information is attached to the vertices on Figure 6.4 by means of the configuration that the network administrators deploy on the routers of their domain.

Under an assumption of full-aggregation, the border routers of ISP 2 are configured to aggregate every route that falls under its prefix, `10.2.0.0/16`. In this case, the router **B0** of ISP 1 would be using the following routes with the first one received from **ISP1.B2** and allowing to reach both the physical network of ISP 2 as well as the physical network of Customer 6. The second route allows to reach the physical network of Customer 8 which has re-homed from ISP 2 to ISP 3.

`NLRI`	`LOCAL_PREF`	`AS_SEQUENCE`	Source	Int. Cost	`NEXT_HOP`	Imm. Next-Hop
`10.2.0.0/16`	120	65002	iBGP	2	**ISP2.B0**	**ISP1.R0**
`10.2.8.0/24`	120	65003, 65008	iBGP	2	**ISP3.B0**	**ISP1.R0**

Figure 6.2: Portion of the effective routes at **ISP1.B0** under full-aggregation.

The various attributes that are considered in the routes described above are laid out according to their priority in the tie-breaking rules of BGP [47] with the exception of the `NEXT_HOP` and immediate next-hop that do not influence the choice but are determined by the other attributes. The same holds for the `NLRI` which only serves as a destination identifier that must be unique across all effective routes within a forwarding table. We will focus on the first fields of the routes presented above, given that all those following the source field are used to distinguish between routes learned from iBGP or eBGP. In the complete set of tie-breaking rules, the `ORIGIN` and `MULTI_EXIT_DISC` attributes appear between the `AS_SEQUENCE` and the source. Furthermore, after the interior cost has been considered, the rules specify that Phase 2 of the Decision Process should find the route with the lowest BGP Identifier followed by the lowest advertising peer address. These last pieces of information guarantee that given an arbitrary set of input routes, the Decision Process will always yield at most one effective route for each distinct `NLRI`.

The effective routes are those which are installed in the forwarding tables of routers and are used to dispatch traffic based on its intended destination according to the Longest-Match Prefix rule. Suppose the effective routes given on Figure 6.2 as those in the forwarding table of router **B0** from ISP 1. If it receives a packet bound for the address `10.2.8.42`, it will perform a lookup in its forwarding table where both routes match for this destination. The LMP rule states that the route with the most specific prefix shall always be used for forwarding, hence the second will be picked. We will say that this route is an LMP-route for destination `10.2.8.42` at router **ISP1.B0**. By extension, it is an LMP-route for the entire prefix `10.2.8.0/24`.

The route information we consider is restricted to three pieces of information; the `NLRI`, the `AS_SEQUENCE` and the advertising peer address. We choose to limit ourselves to the `AS_SEQUENCE` instead of the `AS_PATH` given that the default behaviour on Cisco routers does not require `AS_SET`. The `AS_SEQUENCE` is a list of ASN to which each AS prepends its own with a preference given to shorter lists.

Consider the infrastructure on Figure 6.4 which is obtained from Figure 6.3 by reducing the domains to single BGP speakers. The following points have to be kept in mind in order to understand how the speakers construct their forwarding tables.

- At most one route for each distinct prefix is kept in the set of effective routes
- An effective route can be the LMP-route for more than one destination domain
- Routes are preferred based on the length of the `AS_SEQUENCE` with a tie-breaking on the advertising peer address. In Figure 6.4, we consider that the preference on the peer addresses for Figure 6.4 is given by $\mathbf{0} < \mathbf{I1} < \mathbf{I2} < \mathbf{I3} < \mathbf{C4} < \mathbf{C5} < \mathbf{C6} < \mathbf{C7} < \mathbf{C8}$. The special value of **0** denotes the lowest peer address.
- In the presence of multiple effective routes matching the prefix of a destination domain with network prefixes related by the subprefix relation, the most specific one is favoured according to the LMP rule.
- In the presence of multiple routes with disjoint network prefixes, all are usable to reach a router in the context of multi-homing
- The route attributes undergo two transformations upon exchange between two routers; the ASN of the AS who sends the route is prepended to the `AS_SEQUENCE` and the advertising peer address is set to the address of the peer who sends the route information.

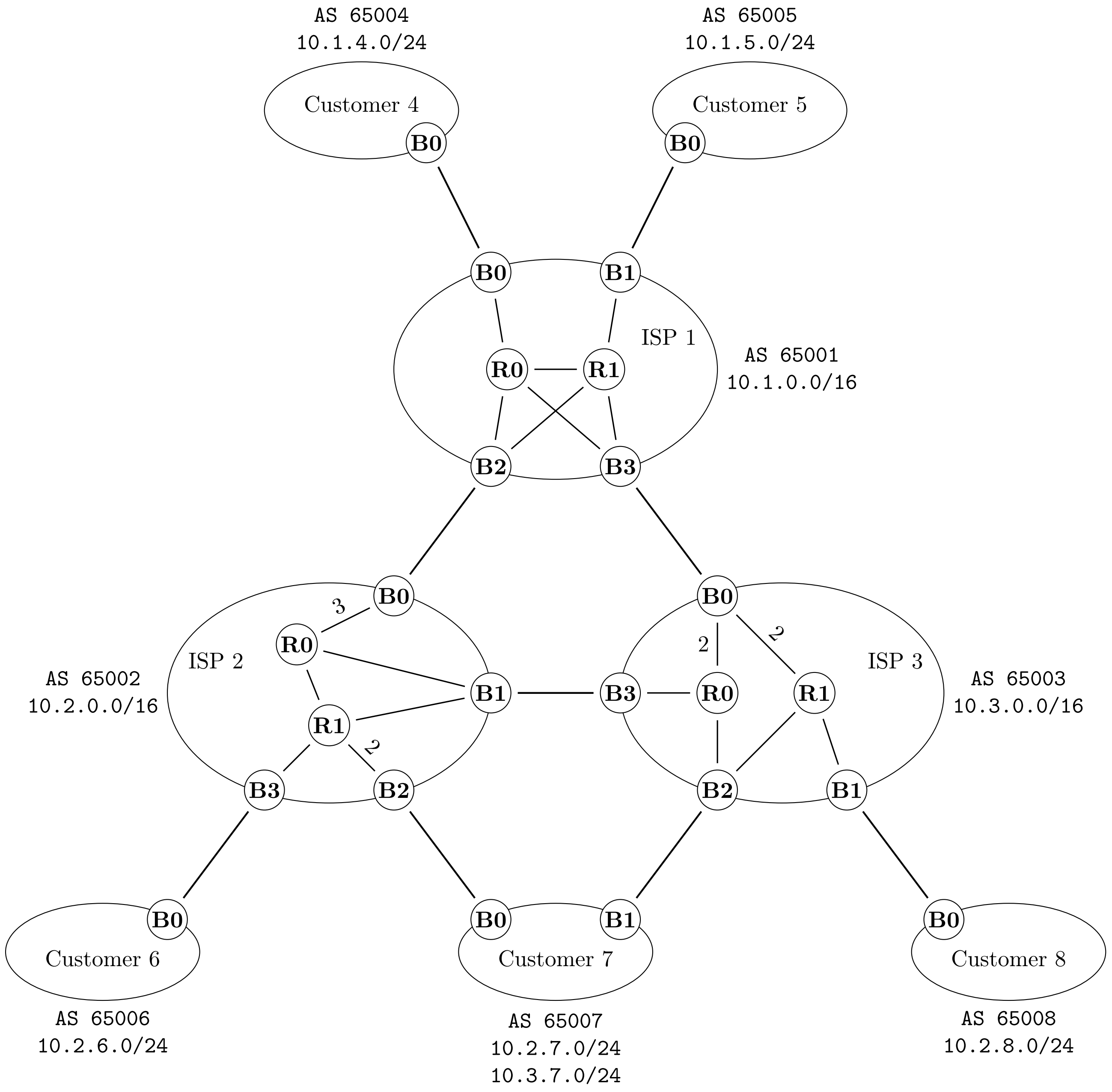


Figure 6.3: Detailed infrastructure with internal structure of ISPs.

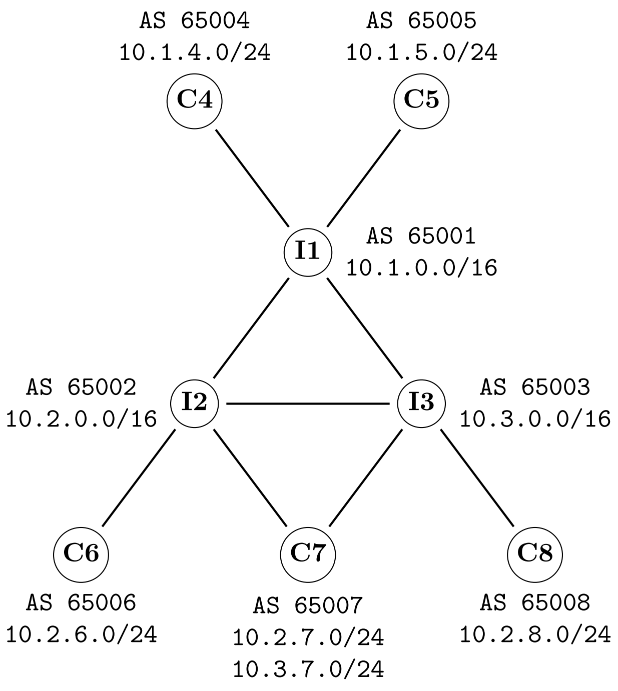


Figure 6.4: Infrastructure without internal structure of ISPs. Links represent external peerings between routers.

- In the presence of aggregation rules, the `NLRI` attribute of routes with a matching prefix is replaced by the one from the aggregation rule and the `AS_SEQUENCE` is reset to the ASN of the aggregating router.

- When a route from the input set has an `AS_SEQUENCE` which contains the AS Number of the speaker running the Decision Process, it should not be considered for the selection of best routes. In our approach, we do not include this additional restriction in the process that construct the best routes. However, the routes for which the `AS_SEQUENCE` contains loops are not given in the forwarding tables or the fixpoints.

This representation considers one degree of freedom for the configurations deployed by network administrators; the set of aggregation rules to be applied. We will consider the case where full-aggregation is used with the resulting forwarding tables given on Figure 6.5. Note that given the existence of a cycle between **I1**, **I2** and **I3**, there should be entries pertaining to the respective prefixes of those routers with an `AS_SEQUENCE` containing a loop. In BGP, a constraint exists on the routes in the input set that must not be considered for Phase II, specifically the routes must not have an `AS_SEQUENCE` which contains the AS Number of the router receiving them. We choose to not include these entries for clarity but the reader should remember of their existence. Under full-aggregation, the routers are configured with their allocated network prefix as the only aggregation rule. In this context, we assume the default behaviour of Cisco routers where the aggregating router resets the `AS_SEQUENCE` to its ASN [36]. The physical routers that are reachable through the corresponding routes are also included in the last column.

Note that all the specific prefixes are not known explicitly in all routers of the infrastructure but this does not prevent forwarding from being performed correctly. A traffic originating at router **C4** bound for the network prefix `10.2.8.0/24` will follow the path **C4** $\rightarrow$ **I1** $\rightarrow$ **I3** $\rightarrow$ **C8** under the LMP rule. In practice, the customers are using their border router to obtain connectivity to the rest of the infrastructure. It is possible to use a default route for destination `0.0.0.0/0` with the external peer of their border router as the `NEXT_HOP` to further reduce the size of their forwarding table.

We will now describe an algebraic model which produces the LMP-routes between any two domains. The LMP-routes are characterized as having the most specific prefix and shortest `AS_SEQUENCE` among all possible routes that exist between any two pair of domains while the effective routes are a subset of all the LMP-routes at a given router which satisfy a unicity constraint based on the `NLRI` field.

Router	NLRI	AS_SEQUENCE	Advertising peer	Pertains to
I1	10.1.4.0/24	65004	**C4**	**C4**
	10.1.5.0/24	65005	**C5**	**C5**
	10.2.0.0/16	65002	**I2**	**I2**, **C6**, **C7**
	10.2.8.0/24	65003, 65008	**I3**	**C8**
	10.3.0.0/16	65003	**I3**	**I3**, **C7**
I2	10.1.0.0/16	65001	**I1**	**I1**, **C4**, **C5**
	10.2.6.0/24	65006	**C6**	**C6**
	10.2.7.0/24	65007	**C7**	**C7**
	10.2.8.0/24	65003, 65008	**I3**	**C8**
	10.3.0.0/16	65003	**I3**	**I3**, **C7**
I3	10.1.0.0/16	65001	**I1**	**I1**, **C4**, **C5**
	10.2.0.0/16	65002	**I2**	**I2**, **C6**, **C7**
	10.3.7.0/24	65007	**C7**	**C7**
	10.2.8.0/24	65008	**C8**	**C8**
C4	10.1.0.0/16	65001	**I1**	**I1**, **C5**
	10.2.0.0/16	65001,65002	**I1**	**I2**, **C6**, **C7**
	10.3.0.0/16	65001,65003	**I1**	**I3**, **C7**
	10.2.8.0/24	65001,65003,65008	**I1**	**C8**
C5	10.1.0.0/16	65001	**I1**	**I1**, **C4**
	10.2.0.0/16	65001,65002	**I1**	**I2**, **C6**, **C7**
	10.3.0.0/16	65001,65003	**I1**	**I3**, **C7**
	10.2.8.0/24	65001,65003,65008	**I1**	**C8**
C6	10.1.0.0/16	65002,65001	**I2**	**I1**, **C4**, **C5**
	10.2.0.0/16	65002	**I2**	**I2**, **C7**
	10.3.0.0/16	65002,65003	**I2**	**I3**, **C7**
	10.2.8.0/24	65002,65003,65008	**I2**	**C8**
C7	10.1.0.0/16	65002,65001	**I2**	**I1**, **C4**, **C5**
	10.2.0.0/16	65002	**I2**	**I2**, **C6**
	10.3.0.0/16	65003	**I3**	**I3**
	10.2.8.0/24	65003,65008	**I3**	**C8**
C8	10.1.0.0/16	65003,65001	**I3**	**I1**, **C4**, **C5**
	10.2.0.0/16	65003,65002	**I3**	**I2**, **C6**, **C7**
	10.3.0.0/24	65003	**I3**	**I3**, **C7**
	10.2.6.0/24	65002,65006	**I2**	**C6**

Figure 6.5: Forwarding tables under full-aggregation for Figure 6.3.

6.2 A simplified model of Route Aggregation in eBGP.

In the previous section, we summarized the context in which we will reason and provided the corresponding forwarding tables obtained under a full-aggregation scenario for Figure 6.4. We will now define a prebimonoid for which left equations over square matrices decribe the same results. In particular, we are interested in the construction of the complete set of compound aggregation rules which would be applied on the set of effective routes originating at a router j as they arrive at router i. In other words, the compound aggregation rules represent those which are effectively applied to various routes originated by some domain as they would traverse every possible sequence of routers. The fixpoints will encode, for any pair of physical vertices, the best compound aggregation rules that are applied between them. As a simplification, the routes are limited to three pieces of information; the `NLRI`, the `AS_SEQUENCE` and the advertising peer address. Formally, routes are drawn from the set

$$\mathcal{R}outes_{ebgp} = \mathcal{P} \times \text{list}(ASN) \times \mathcal{V}$$

The set of prefixes is assumed to include an element $\bar{l}$ which is disjoint from any other prefix. This will allow us to express the absence of aggregation rules but the possibility to update the `AS_SEQUENCE` and the advertising peer nonetheless.

$$\forall\, p \in \mathcal{P} \;:\; p \neq \bar{l} \;\Rightarrow\; \neg(p \sqsubseteq \bar{l}) \;\wedge\; \neg(\bar{l} \sqsubseteq p)$$

We assume that the subprefix relation $\sqsubseteq$ satisfies REFLEXIVITY and TRANSITIVITY. Furthermore, there are some prefixes which are mutually disjoint, for instance e.g. `10.1.0.0/16` and `10.2.0.0/16`. This means that the relation $\sqsubseteq$ is a partial preorder. The `AS_SEQUENCE` is represented by lists of ASN with the preference given to shorter lists as described by the relation $l_1 \lesssim_{\text{length}} l_2 \Leftrightarrow \text{length}(l_1) \leqslant \text{length}(l_2)$. Under this definition, note that the empty list is preferred over any other. This relation can be shown to satisfy REFLEXIVITY, TRANSITIVITY and TOTALITY, thus making it a total preorder. Finally, the advertising peer addresses are represented by the set of vertices of the graph, $\mathcal{V}$, for which we assume a total order relation $\leqslant_{\mathcal{V}}$.

From this relation, we construct a partial preorder which captures the preference in terms of the Decision Process. Given two routes matching the prefix of a given destination the relation $\lesssim_{DP}$ gives the best route for identical `NLRI`.

$$\lesssim_{DP} \;\equiv\; = \;\vec{\times}\; \lesssim_{\text{length}} \;\vec{\times}\; \leqslant_{\mathcal{V}} \tag{6.1}$$

Any preorder relation has a strict version associated with it. We will use the strict version, $<_{DP}$ which can be defined by

$$<_{DP} \;\equiv\; = \;\vec{\times}\; \lesssim_{\text{length}} \;\vec{\times}\; <_{\mathcal{V}}$$

Let us first formalize the behaviour of BGP speakers in terms of sets of rules. We will call the initial routes, denoted by $\mathcal{I}$, the routes that a BGP speakers begins with prior to any received advertisement from their external peers and forms its initial set of effective routes. We will include additional assumptions along this section concerning this element as we define each aspect of the Decision Process. This special set of routes will be used to force the activation of all the aggregation rules of each router. Once the compound aggregation rules have been computed, the effective initial routes at each routers could be combined with the compound aggregation rules computed to produce the sets of effective routes which exist between any two pair of routers. This origination behaviour is envisioned to be expressed by means of other constructs from the Metarouting arsenal. We specifically have semi-modules [8] in mind, which have been illustrated to be suitable for expressing the attachment of logical information (e.g. network prefixes) to physical vertices (e.g. BGP speakers or domains). In the remainder of this chapter, we will conflate the concept of best rules with that of best routes. Despite a terminological distinction used in the networking community, there is no formal difference between them in our model. Given that we reduce domains to single BGP speakers, the initial routes can be used to represent statically configured routes as well as those originated from within the domain. The common ground set which we will use is

$$\mathcal{S}_E = \left\{ \min_{\lesssim_{DP}} (R) \mid R \subseteq \mathcal{R}outes_{ebgp} \right\} \cup \{\mathcal{I}\} \tag{6.2}$$

The minimum operation with respect to $\lesssim_{DP}$ is applied on each possible set of routes in order to guarantee that certain properties hold within the structure we define. In practice this only corresponds to the fact that we consider only best routes in every possible sets involved. In particular, we assume that the set of initial routes satisfies

$$\min_{\lesssim_{DP}} (\mathcal{I}) = \mathcal{I} \tag{6.3}$$

This is not a very strong constraint on the initial routes given that it only means that they must form a set containing best routes for each distinct `NLRI`. As a courtesy of Lemma 4.18, the minimum operations can be repeatedly applied for any two elements A and B drawn from Set (6.2) without altering the result.

$$\min_{\lesssim_{DP}} (A \cup B) = \min_{\lesssim_{DP}} \left(\min_{\lesssim_{DP}} (A) \cup B \right) \tag{6.4}$$

However given the presence of the set $\mathcal{I}$, we need to complete the definitions to handle its presence in the terms involved. Formally, we assume for any $\min_{\lesssim_{DP}} (A) = A \subseteq \mathcal{R}outes_{ebgp}$ that

$$\min_{\lesssim_{DP}} (A \cup \mathcal{I}) = \min_{\lesssim_{DP}} (A) \cup \mathcal{I} \tag{6.5}$$

The interpretation is that whenever the best routes have to be selected by the Decision Process, we can delay the selection among the initial routes to a later point. This choice stems from the decision to separate between aggregation and origination of routes and will become apparent by the end of this section.

Throughout its execution, a BGP speaker i will produce its effective routes based on those of its peers q after they have applied some transformation in terms of aggregation and modification of attributes. We can formulate this behaviour where the initial routes are kept in the set of effective routes at all time

$$\mathbf{eroutes}_i = \min_{\lesssim_{DP}} \left(\bigcup_q f_{i,q} \left(\mathbf{eroutes}_q \right) \right) \cup \mathcal{I} \tag{6.6}$$

According to Definition (6.1), the result of applying the minimum operation with respect to $\lesssim_{DP}$ on the input set I is the set of best routes for each distinct `NLRI` based on their attributes. The functions $f_{i,q}$ express the transformation of routes according to the aggregation rules R_q configured at q and whether i is a connected peer of q. In terms of the Decision Process, those functions represent Phase III at peer q while the application of $\min_{\lesssim_{DP}}$ represents Phase II at peer i. Phase I is not represented in our model given that we do not include the `LOCAL_PREF` attribute. The functions are defined as

$$f_{i,q}(E) = \begin{cases} \varnothing & \text{i is not an peer of q} \\ \min_{\lesssim_{DP}} \left(\mathbf{aggregate-routes}_{i,q}(E) \cup \mathbf{remaining-routes}_{i,q}(E) \right) & \text{i is an peer of q} \end{cases} \tag{6.7}$$

We consider that among the resulting routes, only the best ones are advertised to each peer i. In particular, given that a router cannot be engaged in a peering session with itself, the term of the union in Definition (6.6) where $q = i$ will be the empty set. In practice, a BGP speaker only advertises one route per `NLRI`. This allows us to express the peer relations which can exist between routers. It is expected that a router receives route information from another router only if they are engaged in an external peering session. On the other hand, if two routers are external peers of each other, peer q will perform the aggregation and modification of its set of effective routes and provide them to peer i. In practice the set of routes effectively received by different peers of q could be different based on the import and export policies between them, a consideration we do not include in our model. This means that we can either consider that router i receives all the advertised routes of q or none. Consider a router q of a domain which has been allocated the ASN a. Given a set of aggregation rules R_q configured at router q and a set of routes E, the transformation that set E undergoes results in a set where

- All the routes (n_e, l_e, v_e) from E that fall under the prefix of an aggregation rule, $n_e \sqsubset n_a \in R_q$, are replaced by a unique route with prefix n_a which has an `AS_SEQUENCE` containing only a and q as the advertising peer. In this case, we follow a guideline pointed out in [36] whereby child routes must have a strictly more specific prefix than aggregate routes to activate them.

$$\mathbf{aggregate-routes}_{i,q}(E) = \{(n_a, [\mathsf{a}], q) \mid (n_e, l_e, v_e) \in E \ : \ \exists\ n_a \ \in \ R_q \ : \ n_e \sqsubset n_a\}$$

- All the routes (n_e, l_e, v_e) from E for which there is no aggregation rule with a covering prefix, $\nexists\ n_a \ \in \ R \ : \ n_e \sqsubset n_a$, have the list $[a]$ concatenated to their `AS_SEQUENCE` and their advertising peer set to q.

$$\mathbf{remaining-routes}_{i,q}(E) = \{(n_e, [\mathsf{a}] + + l_e, q) \mid (n_e, l_e, v_e) \in E \ : \ \nexists\ n_a \ \in \ R_q \ : \ n_e \sqsubset n_a\}$$

Consider the routers **I1**, **I2** and **I3** for which we assume the use of a unique aggregation rule for their allocated prefix. Furthermore, suppose that the set of effective routes for **I1** and **I3** at some point is

$$\begin{aligned}\mathbf{eroutes}_{\mathbf{I1}} &= \{(\texttt{10.1.4.0/24}, [\texttt{65004}], \mathbf{C4}), (\texttt{10.1.5.0/24}, [\texttt{65005}], \mathbf{C5}), (\texttt{10.3.0.0/16}, [\texttt{65003}], \mathbf{I3})\} \cup \mathcal{I}\\ \mathbf{eroutes}_{\mathbf{I3}} &= \{(\texttt{10.3.7.0/16}, [\texttt{65007}], \mathbf{C7}), (\texttt{10.2.8.0/24}, [\texttt{65008}], \mathbf{C8})\} \cup \mathcal{I}\end{aligned}$$

The sets of routes received by **I2** from **I1** and **I3** are obtained by taking the union of the sets of aggregate and unaggregated routes from those respective peers

$$\begin{aligned}f_{\mathbf{I2},\mathbf{I1}}(\mathbf{eroutes}_{\mathbf{I1}}) &= \{(\texttt{10.1.0.0/16}, [\texttt{65001}], \mathbf{I1}), (\texttt{10.3.0.0/16}, [\texttt{65001}, \texttt{65003}], \mathbf{I1})\}\\ f_{\mathbf{I2},\mathbf{I3}}(\mathbf{eroutes}_{\mathbf{I3}}) &= \{(\texttt{10.3.0.0/16}, [\texttt{65003}], \mathbf{I3}), (\texttt{10.2.8/24}, [\texttt{65003}, \texttt{65008}], \mathbf{I3})\}\end{aligned}$$

Given the input set formed by the union of those received routes, after running the Decision Process at **I2**, the new set of routes will be based on that of those advertised by its peers. In particular, we can omit the term for $q = \mathbf{I2}$ given that it will be the empty set by Definition (6.7).

$$\begin{aligned}\mathbf{eroutes}_{\mathbf{I2}} &= \min_{\lesssim_{DP}} \left(f_{\mathbf{I2},\mathbf{I1}}(\mathbf{eroutes}_{\mathbf{I1}}) \cup f_{\mathbf{I2},\mathbf{I3}}(\mathbf{eroutes}_{\mathbf{I3}})\right) \cup \mathcal{I}\\ &= \{(\texttt{10.1.0.0/16}, [\texttt{65001}], \mathbf{I1}), (\texttt{10.3.0.0/16}, [\texttt{65003}], \mathbf{I3}), (\texttt{10.2.8/24}, [\texttt{65003}, \texttt{65008}], \mathbf{I3})\} \cup \mathcal{I}\end{aligned}$$

We will now attempt to formalize the behaviour expressed earlier in terms of the solutions to recurrence equations. Given that all the definitions from Chapter 4 regarding square matrices and recurrence equations are based on an underlying prebimonoid, we shall endeavour to capture the same behaviour by identifying a suitable definition for the additive and multiplicative laws. The approach based on square matrices has the result of producing more best routes than the one described by Equation (6.6). We will use a reduction function defined on the square matrices which will only keep the best routes across all destinations j.

$$\phi_i(\mathbf{L}) = \min_{\lesssim_{DP}} \left(\bigcup_j \mathbf{L}_{i,j}\right) \tag{6.8}$$

The pivotal relation we seek to explore is whether the fixpoint to given recurrence equations encode the same best rules as those computed according to Equation (6.6). Our goal for the remainder of this section will be to define a prebimonoid such that, at each iteration k, each row of matrix $\mathbf{L}^k$ and the corresponding entry of $\mathbf{eroutes}^k$ encode the same best routes.

$$\phi_i\left(\mathbf{L}^k\right) = \mathbf{eroutes}_i^k \tag{6.9}$$

The two recurrence relations describing the computation of the fixpoint $\mathbf{L}$ and the effective routes exhibit an interesting structural difference. The computation of the fixpoint matrix keeps separate the best routes based on their associated physical destination while the computation of effective routes dismisses this information to focus only on the best routes for distinct `NLRI`. Relation (6.9) effectively provides us with the means to derive an efficient algorithm for the computation of the relevant information from the computation of the fixpoint. While the aggregation problem can be studied algebraically, all the corresponding definitions can be used as the basis for the implementation of a protocol to perform the exchange and aggregation of route information *à la* BGP. We will give another illustration to this interesting fact in Section 6.3.

$$\mathbf{L}_{i,j}^0 = \mathbf{I}_{i,j} \tag{6.10}$$

$$\mathbf{L}_{i,j}^{k+1} = \sum_q^{\oplus_E} \left(\mathbf{A}_{i,q} \otimes_E \mathbf{L}_{q,j}^k\right) \oplus_E \mathbf{I}_{i,j} \tag{6.11}$$

$$\mathbf{eroutes}_i^0 = \mathcal{I} \tag{6.12}$$

$$\mathbf{eroutes}_i^{k+1} = \min_{\lesssim_{DP}} \left(\bigcup_q f_{i,q}\left(\mathbf{eroutes}_q^k\right)\right) \cup \mathcal{I} \tag{6.13}$$

$$\begin{tikzcd}
\mathbf{L}^k \arrow[r, "\phi_i"] \arrow[d, "(1.14)"'] & \mathbf{eroutes}^k \arrow[d, "(1.16)"] \\
\mathbf{L}^{k+1} \arrow[r, "\phi_i"] & \mathbf{eroutes}^{k+1}
\end{tikzcd}$$

Figure 6.6: Relationship between two successive matrices produced by Equation (6.11) and the computation of effective routes according to Equation (6.13). By applying the function ϕ_i for all sources i one can produce the entire contents of $\mathbf{eroutes}^k$ from $\mathbf{L}^k$.

We consider in our approach that route aggregation is a multipath problem given that multiple routes with distinct prefixes can be used to reach a domain, a useful mean of expressing multi-homing scenarios in our approach. The elementary objects manipulated are elements from the set (6.2). We will place the selection of best routes in the additive law while the aggregation and modification of routes will be defined in the multiplicative law. Given two sets of routes A and B, the rule for selecting the best ones among them is given by

$$A \oplus_E B = \begin{cases} \min_{\lesssim_{DP}} (A \cup B) & A \neq \mathcal{I} \neq B \\ A \cup B & \text{otherwise} \end{cases} \tag{6.14}$$

In particular, we separate the case based on the initial routes. The choice is consistent with Property (6.5) by which we can obtain the appropriate treatment from the first case of the definition.

$$\min_{\lesssim_{DP}} (A \cup \mathcal{I}) = \min_{\lesssim_{DP}} (A) \cup \mathcal{I} = A \cup \mathcal{I}$$

When two routes pertain to the same physical destination, it is only necessary to keep the best ones based on their attributes. In the case of multi-homing, if two routes correspond to the same physical destination, only the best routes for distinct `NLRI` will be kept. These two situations are depicted below between the respective routers involved in them. The route information corresponding to the two scenarios are mapped to the relevant portion of Figure 6.4.

We can express the process by which a router aggregates and modifies the attributes of routes by taking all the aggregation rules from a set A and apply them to the set B. The expression of the result can be split in two intermediate definitions.

$$\begin{aligned}
\mathbf{aggregate}(A,B) &= \{(n_a, l_a, v_a) \in A \mid (n_b, l_b, v_b) \in B \ \wedge \ n_b \ \sqsubset \ n_a\} \\
\mathbf{remaining}(A,B) &= \{(n_b, l_a{+}{+}l_b, v_a) \mid (n_a, l_a, v_a) \in A \ \wedge \ (n_b, l_b, v_b) \in B \ : \ \nexists \, (n_a, l_a, v_a) \ \in \ A \ : \ (n_b \sqsubset n_a)\}
\end{aligned}$$

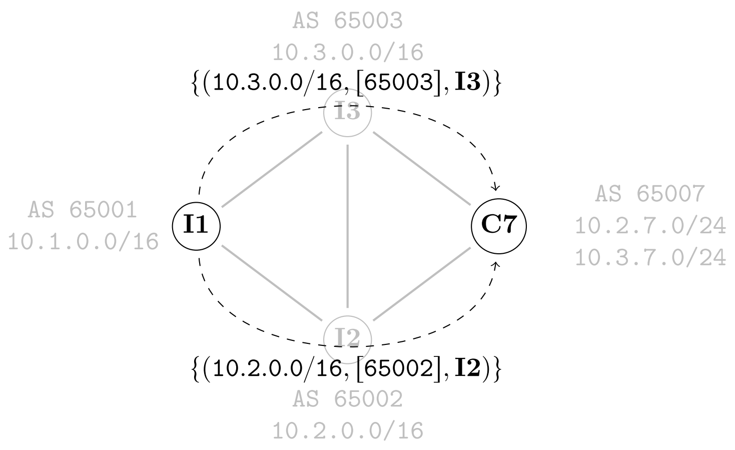
The first set contains the aggregate routes of A that are activated by the presence of one or more child routes in B. As a consequence, the activated aggregate routes, whose `AS_SEQUENCE` consists of the ASN of the router and has itself as the advertising peer, find their way into the set of routes to be advertised. Furthermore, all the remaining routes, which do not fall under any configured aggregation rule, end up in the set to be advertised after their attributes are modified accordingly. We include the specific treatment for $\mathcal{I}$ in the definition of the multiplicative law

$$A \otimes_E B = \begin{cases} A & B = \mathcal{I} \\ B & A = \mathcal{I} \\ \min_{\lesssim_{DP}} (\mathbf{aggregate}(A,B) \cup \mathbf{remaining}(A,B)) & \text{otherwise} \end{cases} \tag{6.15}$$

In particular, we define $\mathcal{I}$ as the identity for this operation. Given that this element plays a role in the origination of routes, the least restrictive assumption on it is that it activates all the aggregate routes. The course of action here is to activate all the aggregation rules that are involved so that all the routers have the entire set of aggregation rules used by their peers. Going back to the interpretation of multiplication presented earlier, the best rules known between **I1** and **C7** for the two respective prefixes

$$\{(\texttt{10.2.0.0/16}, [\texttt{65002}], \mathbf{I2}), (\texttt{10.3.0.0/16}, [\texttt{65003}], \mathbf{I3})\}$$

Suppose now that **I3** originates a route for one of its prefixes, `10.2.7.0/24`. We can obtain the best route for the originating domain through the multiplication of the compound aggregation rule by the



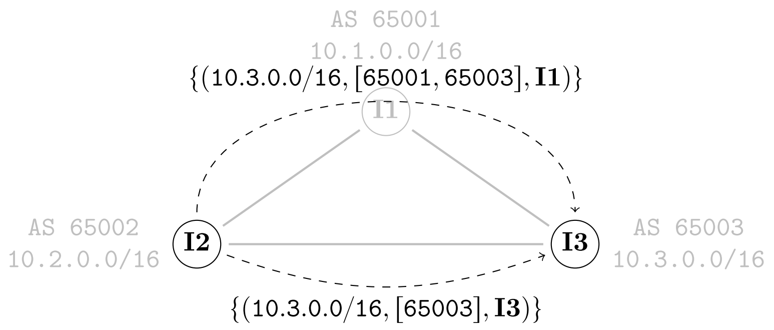

AS 65001
10.1.0.0/16
{(10.3.0.0/16, [65001, 65003], I1)}
I1
AS 65002
10.2.0.0/16
I2
I3
AS 65003
10.3.0.0/16
{(10.3.0.0/16, [65003], I3)}

Figure 6.7: Interpretation of the additive law in terms of graph. The upper graph illustrates the multi-homing scenario while the lower graph presents a simple case of route selection.

originated route. The result will be the best route available to reach the network of Customer 7 but **I1** will never have heard of the possibility to reach it through the prefix `10.3.0.0/16`.

$$\begin{aligned}&\{(\mathsf{10.2.0.0/16}, [\mathsf{65002}], \mathbf{I2}), (\mathsf{10.3.0.0/16}, [\mathsf{65003}], \mathbf{I3})\} \otimes_E \{(\mathsf{10.2.7.0/24}, [\mathsf{65007}], \mathbf{C7})\}\\&= \{(\mathsf{10.2.0.0/16}, [\mathsf{65002}], \mathbf{I2})\}\end{aligned}$$

In this form, the computation of the aggregation rules is biased towards a particular way origination is performed. When router **C7** originates the route to this prefix, it would be advertised only on the link to the relevant provider, namely **I2**. In this case, the result we gave above is consistent with the routes discovered by BGP. Depending on the particular configuration of **C7**, it might begin advertising this route to both its providers, in which case it will pass unaggregated through **I3** and arrive as such at **I1**. Upon comparing this more specific route with the aggregated one received from **I2**, the former would be favoured.

On the other hand, suppose now that **I3** originates a route for each of its prefixes, `10.2.7.0/24` and `10.3.7.0/24`, then the best routes to reach it are the aggregate routes provided by the respective providers of Customer 7.

$$\begin{aligned}&\{(\mathsf{10.2.0.0/16}, [\mathsf{65002}], \mathbf{I2}), (\mathsf{10.3.0.0/16}, [\mathsf{65003}], \mathbf{I3})\} \otimes_E \{(\mathsf{10.2.7.0/24}, [\mathsf{65007}], \mathbf{C7}), (\mathsf{10.3.7.0/24}, [\mathsf{65007}], \mathbf{C7})\}\\&\qquad = \{(\mathsf{10.2.0.0/16}, [\mathsf{65002}], \mathbf{I2}), (\mathsf{10.3.0.0/16}, [\mathsf{65003}], \mathbf{I3})\}\end{aligned}$$

For the scenarios which we are interested in, the routers involved are configured with a certain ASN, which they prepend to all the routes, while systematically setting the advertising peer address to their own before sending the routes. Furthermore, the routers can be configured with a set of aggregation rules that they apply to route information as it flows through them. On one side we have a set of rules represented by a set of prefixes on which aggregation should be performed together with an integer for the ASN and some identifier for the advertising peer. On the other side, we have sets of routes described as the association of a prefix with a list of ASN and an identifier for the advertising peer. In our approach, we choose to represent each aggregation rule by a route whose prefix is that of the aggregation rule, has an

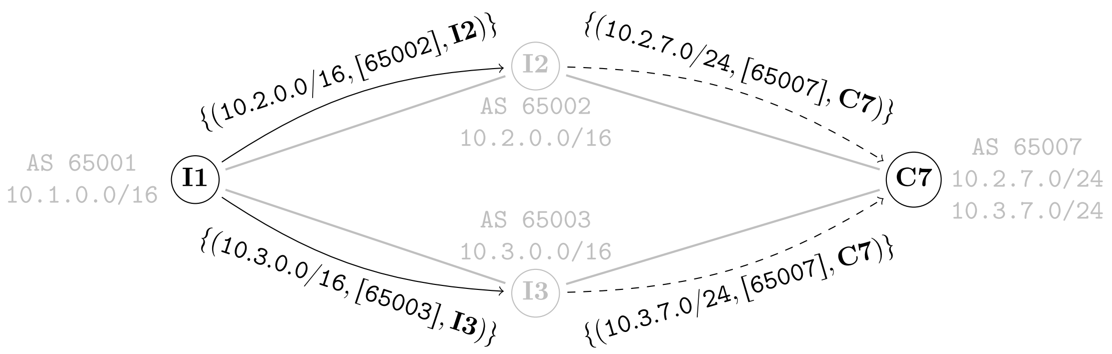

{(10.2.0.0/16, [65002], I2)}
{(10.2.7.0/24, [65007], C7)}
I2
AS 65002
10.2.0.0/16
AS 65001
10.1.0.0/16
I1
C7
AS 65007
10.2.7.0/24
10.3.7.0/24
AS 65003
10.3.0.0/16
{(10.3.0.0/16, [65003], I3)}
I3
{(10.3.7.0/24, [65007], C7)}

Figure 6.8: Interpretation of the multiplicative law based on graph.

`AS_SEQUENCE` composed of the ASN configured and the advertising peer address set to the vertex which represents the aggregating router. If a router j represents a domain which has been assigned the ASN(j) and is configured to use the set of aggregation rules R_j, we can construct the multiplicative operand representing the processing performed by peer j on its best routes before advertising them according to

$$A = \begin{cases} \varnothing & i \text{ is not a peer of } j \\ \{(\bar{l}, \text{ASN(j)}, j)\} & i \text{ is a peer of } j \ \wedge \ R_j = \varnothing \\ \{(n_a, \text{ASN(j)}, j) \mid n_a \in R_j\} & i \text{ is a peer of } j \ \wedge \ R_j \neq \varnothing \end{cases} \tag{6.16}$$

In the case where the two routers are not peers of each other, no route information is exchanged between them. Even in the presence of a peering session, there might be no aggregation rules configured, a case for which we use the special element $\bar{l}$ as the unique aggregation rule introduced. Given that it is disjoint from any other prefix, no route from the other operand B will have a matching subprefix. This means that all the routes from B will simply have their `AS_SEQUENCE` and advertising peer updated. This behaviour is the one to be expected when route aggregation is not used and correspond to the elementary aspects of the exchange of route information in the BGP protocol. The regime of no-aggregation which was discussed in Chapter 3 can be represented by using Construction (6.16) with all the R_j set to the empty set.

The approaches from Definition (6.11) and Definition (6.13) will be related further after we have proved that our defined structure satisfies various properties. The complete structure used to model eBGP uses the ground set which is given by Definition (6.2) and has the additive law of Definition (6.14) and the multiplicative law of Definition (6.15). We show that the resulting structure is an IDEMPOTENT prebimonoid by proving that it satisfies all the relevant properties in the two subsequent lemmas.

$$\text{eBGP} = \left(\left\{ \min_{\lesssim_{DP}} (R) \mid R \subseteq \mathcal{R}outes_{ebgp} \right\} \cup \{\mathcal{I}\}, \oplus_E, \otimes_E \right) \tag{6.17}$$

Lemma 6.2.1. *The additive law $\oplus_E$ is* ASSOCIATIVE, COMMUTATIVE, IDEMPOTENT *and admits $\varnothing$ as an* IDENTITY.

Proof. • ASSOCIATIVITY. The proof of ASSOCIATIVITY directly follows from ASSOCIATIVITY of $\cup$ and (6.4).

$$\begin{aligned} (A \oplus_E B) \oplus_E C &= \min_{\lesssim_{DP}} \left(\min_{\lesssim_{DP}} (A \cup B) \cup C \right) \\ &= \min_{\lesssim_{DP}} ((A \cup B) \cup C) && \text{(By (6.4))} \\ &= \min_{\lesssim_{DP}} (A \cup (B \cup C)) && (\text{ASSOCIATIVITY of } \cup) \\ &= \min_{\lesssim_{DP}} \left(A \cup \min_{\lesssim_{DP}} (B \cup C) \right) && \text{(By (6.4))} \\ &= A \oplus_E (B \oplus_E C) \end{aligned}$$

- COMMUTATIVITY of $\oplus_E$ follows from the COMMUTATIVITY of $\cup$

$$A \oplus_E B = \min_{\lesssim_{DP}} (A \cup B) = \min_{\lesssim_{DP}} (B \cup A) = B \oplus_E A$$

- IDEMPOTENCY holds given the IDEMPOTENCY of $\cup$ and the fact that any element from the ground set satisfies $A = \min_{\lesssim_{DP}} (A)$.

$$\begin{aligned} A \oplus_E A &= \min_{\lesssim_{DP}} (A \cup A) \\ &= \min_{\lesssim_{DP}} (A) \\ &= A \end{aligned}$$

- The empty set is an IDENTITY for $\oplus_E$ given that it is an identity for the underlying $\cup$ operation.

$$\begin{aligned} A \oplus_E \varnothing = \min_{\lesssim_{DP}} (A \cup \varnothing) &= \min_{\lesssim_{DP}} (A) = \min_{\lesssim_{DP}} (\varnothing \cup A) = \varnothing \oplus_E A \\ &= A \end{aligned}$$

□

Lemma 6.2.2. *The multiplicative law $\otimes_E$ admits the empty set as an* ANNIHILATOR, *$\mathcal{I}$ as an* IDENTITY.

Proof. • The proof that $\varnothing$ is an ANNIHILATOR for $\otimes_E$ works by injecting it into Definition (6.15). If $A = \varnothing$, then the two sets

$$\begin{aligned} \textbf{aggregate}\,(\varnothing, B) &= \{(n_a, l_a, v_a) \in \varnothing \mid (n_b, l_b, v_b) \in B \ \wedge \ n_b \ \sqsubset \ n_a\} \\ \textbf{remaining}\,(\varnothing, B) &= \{(n_b, l_a{+\!+}l_b, v_a) \mid (n_a, l_a, v_a) \in \varnothing \ \wedge \ (n_b, l_b, v_b) \in B \ : \ \nexists\, (n_a, l_a, v_a) \ \in \ \varnothing \ : \ (n_b \sqsubset n_a)\} \end{aligned}$$

are empty and their union is as well.

$$\varnothing \otimes_E B = \varnothing \cup \varnothing = \varnothing$$

Similarly, if $B = \varnothing$, then two sets

$$\begin{aligned} \textbf{aggregate}\,(A, \varnothing) &= \{(n_a, l_a, v_a) \in A \mid (n_b, l_b, v_b) \in \varnothing \ \wedge \ n_b \ \sqsubset \ n_a\} \\ \textbf{remaining}\,(A, \varnothing) &= \{(n_b, l_a{+\!+}l_b, v_a) \mid (n_a, l_a, v_a) \in A \ \wedge \ (n_b, l_b, v_b) \in \varnothing \ : \ \nexists\, (n_a, l_a, v_a) \ \in \ A \ : \ (n_b \sqsubset n_a)\} \end{aligned}$$

are also empty and so is their union.

$$A \otimes_E \varnothing = \varnothing \cup \varnothing = \varnothing$$

- The proof that $\mathcal{I}$ is an IDENTITY follows directly from Definition (6.15).

□

Conveniently enough, the definitions of the two laws closely resemble the definitions of the computation of the effective routes. We can formally relate multiplication with the application of the functions $f_{i,j}$ as the following lemma.

Lemma 6.2.3. *Given a function $f_{i,j}$ and a set A constructed according to Rule (6.16), we have*

$$f_{i,j}\,(B) = A \otimes_E B$$

Proof. • If i is not a peer of j, the result follows from Definition (6.7) and because the empty set $\varnothing$ is an ANNIHILATOR for $\otimes_E$

$$f_{i,j}\,(B) = \varnothing = \varnothing \otimes_E B$$

- If i is a peer of j but the set of aggregation rules of the latter is empty, the function can be simplified as

$$f_{i,q}\,(B) = \textbf{remaining} - \textbf{routes}_{i,q}(B) \qquad R_q = \varnothing$$

with the resulting routes being all those from B with their `AS_SEQUENCE` extended by the ASN of j and the advertising peer set to j. On the other hand, the operand A for multiplication only contains one entry and we have that $\{(\bar{l}, \mathrm{ASN}(j), j)\} \otimes_E B$ contains all the routes from B with the same modifications.

- If i is a peer of j and the set of aggregation rules is not empty, we can see that the results of the following two expression are identical

$$\begin{aligned} f_{i,j}(B) &= \textbf{aggregate} - \textbf{routes}_{i,j}(B) \cup \textbf{remaining} - \textbf{routes}_{i,j}(B) \\ A \otimes_E B &= \textbf{aggregate}(A,B) \cup \textbf{remaining}(A,B) \end{aligned}$$

□

We can use Prebimonoid 6.17 to encode any aggregation scenario described by the allocation of ASN to routers which use certain aggregation rules and are engaged in peering sessions. The encoding takes the form of an adjacency matrix with elements in Prebimonoid 6.17 obtained by applying Construction (6.16) to produce each of its entries. In Chapter 4 we described the convergence of the Bellman-Ford Algorithm under an assumption of LEFT-INFLATIONARITY. However, the entries $\mathbf{A}_{i,j}$ produced by the application of Construction (6.16) are not LEFT-INFLATIONARY in general as illustrated by the following counter-example

$$\begin{aligned} A &= \{(\texttt{10.1.0.0/16}, [\texttt{65001}], \mathbf{I1})\} \\ B &= \{(\texttt{10.1.4.0/24}, [\texttt{65004}], \mathbf{C4}), (\texttt{10.1.5.0/24}, [\texttt{65005}], \mathbf{C5})\} \\ A \otimes_E B &= \{(\texttt{10.1.0.0/16}, [\texttt{65001}], \mathbf{I1})\} \\ B \oplus_E (A \otimes_E B) &= \{(\texttt{10.1.0.0/16}, [\texttt{65001}], \mathbf{I1}), \{(\texttt{10.1.4.0/24}, [\texttt{65004}], \mathbf{C4}), (\texttt{10.1.5.0/24}, [\texttt{65005}], \mathbf{C5})\}\} \end{aligned}$$

It is not possible to make any claims as to the convergence based on the $\lesssim_{DP}$ relation. However it is known that path-vectoring methods do converge and it can be shown by relying on an alternate order relation that is reminiscent on the preference according to the LMP-rule. In [36], the authors claim that the use of sink routes with the lowest administrative distance is sufficient to guarantee convergence to loop-free forwarding routes. However this solution only prevent traffic from being forwarded endlessly along a cycle of routers. It does not give any insight as to why forwarding entries resulting in the creation of a loop have been installed in forwarding tables. Another possibility would be to note that for every physical destination, the routes can only have their associated prefixes generalized or their metric reduced. The formalisation of this observation would rely on a different order relation between routes, namely

$$\lesssim_{LMP} \quad \equiv \quad \sqsubseteq \; \vec{\times} \; \lesssim_{\text{length}} \; \vec{\times} \; \leqslant_{\mathcal{V}}$$

This order relation could be lifted to sets of routes so that one set is greater than another if-and-only-if all the routes from the former have more specific or disjoint prefixes from all the routes in the latter. This new relation would satisfy INFLATIONARITY but the context would be different than the computation of a fixpoint to a recurrence equation. Based on the natural order relation, we are unable to make any claims as to the convergence of the Bellman-Ford Algorithm 4 under route aggregation. In other words, it is unclear whether any configuration of aggregation rules leads to a convergence of each router to its best routes and how many iterations are needed to reach it.

On another aspect, the entries of the adjacency matrices satisfy LEFT-DISTRIBUTIVITY. We will rely on this result to prove the adequacy of our approach with the description of the computation of effective routes.

Lemma 6.2.4. *Given a set of BGP speakers $\mathcal{V}$ engaged in external peering sessions represented by a set of edges $\mathcal{E}$. Each BGP speaker j is configured with an AS Number ASN(j) and is configured with a set of aggregation rules R_j. The operands A obtained by applying Construction (6.16) are* LEFT-DISTRIBUTIVE *for all $B, C \in \mathcal{S}_E$*

$$A \otimes_E (B \oplus_E C) = (A \otimes_E B) \oplus_E (A \otimes_E C) \tag{6.18}$$

Proof. We break down the proof according to the cases of Definition (6.16).

- When i is not a peer of j, the operand is the empty set which is an ANNIHILATOR for the multiplication.

$$\begin{aligned} \varnothing \otimes_E (B \oplus_E C) &= \varnothing \\ (\varnothing \otimes_E B) \oplus_E (\varnothing \otimes_E C) &= \varnothing \oplus_E \varnothing = \varnothing \end{aligned}$$

- When i is a peer of j but no aggregation rules are used by j, the operand A is limited to $\{(\bar{l}, [\mathrm{ASN}(j)], j)\}$ which has the effect of changing the attributes of each individual route from B without replacing any of them by an aggregate route. In other words, we have

$$
\begin{aligned}
A \otimes_E B &= \min_{\lesssim_{DP}} (\textbf{remaining}\,(A, B)) \\
A \otimes_E C &= \min_{\lesssim_{DP}} (\textbf{remaining}\,(A, C)) \\
A \otimes_E (B \oplus_E C) &= \min_{\lesssim_{DP}} (\textbf{remaining}\,(A, B \oplus_E C))
\end{aligned}
$$

  Suppose the routes $r_b = (n_b, l_b, v_b)$ in B and $r_c = (n_c, l_c, v_c)$ in C. Note that we have directly $r_b \in (A \otimes_E B)$ and $r_c \in (A \otimes_E C)$. We consider two distinct cases based on the relation between r_b and r_c.

  If the prefixes of the two routes are different, $n_b \neq n_c$, then both routes will appear in $B \oplus_E C$. After having their attributes updated, the prefixes will still be different and therefore the resulting routes will both appear in $A \otimes_E (B \oplus_E C)$. Furthermore, they will also appear $(A \otimes_E B) \oplus_E (A \otimes_E C)$.

  On the other hand, if the prefixes of the routes are identical, the value of their metric will determine which will appear in $B \oplus_E C$. Suppose without any loss of generality that $r_b \lesssim_{DP} r_c$. This means that

$$l_b <_{\text{length}} l_c \;\vee\; (l_b \sim_{\text{length}} l_c \;\wedge\; v_b \leqslant_{\mathcal{V}} v_c)$$

  where $\sim_{\text{length}}$ means that the two lists are of equal length. After their attributes are updated, this relation is preserved

$$(n_b, [\mathrm{ASN}(\mathtt{j})] + +l_b, j) \lesssim_{DP} (n_c, [\mathrm{ASN}(\mathtt{j})] + +l_c, j)$$

  given that $n_b = n_c$ and that the concatenation of a common ASN to the `AS_SEQUENCE` of two routes does not change the preference on the resulting `AS_SEQUENCE`.

$$
\begin{aligned}
l_b <_{\text{length}} l_c &\;\Rightarrow\; ([\mathrm{ASN}(\mathtt{j})] + +l_b) <_{\text{length}} ([\mathrm{ASN}(\mathtt{j})] + +l_c) \\
l_b \sim_{\text{length}} l_c &\;\Rightarrow\; ([\mathrm{ASN}(\mathtt{j})] + +l_b) \sim_{\text{length}} ([\mathrm{ASN}(\mathtt{j})] + +l_c)
\end{aligned}
$$

  As a consequence, only the route resulting from r_b will appear in the two hand sides of Equation (6.18) if $r_b <_{DP} r_c$ while both resulting routes will end up in both hand sides as well if the two routes are equivalent in terms of their attributes, $r_b \sim_{DP} r_c$.

- In the most general case, the set of aggregation routes A contains a certain number of rules but all of them have identical second and third components. When i is a peer of j and the set A contains r aggregation rules and has the form

$$A = \{(n_{a1}, [\mathrm{ASN}(j)], j), \ldots, (n_{ar}, [\mathrm{ASN}(j)], j)\}$$

  We will again consider two routes $r_b = (n_b, l_b, v_b)$ in B and $r_c = (n_c, l_c, v_c)$ in C. Note that we have directly $r_b \in (A \otimes_E B)$ and $r_c \in (A \otimes_E C)$. We consider two distinct cases based on the relation between r_b and r_c.

  If both have different prefixes, we need only to look for an aggregate route with an identical prefix to one of them. In the case such a route $r = (n, l, v)$ exists in A but there is no matching route in either B or C, both routes r_b and r_c will have their attributes updated and appear in both $A \otimes_E (B \oplus_E C)$ and $(A \otimes_E B) \oplus_E (A \otimes_E C)$. On the other hand, if a matching route exists, the routes resulting from r_b and r_c will not appear in either terms if the prefixes are identical to that of r. Without loss of generality, the route r must be more interesting than the route resulting from r_b given that the concatenation of lists satisfies $l \lesssim_{\text{length}} l{+}{+}l_b$

$$(n, l, v) \lesssim_{DP} (n_b, l{+}{+}l_b, v)$$

  In particular, the routes are equivalent in terms of $\lesssim_{DP}$ if the `AS_SEQUENCE` from route r_b is empty in which case, both the route resulting from r_b and r appear in both hand sides of Equation (6.18). The same argument can be applied to r_c.

  If both routes have identical prefixes, the presence of an aggregate route r matching one matches also the other. As a consequence, r finds its way into both hand sides of Equation (6.18) while r_b

and r_c are present in neither. On the other hand, if no aggregate route matches the prefixes, the relationship in terms of $\lesssim_{DP}$ will determine the outcome. Without loss of any generality, suppose that $r_b \lesssim_{DP} r_c$. As a consequence, the route resulting from r_b appears in the left handside of Equation (6.18). The two resulting routes will preserve this ordering because the concatenation of lists preserves the ordering in terms of $\lesssim_{\text{length}}$

$$\begin{aligned} l_b <_{\text{length}} l_c &\Rightarrow l{+}{+}l_b <_{\text{length}} l{+}{+}l_c \\ l_b \sim_{\text{length}} l_c &\Rightarrow l{+}{+}l_b \sim_{\text{length}} l{+}{+}l_c \end{aligned}$$

We therefore have that the route resulting from r_b will appear in the right hand side of Equation (6.18) because $(n_b, l{+}{+}l_b, v) \lesssim_{DP} (n_c, l{+}{+}l_c, v)$. Furthermore, $r_b \lesssim_{DP} r_c$ imlies that r_b appears in $B \oplus_E C$ and in the left hand side of Equation (6.18).

□

This result allows us to point out the possibility of a minor divergence between our approach and what is expected of BGP speakers. Specifically, we had to handle the case where multiple routes with identical `NLRI` are the best routes whereas BGP is specified so that at most one route is kept. One of the origins of this potential divergence is that the ordering based on the length of the `AS_SEQUENCE` does not satisfy ANTI-SYMMETRY. The particular case would occur if two routes for a given `NLRI` had two equally long `AS_SEQUENCE` and the same advertising peer.

$$(\mathsf{10.2.8.0/24}, [\mathsf{65003}, \mathsf{65008}], \mathbf{I3}) \lesssim_{DP} (\mathsf{10.2.8.0/24}, [\mathsf{65002}, \mathsf{65008}], \mathbf{I3})$$

By virtue of the way the entries of the adjacency matrix are constructed, i.e. Construction (6.16), such a situation is impossible, given that a peer will always prepend its ASN and set itself as the advertising peer. As a consequence, the first element of the `AS_SEQUENCE` must match the ASN assigned to the advertising peer and the second route presented above simply cannot ever be created by any iteration of Equation (6.11) on any scenario encoded according to Construction (6.16).

It is important to note also a potential limitation of the applicability of the approach we propose. In the proof above, we used the fact that the ordering of routes is preserved under modification of their attributes.

$$(n_b, l_b, v_b) \lesssim_{DP} (n_c, l_c, v_c) \Rightarrow (n_b, l{+}{+}l_b, v) \lesssim_{DP} (n_c, l{+}{+}l_c, v)$$

Given that under the relation $\lesssim_{DP}$, the prefixes are identical and the value of the metric of the first route is more interesting than the second one. This situation is based on the MONOTONIC nature of the underlying metric. This property states that when two values of the metric are ordered in some way, their transformation by a common value will preserve the ordering on the resulting values. This property is related to the DISTRIBUTIVITY of a metric with the latter being a sufficient condition for the former. Recent research has shown that metrics which do not satisfy DISTRIBUTIVITY are commonplace in network protocols for example when Quality of Service is involved [52][54]. We will now prove that Relation (6.9) holds when the adjacency matrix is LEFT-DISTRIBUTIVE. At this stage, we are unable to prove or disprove this relation for the case where the underlying metric is not LEFT-DISTRIBUTIVE, a question that we leave for further investigation. With all the properties satisfied by the prebimonoid we defined, we can relate the two approaches by showing that at each iteration k, the results are consistent under the reduction functions ϕ_i.

Theorem 6.2.1. *Given a set of BGP speakers $\mathcal{V}$ engaged in external peering sessions represented by a set of edges $\mathcal{E}$. Each BGP speaker j is configured with an AS Number $ASN(j)$ and with a set of aggregation rules R_j. Given an adjacency matrix $\mathbf{A}$ over Prebimonoid 6.17 where each entry is obtained by using Construction (6.16), we have*

$$\phi_i\left(\mathbf{L}^k\right) = \mathbf{eroutes}_i^k$$

Proof. We proceed by induction on k.

- $k = 0$. For the base case, we have

$$\begin{aligned}
\phi_i\left(\mathbf{L}^0\right) = \min_{\lesssim_{DP}}\left(\bigcup_j \mathbf{L}^0_{i,j}\right) &= \min_{\lesssim_{DP}}\left(\bigcup_j \mathbf{I}_{i,j}\right) && \text{(By Definition (6.10))}\\
&= \min_{\lesssim_{DP}}(\mathcal{I}) && (\mathbf{I}_{i,j} = \mathcal{I},\, \mathbf{I}_{i,i} = \varnothing)\\
&= \mathcal{I} && \text{(By Property (6.3))}\\
&= \mathbf{eroutes}^0_i && \text{(By Definition (6.12))}
\end{aligned}$$

- $k > 0$. The Induction Hypothesis is that for all i we have

$$\phi_i\left(\mathbf{L}^k\right) = \mathbf{eroutes}^k_i$$

  We start from the left hand side of the goal

$$\begin{aligned}
\phi_i\left(\mathbf{L}^{k+1}\right) &= \min_{\lesssim_{DP}}\left(\bigcup_j \mathbf{L}^{k+1}_{i,j}\right)\\
&= \min_{\lesssim_{DP}}\left(\bigcup_j\left(\sum_q^{\oplus_E} \mathbf{A}_{i,q} \otimes_E \mathbf{L}^k_{q,j} \oplus_E \mathbf{I}_{i,j}\right)\right) && \text{(By Definition (6.11))}\\
&= \min_{\lesssim_{DP}}\left(\bigcup_j \min_{\lesssim_{DP}}\left(\bigcup_q \mathbf{A}_{i,q} \otimes_E \mathbf{L}^k_{q,j} \cup \mathbf{I}_{i,j}\right)\right) && \text{(By Definition (6.14))}\\
&= \min_{\lesssim_{DP}}\left(\bigcup_j\bigcup_q \mathbf{A}_{i,q} \otimes_E \mathbf{L}^k_{q,j} \cup \mathbf{I}_{i,j}\right) && \text{(By Property (6.4))}\\
&= \min_{\lesssim_{DP}}\left(\bigcup_q\bigcup_j \mathbf{A}_{i,q} \otimes_E \mathbf{L}^k_{q,j} \cup \mathbf{I}_{i,j}\right) && (\cup\text{-COMMU., ASSOC.})\\
&= \min_{\lesssim_{DP}}\left(\bigcup_q\bigcup_j \mathbf{A}_{i,q} \otimes_E \mathbf{L}^k_{q,j} \cup \mathcal{I}\right) && (\mathbf{I}_{i,j} = \mathcal{I},\, \mathbf{I}_{i,i} = \varnothing)\\
&= \min_{\lesssim_{DP}}\left(\bigcup_q\bigcup_j \mathbf{A}_{i,q} \otimes_E \mathbf{L}^k_{q,j}\right) \cup \mathcal{I} && \text{(By Property (6.5))}\\
&= \min_{\lesssim_{DP}}\left(\bigcup_q \min_{\lesssim_{DP}}\left(\bigcup_j \mathbf{A}_{i,q} \otimes_E \mathbf{L}^k_{q,j}\right)\right) \cup \mathcal{I} && \text{(By Property (6.4))}\\
&= \min_{\lesssim_{DP}}\left(\bigcup_q \mathbf{A}_{i,q} \otimes_E \min_{\lesssim_{DP}}\left(\bigcup_j \mathbf{L}^k_{q,j}\right)\right) \cup \mathcal{I} && \text{(By Lemma 6.2.4)}\\
&= \min_{\lesssim_{DP}}\left(\bigcup_q f_{i,q}\left(\min_{\lesssim_{DP}}\left(\bigcup_j \mathbf{L}^k_{q,j}\right)\right)\right) \cup \mathcal{I} && \text{(By Lemma 6.2.3)}\\
&= \min_{\lesssim_{DP}}\left(\bigcup_q f_{i,q}\left(\mathbf{eroutes}^k_q\right)\right) \cup \mathcal{I} && \text{(By Induction Hypothesis)}\\
&= \mathbf{eroutes}^{k+1}_i && \text{(By Definition (6.13))}
\end{aligned}$$

□

This means that we can study the aggregation problem from the perspective of fixpoints to left equation over Prebimonoid 6.17 and obtain the same best routes as those that BGP speakers would. While the fixpoints maintain a distinction between each destination j, the effective routes at router i can be obtained by application of the function ϕ_i.

We will now illustrate the behaviour of Prebimonoid 6.17 in the context of aggregation, multi-homing and re-homing. In each scenarios, we will relate the sets of best routes obtained, $\mathbf{L}$, to the corresponding forwarding tables by means of the reduction functions ϕ_i. First we will show how the length of the `AS_SEQUENCE` influences the choice of best routes.

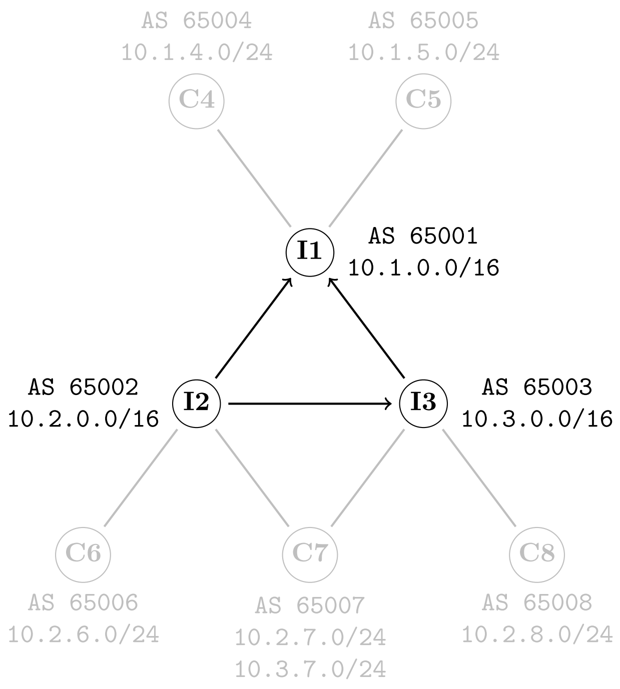


Figure 6.9: Infrastructure without internal structure of ISPs.

6.2.1 Role of the length of `AS_SEQUENCE`

The tie-breaking rules laid out in [47] state that among multiple routes with identical `NLRI`, the one with the shortest `AS_SEQUENCE` should be favoured and kept as the best route. According to the representation of route information we chose, the relation $\lesssim_{DP}$ captures this behaviour through the inclusion of the $\lesssim_{\text{length}}$ preference on lists based on their length. Let us consider the scenario depicted on Figure 6.9 where we limit the exchange of route information by not considering the bidirectional nature of eBGP peerings. An arrow from router i to router j means that i will receive route information from j after the latter performs aggregation and modifies the attributes of its best routes. This restriction is only introduced to limit the size of the resulting fixpoints and does not hinder in any way the ability to represent the complete scenario where route information flows in both directions of a peering. At the beginning of the execution, the routers only have their initial routes, $\mathcal{I}$, from which the computation starts.

Router	Allocated prefixes	AS Number	Aggregation rules
I1	`10.1.0.0/16`	65001	`10.1.0.0/16`
I2	`10.2.0.0/16`	65002	`10.2.0.0/16`
I3	`10.3.0.0/16`	65003	`10.3.0.0/16`

Figure 6.10: Configuration at each routers

In this scenario, the configuration used by the three routers is summarized on Table 6.12 where the configured AS number and allocated prefixes are given. Furthermore, we include sets of aggregation rules which reflect a full-aggregation scenario. The initial sets of effective routes are limited to the initial routes. From the configurations listed on Table 6.9, we can derive the functions $f_{i,j}$ which are applied by each peer to its set of effective routes before it is advertised to its peers. After one iteration, the resulting effective routes include all those received from the direct peers.

$$\begin{aligned}
\mathbf{eroutes}^1_{\mathbf{I1}} &= \mathcal{I} \\
\mathbf{eroutes}^1_{\mathbf{I2}} &= \{(\texttt{10.1.0.0/16}, [\texttt{65001}], \mathbf{I1}), (\texttt{10.3.0.0/16}, [\texttt{65003}], \mathbf{I3})\} \cup \mathcal{I} \\
\mathbf{eroutes}^1_{\mathbf{I3}} &= \{(\texttt{10.1.0.0/16}, [\texttt{65001}], \mathbf{I1})\} \cup \mathcal{I}
\end{aligned}$$

At this point, **I2** received two direct routes from **I1** and **I3** with their respective prefixes. The effective routes of **I2** are obtained by applying the Decision Process on the input set formed by those received routes. However, as a result of **I3** obtaining knowledge of a new route, it will begin to advertise this route for `10.1.0.0/16` to **I2** who will compare it against the direct one it already knows.

$$\begin{aligned}
f_{\mathbf{I2},\mathbf{I3}}\left(\mathbf{eroutes}^1_{\mathbf{I3}}\right) &= \{(\mathtt{10.1.0.0/16}, [\mathtt{65003}, \mathtt{65001}], \mathbf{I3}), (\mathtt{10.3.0.0/16}, [\mathtt{65003}], \mathbf{I3})\} \\
f_{\mathbf{I2},\mathbf{I1}}\left(\mathbf{eroutes}^1_{\mathbf{I1}}\right) &= \{(\mathtt{10.1.0.0/16}, [\mathtt{65001}], \mathbf{I1})\}
\end{aligned}$$

Upon performing the Decision Process on this new input set, **I2** will only keep the direct routes for the two respective prefixes. At this point, the effective routes have not changed and the computation has converged.

$$\begin{aligned}
\mathbf{eroutes}^2_{\mathbf{I1}} &= \mathcal{I} \\
\mathbf{eroutes}^2_{\mathbf{I2}} &= \{(\mathtt{10.1.0.0/16}, [\mathtt{65001}], \mathbf{I1}), (\mathtt{10.3.0.0/16}, [\mathtt{65003}], \mathbf{I3})\} \cup \mathcal{I} \\
\mathbf{eroutes}^2_{\mathbf{I3}} &= \{(\mathtt{10.1.0.0/16}, [\mathtt{65001}], \mathbf{I1})\} \cup \mathcal{I}
\end{aligned}$$

We can describe this scenario in terms of the Prebimonoid 6.17 by using the following adjacency matrix obtained according to Construction (6.16).

$$\mathbf{A} = \begin{array}{c} \\ \mathbf{I1} \\ \mathbf{I2} \\ \mathbf{I3} \end{array} \begin{array}{c} \begin{array}{ccc} \mathbf{I1} & \mathbf{I2} & \mathbf{I3} \end{array} \\ \left[\begin{array}{ccc} \varnothing & \varnothing & \varnothing \\ \{(\mathtt{10.1/16}, [\mathtt{65001}], \mathbf{I1})\} & \varnothing & \{(\mathtt{10.3/16}, [\mathtt{65003}], \mathbf{I3})\} \\ \{(\mathtt{10.1/16}, [\mathtt{65001}], \mathbf{I1})\} & \varnothing & \varnothing \end{array}\right] \end{array}$$

The initial effective routes are described by the identity for matrix multiplication. In this matrix, the diagonal contains the multiplicative unit of the prebimonoid, $\mathcal{I}$, while the rest of the matrix is set to the multiplicative annihilator, $\varnothing$.

$$\mathbf{L}^0 = \begin{array}{c} \\ \mathbf{I1} \\ \mathbf{I2} \\ \mathbf{I3} \end{array} \begin{array}{c} \begin{array}{ccc} \mathbf{I1} & \mathbf{I2} & \mathbf{I3} \end{array} \\ \left[\begin{array}{ccc} \mathcal{I} & \varnothing & \varnothing \\ \varnothing & \mathcal{I} & \varnothing \\ \varnothing & \varnothing & \mathcal{I} \end{array}\right] \end{array}$$

After one application of Definition (6.11), all the routers have the best routes from their directly connected peers. Specifically, **I2** has routes to reach both its direct neighbors and **I3** has a route to reach **I1**.

$$\mathbf{L}^1 = \begin{array}{c} \\ \mathbf{I1} \\ \mathbf{I2} \\ \mathbf{I3} \end{array} \begin{array}{c} \begin{array}{ccc} \mathbf{I1} & \mathbf{I2} & \mathbf{I3} \end{array} \\ \left[\begin{array}{ccc} \mathcal{I} & \varnothing & \varnothing \\ \{(\mathtt{10.1.0.0/16}, [\mathtt{65001}], \mathbf{I1})\} & \mathcal{I} & \{(\mathtt{10.3.0.0/16}, [\mathtt{65003}], \mathbf{I3})\} \\ \{(\mathtt{10.1.0.0/16}, [\mathtt{65001}], \mathbf{I1})\} & \varnothing & \mathcal{I} \end{array}\right] \end{array}$$

During the next application of Definition (6.11), the comparison performed by **I2** with respect to the routes received from **I1** and **I3** pertaining to the prefix `10.1.0.0/16` occurs without changing the corresponding entry

$$\begin{aligned}
\mathbf{L}^2_{\mathbf{I2},\mathbf{I1}} &= (\mathbf{A}_{\mathbf{I2},\mathbf{I1}} \otimes_E \mathbf{L}_{\mathbf{I1},\mathbf{I1}}) \oplus_E (\mathbf{A}_{\mathbf{I2},\mathbf{I3}} \otimes_E \mathbf{L}_{\mathbf{I3},\mathbf{I1}}) \\
&= \{(\mathtt{10.1.0.0/16}, [\mathtt{65001}], \mathbf{I1})\} \oplus_E \{(\mathtt{10.1.0.0/16}, [\mathtt{65003}, \mathtt{65001}], \mathbf{I3})\} \\
&= \{(\mathtt{10.1.0.0/16}, [\mathtt{65001}], \mathbf{I1})\}
\end{aligned}$$

The choice is consistent with the preference being given to routes having a shortest `AS_SEQUENCE` and the resulting matrix is identical to the previous one

$$\mathbf{L}^2 = \begin{array}{c} \\ \mathbf{I1} \\ \mathbf{I2} \\ \mathbf{I3} \end{array} \begin{array}{c} \begin{array}{ccc} \mathbf{I1} & \mathbf{I2} & \mathbf{I3} \end{array} \\ \left[\begin{array}{ccc} \mathcal{I} & \varnothing & \varnothing \\ (\mathtt{10.1.0.0/16}, [\mathtt{65001}], \mathbf{I1}) & \mathcal{I} & (\mathtt{10.3.0.0/16}, [\mathtt{65003}], \mathbf{I3}) \\ (\mathtt{10.1.0.0/16}, [\mathtt{65001}], \mathbf{I1}) & \varnothing & \mathcal{I} \end{array}\right] \end{array}$$

In this situation, both processes reach convergence after one iteration. Furthermore, at each iteration, the current matrix $\mathbf{L}$ satisfies the Relation (6.9), $\phi_i\left(\mathbf{L}^k\right) = \mathbf{eroutes}^k_i$ and we can obtain the effective routes by applying for each i the reduction function ϕ_i to the current matrix.

6.2.2 Simple route aggregation

The basic functionality behind route aggregation is to conceal details about all the routes it knows down to a minimal amount. This elementary mecanism is illustrated on Figure 6.11 where ISP 1 acts as a connectivity provider for Customer 4 and 5. In this scenario, ISP 1 sub-allocated portions of its own network prefix to those Customers who can assign addresses taken from their allocated range to render some of their routers and hosts reachable from the rest of the infrastructure. In order to limit the amount of route entries in the forwarding tables, ISP 1 aggregates the routes for the prefixes of its two customers, `10.1.4.0/24` and `10.1.5.0/24`, before advertising a unique route to its own prefix `10.1.0.0/16`. The adjacency matrix that encodes this configuration is given below where the entry that governs the way **I1** export its routes to **I2** will have the effect of advertising this aggregate route in place of any route falling under its prefix

$$\mathbf{A_{I2,I1}} = \{(\texttt{10.1.0.0/16}, [\texttt{65001}], \mathbf{I1})\}$$

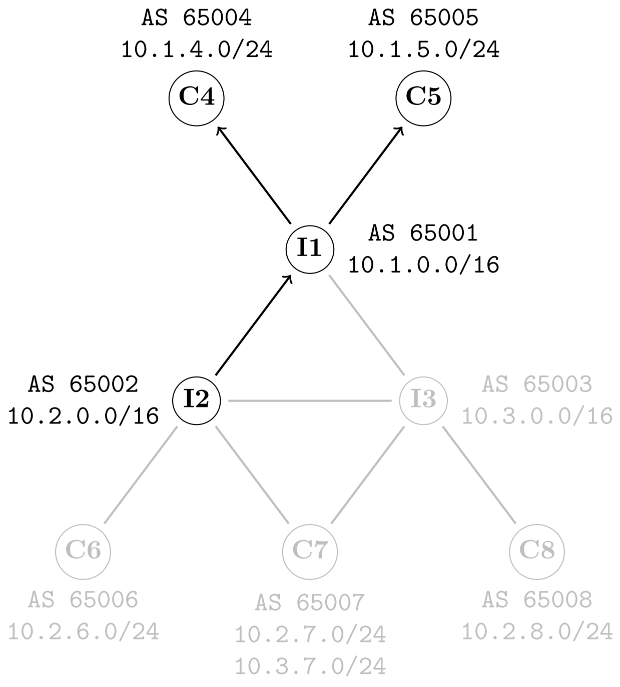


Figure 6.11: Infrastructure with emphasis on the aggregation of routes towards Customers 4 and 5.

Router	Allocated prefixes	AS Number	Aggregation rules
I1	`10.1.0.0/16`	65001	`10.1.0.0/16`
I2	`10.2.0.0/16`	65002	`10.2.0.0/16`
C4	`10.1.4.0/24`	65004	`10.1.4.0/24`
C5	`10.1.5.0/24`	65005	`10.1.5.0/24`

Figure 6.12: Configuration at each routers of Figure 6.11

The effective routes at the beginning of the execution are limited to the sets of initial routes. After one application of Definition (6.13), we obtain

$$\begin{aligned}
\mathbf{eroutes}^1_{\mathbf{I1}} &= \{(\texttt{10.1.4/24}, [\texttt{65004}], \mathbf{C4}), (\texttt{10.1.5/24}, [\texttt{65005}], \mathbf{C5})\} \cup \mathcal{I} \\
\mathbf{eroutes}^1_{\mathbf{I2}} &= \{(\texttt{10.1/16}, [\texttt{65001}], \mathbf{I1})\} \cup \mathcal{I} \\
\mathbf{eroutes}^1_{\mathbf{C4}} &= \mathcal{I} \\
\mathbf{eroutes}^1_{\mathbf{C5}} &= \mathcal{I}
\end{aligned}$$

At this point, the computation has stabilized since the aggregate route which **I1** will advertise in place of the routes for its two prefixes will not change any sets of effective routes. In particular, **I1** will keep the effective routes towards its customers which enables proper forwarding to them.

$$\begin{aligned}
\mathbf{eroutes}^2_{\mathbf{I1}} &= \{(\texttt{10.1.4/24}, [\texttt{65004}], \mathbf{C4}), (\texttt{10.1.5/24}, [\texttt{65005}], \mathbf{C5})\} \cup \mathcal{I} \\
\mathbf{eroutes}^2_{\mathbf{I2}} &= \{(\texttt{10.1/16}, [\texttt{65001}], \mathbf{I1})\} \cup \mathcal{I} \\
\mathbf{eroutes}^2_{\mathbf{C4}} &= \mathcal{I} \\
\mathbf{eroutes}^2_{\mathbf{C5}} &= \mathcal{I}
\end{aligned}$$

The following adjacency matrix describes how all the routers are configured. In particular, **C4** and **C5** aggregate on their respective prefixes to avoid exporting internal details concerning their infrastructure. In the same way, the routers **I1** and **I2** perform route aggregation based on a unique aggregation rule that covers their allocated prefix.

$$\mathbf{A} = \begin{array}{c|cccc} & \mathbf{I1} & \mathbf{I2} & \mathbf{C4} & \mathbf{C5} \\ \hline \mathbf{I1} & \varnothing & \varnothing & (\texttt{10.1.4/24}, [\texttt{65004}], \mathbf{C4}) & (\texttt{10.1.5/24}, [\texttt{65005}], \mathbf{C5}) \\ \mathbf{I2} & (\texttt{10.1/16}, [\texttt{65001}], \mathbf{I1}) & \varnothing & \varnothing & \varnothing \\ \mathbf{C4} & \varnothing & \varnothing & \varnothing & \varnothing \\ \mathbf{C5} & \varnothing & \varnothing & \varnothing & \varnothing \end{array}$$

After one application of Definition (6.11), we obtain the best routes involving the direct connections.

$$\mathbf{L}^1 = \begin{array}{c|cccc} & \mathbf{I1} & \mathbf{I2} & \mathbf{C4} & \mathbf{C5} \\ \hline \mathbf{I1} & \mathcal{I} & \varnothing & (\texttt{10.1.4/24}, [\texttt{65004}], \mathbf{C4}) & (\texttt{10.1.5/24}, [\texttt{65005}], \mathbf{C5}) \\ \mathbf{I2} & (\texttt{10.1/16}, [\texttt{65001}], \mathbf{I1}) & \mathcal{I} & \varnothing & \varnothing \\ \mathbf{C4} & \varnothing & \varnothing & \mathcal{I} & \varnothing \\ \mathbf{C5} & \varnothing & \varnothing & \varnothing & \mathcal{I} \end{array}$$

The fixpoint stabilizes after a second application of Definition (6.11) once the best routes for the customers are found.

$$\mathbf{L}^2 = \begin{array}{c|cccc} & \mathbf{I1} & \mathbf{I2} & \mathbf{C4} & \mathbf{C5} \\ \hline \mathbf{I1} & \mathcal{I} & \varnothing & (\texttt{10.1.4/24}, [\texttt{65004}], \mathbf{C4}) & (\texttt{10.1.5/24}, [\texttt{65005}], \mathbf{C5}) \\ \mathbf{I2} & (\texttt{10.1/16}, [\texttt{65001}], \mathbf{I1}) & \mathcal{I} & (\texttt{10.1.0.0/16}, [\texttt{65001}], \mathbf{I1}) & (\texttt{10.1.0.0/16}, [\texttt{65001}], \mathbf{I1}) \\ \mathbf{C4} & \varnothing & \varnothing & \mathcal{I} & \varnothing \\ \mathbf{C5} & \varnothing & \varnothing & \varnothing & \mathcal{I} \end{array}$$

While **I1** keeps to itself the effective routes for `10.1.4.0/24` and `10.1.5.0/24`, it advertises the unique route for `10.1.0.0/16` to **I2**. We can relate the contents of the fixpoint with that of the forwarding tables by applying ϕ_i to $\mathbf{L}^2$.

$$\begin{aligned}
\phi_{\mathbf{I1}}\left(\mathbf{L}^2\right) &= \{(\texttt{10.1.4.0/24}, [\texttt{65004}], \mathbf{C4}), (\texttt{10.1.5.0/24}, [\texttt{65005}], \mathbf{C5}), (\texttt{10.2.0.0/16}, [\texttt{65002}], \mathbf{I2})\} \cup \mathcal{I} \\
\phi_{\mathbf{I2}}\left(\mathbf{L}^2\right) &= \{(\texttt{10.1.0.0/16}, [\texttt{65001}], \mathbf{I1})\} \cup \mathcal{I}
\end{aligned}$$

Note that in this case, the computation of the effective routes according to Equation (6.13) converges faster than the one based on the recurrence equation. Further work into the details of both definitions would provide insight into this slight difference.

6.2.3 Multi-homing of customers

A particular scenario which can occur in the real-world is known as multi-homing where a customer is reachable and obtains connectivity through more than one provider. Consider the case of Customer 7, a domain multi-homed through its two providers; ISP 2 and ISP 3. In this situation, the network prefixes are assumed to be Provider-Aggregatable in the respective ranges of the providers. Customer 7 is reachable through two distinct routes from ISP 1, either by means of the prefix `10.2.7.0/24` through ISP 2 or by using the prefix `10.3.7.0/24` through ISP 3.

After one iteration, the computation of effective routes will produce the following sets at each router

$$\begin{aligned}
\mathbf{eroutes}^1_{\mathbf{I1}} &= \{(\texttt{10.2.0.0/16}, [\texttt{65002}], \mathbf{I2}), (\texttt{10.3.0.0/16}, [\texttt{65003}], \mathbf{I3})\} \cup \mathcal{I} \\
\mathbf{eroutes}^1_{\mathbf{I2}} &= \{(\texttt{10.2.7.0/24}, [\texttt{65007}], \mathbf{C7})\} \cup \mathcal{I} \\
\mathbf{eroutes}^1_{\mathbf{I3}} &= \{(\texttt{10.3.7.0/24}, [\texttt{65007}], \mathbf{C7})\} \cup \mathcal{I} \\
\mathbf{eroutes}^1_{\mathbf{C7}} &= \mathcal{I}
\end{aligned}$$

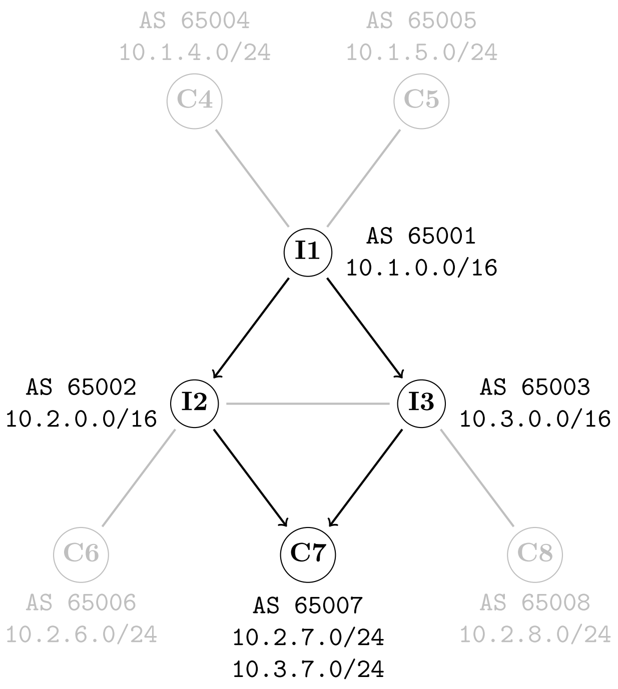


Figure 6.13: Infrastructure with emphasis on the multi-homing of Customer 7 through ISP 2 and ISP 3.

An additional application will not result in any changes, thus all the sets of effective routes have already stabilized on their final contents.

$$\begin{aligned}
\textbf{eroutes}^2_{\textbf{I1}} &= \{(\texttt{10.2.0.0/16}, [\texttt{65002}], \textbf{I2}), (\texttt{10.3.0.0/16}, [\texttt{65003}], \textbf{I3})\} \cup \mathcal{I} \\
\textbf{eroutes}^2_{\textbf{I2}} &= \{(\texttt{10.2.7.0/24}, [\texttt{65007}], \textbf{C7})\} \cup \mathcal{I} \\
\textbf{eroutes}^2_{\textbf{I3}} &= \{(\texttt{10.3.7.0/24}, [\texttt{65007}], \textbf{C7})\} \cup \mathcal{I} \\
\textbf{eroutes}^2_{\textbf{C7}} &= \mathcal{I}
\end{aligned}$$

The adjacency matrix for the same scenario is given below based on the peerings depicted on Figure 6.13.

$$\mathbf{A} = \begin{array}{c} \\ \textbf{I1} \\ \textbf{I2} \\ \textbf{I3} \\ \textbf{C7} \end{array} \begin{array}{cccc} \textbf{I1} & \textbf{I2} & \textbf{I3} & \textbf{C7} \\ \end{array}
\left[\begin{array}{cccc}
\varnothing & (\texttt{10.2/16}, [\texttt{65002}], \textbf{I2}) & (\texttt{10.3/16}, [\texttt{65003}], \textbf{I3}) & \varnothing \\
\varnothing & \varnothing & (\texttt{10.3/16}, [\texttt{65003}], \textbf{I3}) & (\texttt{10.2.7/24}, [\texttt{65007}], \textbf{C7}) \\
\varnothing & \varnothing & \varnothing & (\texttt{10.3.7/24}, [\texttt{65007}], \textbf{C7}) \\
\varnothing & \varnothing & \varnothing & \varnothing
\end{array}\right]$$

From the perspective of ISP 1, there are two ways to reach Customer 7; one route is in the `10.2.0.0/16` prefix and the other one in `10.3.0.0/16`. ISP 1 should receive one advertisement from ISP 2 covering the route `10.2.7.0/24` and another one from ISP 3 covering the route `10.3.7.0/24`. The fixpoint to the left equation is given below

$$\mathbf{L}^2 = \begin{array}{c|ccc}
 & \textbf{I1} & \textbf{I2} & \textbf{I3} \\
\hline
\textbf{I1} & \mathcal{I} & (\texttt{10.2.0.0/16}, [\texttt{65002}], \textbf{I2}) & (\texttt{10.3.0.0/16}, [\texttt{65003}], \textbf{I3}) \\
\textbf{I2} & \varnothing & \mathcal{I} & \varnothing \\
\textbf{I3} & \varnothing & \varnothing & \mathcal{I} \\
\textbf{C7} & \varnothing & \varnothing & \varnothing
\end{array}$$

$$\mathbf{L}^2 = \begin{array}{c|c}
 & \textbf{C7} \\
\hline
\textbf{I1} & \{(\texttt{10.2.0.0/16}, [\texttt{65002}], \textbf{I2}), (\texttt{10.3.0.0/16}, [\texttt{65003}], \textbf{I3})\} \\
\textbf{I2} & (\texttt{10.2.7/24}, [\texttt{65007}], \textbf{C7}) \\
\textbf{I3} & (\texttt{10.3.7/24}, [\texttt{65007}], \textbf{C7}) \\
\textbf{C7} & \mathcal{I}
\end{array}$$

In this format, the entries of the matrix represent the prefixes through which a domain is known from the perspective of another. ISP 1 has the knowledge to reach Customer 7 by using one of two possible prefixes; if a packet arrives bound for a destination address in the range `10.2.0.0/16` (resp. `10.3.0.0/16`), it will forward it to the implied path that goes through ISP 2 (resp. ISP 3) and if the destination address further falls in the more specific prefixes allocated to Customer 7, the respective next-hops will forward the packet to Customer 7. The routes obtained from router **I2** and **I3** after aggregation and extension are

$$\begin{aligned}
\mathbf{A_{I1,I2}} \otimes_E \mathbf{L_{I2,C7}} &= (\texttt{10.2.0.0/16}, [\texttt{65002}], \mathbf{I2}) \otimes_E (\texttt{10.2.7/24}, [\texttt{65007}], \mathbf{C7}) \\
&= (\texttt{10.2.0.0/16}, [\texttt{65002}], \mathbf{I2}) \qquad (\mathbf{I1} \to \mathbf{I2} \to \mathbf{C7}) \\
\mathbf{A_{I1,I3}} \otimes_E \mathbf{L_{I3,C7}} &= (\texttt{10.2.0.0/16}, [\texttt{65003}], \mathbf{I3}) \otimes_E (\texttt{10.3.7/24}, [\texttt{65007}], \mathbf{C7}) \\
&= (\texttt{10.3.0.0/16}, [\texttt{65003}], \mathbf{I3}) \qquad (\mathbf{I1} \to \mathbf{I3} \to \mathbf{C7})
\end{aligned}$$

The entry $\mathbf{L_{I1,C7}}$ contains the best routes from Customer 7 after they are aggregated under the respective prefixes of ISP 2 and ISP 3.

$$\begin{aligned}
\mathbf{L_{I1,C7}} &= (\mathbf{A_{I1,I2}} \otimes_E \mathbf{L_{I2,C7}}) \oplus_E (\mathbf{A_{I1,I3}} \otimes_E \mathbf{L_{I3,C7}}) \\
&= \{(\texttt{10.2.0.0/16}, [\texttt{65002}], \mathbf{I2})\} \oplus_E \{(\texttt{10.3.0.0/16}, [\texttt{65003}], \mathbf{I3})\} \\
&= \{(\texttt{10.2.0.0/16}, [\texttt{65002}], \mathbf{I2}), (\texttt{10.3.0.0/16}, [\texttt{65003}], \mathbf{I3})\}
\end{aligned}$$

The rows of the fixpoint produced are consistent with the sets of effective routes under the application of the reduction functions ϕ_i.

6.2.4 Re-homing of customers

The last scenario that can occur in real-world infrastructures is known as re-homing whereby a customer can change its connectivity from one provider to another while keeping its allocated prefix. Consider the case of Customer 8, which has re-homed by changing its connectivity from ISP 2 to ISP 3 while keeping the Provider-Aggregatable prefix `10.2.8.0/24`. The conditions and conventions under which this situation is handled in real-world scenarios are given in [1]. In this case, the rule that must be used by the ISP who provides connectivity to a re-homed Customer is to advertise the route to the specific prefix to its neighbors. ISP 1 will have two separate entries in its forwarding table; one for `10.2.0.0/16` with ISP 2 as its next-hop and another one for `10.2.8.0/24` with ISP 3 as its next-hop. Any packet bound for an address in `10.2.8.0/24` will be forwarded to ISP 3 to which the relevant destination is connected while any address falling in the more general `/16` prefix will be handed to ISP 2 by virtue of the LMP rule.

After one iteration, the computation of the effective routes will produce the following sets at each router

$$\begin{aligned}
\mathbf{eroutes^1_{I1}} &= \{(\texttt{10.2.0.0/16}, [\texttt{65002}], \mathbf{I2}), (\texttt{10.3.0.0/16}, [\texttt{65003}], \mathbf{I3})\} \cup \mathcal{I} \\
\mathbf{eroutes^1_{I2}} &= \{(\texttt{10.3.0.0/16}, [\texttt{65003}], \mathbf{I3})\} \cup \mathcal{I} \\
\mathbf{eroutes^1_{I3}} &= \{(\texttt{10.2.8.0/24}, [\texttt{65008}], \mathbf{C8})\} \cup \mathcal{I} \\
\mathbf{eroutes^1_{C8}} &= \mathcal{I}
\end{aligned}$$

After the second iteration, all the routes will be available known at each router. In particular, **I1** will have two routes for prefixes that are related by the subprefix relation.

$$\begin{aligned}
\mathbf{eroutes^2_{I1}} &= \{(\texttt{10.2.0.0/16}, [\texttt{65002}], \mathbf{I2}), (\texttt{10.3.0.0/16}, [\texttt{65003}], \mathbf{I3}), (\texttt{10.2.8.0/24}, [\texttt{65003}, \texttt{65008}], \mathbf{I3})\} \cup \mathcal{I} \\
\mathbf{eroutes^2_{I2}} &= \{(\texttt{10.3.0.0/16}, [\texttt{65003}], \mathbf{I3}), (\texttt{10.2.8.0/24}, [\texttt{65003}, \texttt{65008}], \mathbf{I3})\} \cup \mathcal{I} \\
\mathbf{eroutes^2_{I3}} &= \{(\texttt{10.2.8.0/24}, [\texttt{65008}], \mathbf{C8})\} \cup \mathcal{I} \\
\mathbf{eroutes^2_{C8}} &= \mathcal{I}
\end{aligned}$$

The problem can be encoded alternatively into the following adjacency matrix.

$$\mathbf{A} = \begin{array}{c|cccc} & \mathbf{I1} & \mathbf{I2} & \mathbf{I3} & \mathbf{C8} \\ \hline \mathbf{I1} & \varnothing & (\texttt{10.2/16}, [\texttt{65002}], \mathbf{I2}) & (\texttt{10.3/16}, [\texttt{65003}], \mathbf{I3}) & \varnothing \\ \mathbf{I2} & \varnothing & \varnothing & (\texttt{10.3/16}, [\texttt{65003}], \mathbf{I3}) & \varnothing \\ \mathbf{I3} & \varnothing & \varnothing & \varnothing & (\texttt{10.2.8/24}, [\texttt{65008}], \mathbf{C8}) \\ \mathbf{C8} & \varnothing & \varnothing & \varnothing & \varnothing \end{array}$$

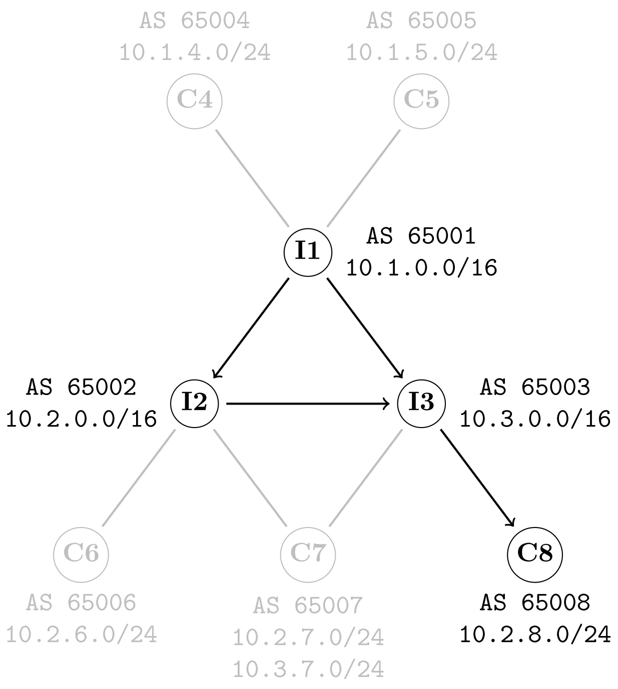


Figure 6.14: Infrastructure with emphasis on the re-homing of Customer 8 from ISP 2 to ISP 3.

ISP 3 will provide directly the unaggregated route towards Customer 8 to ISP 1 and ISP 2. This route will be aggregated under the prefix of ISP 2 before being re-advertised to ISP 1. However the more specific one will be kept as the best route towards Customer 8 due to it having the most specific prefix. This situation is reflected in the fixpoint produced. Once again, the function ϕ_i relates consistently the rows of the fixpoint to the various sets of effective routes obtained by the routers involved.

$$\begin{aligned}
\mathbf{A_{I1,I2}} \otimes_E \mathbf{L_{I2,C8}} &= (\mathsf{10.2.0.0/16}, [\mathsf{65002}], \mathbf{I2}) \otimes_E (\mathsf{10.2.8.0/24}, [\mathsf{65003}, \mathsf{65008}], \mathbf{I3}) \\
&= (\mathsf{10.2.0.0/16}, [\mathsf{65002}], \mathbf{I2}) \qquad (\mathbf{I1} \to \mathbf{I2} \to \mathbf{I3} \to \mathbf{C8}) \\
\mathbf{A_{I1,I3}} \otimes_E \mathbf{L_{I3,C8}} &= (\mathsf{10.3.0.0/16}, [\mathsf{65003}], \mathbf{I3}) \otimes_E (\mathsf{10.2.8.0/24}, [\mathsf{65008}], \mathbf{C8}) \\
&= (\mathsf{10.2.8.0/24}, [\mathsf{65003}, \mathsf{65008}], \mathbf{I3}) \qquad (\mathbf{I1} \to \mathbf{I2} \to \mathbf{C8})
\end{aligned}$$

The fixpoint is given below and it encodes the best routes for each destination. Each row is consistent under the application of the reduction function ϕ_i with the sets of effective routes given above.

$$\mathbf{L}^3 = \begin{array}{c} \\ \mathbf{I1} \\ \mathbf{I2} \\ \mathbf{I3} \\ \mathbf{C8} \end{array} \begin{array}{cccc} \mathbf{I1} & \mathbf{I2} & \mathbf{I3} & \mathbf{C8} \\ \mathcal{I} & (\mathsf{10.2/16}, [\mathsf{65002}], \mathbf{I2}) & (\mathsf{10.3/16}, [\mathsf{65003}], \mathbf{I3}) & (\mathsf{10.2.8/24}, [\mathsf{65003}, \mathsf{65008}], \mathbf{I3}) \\ \varnothing & \mathcal{I} & (\mathsf{10.3/16}, [\mathsf{65003}], \mathbf{I3}) & (\mathsf{10.2.8/24}, [\mathsf{65003}, \mathsf{65008}], \mathbf{I3}) \\ \varnothing & \varnothing & \mathcal{I} & (\mathsf{10.2.8/24}, [\mathsf{65008}], \mathbf{C8}) \\ \varnothing & \varnothing & \varnothing & \mathcal{I} \end{array}$$

6.3 A simplified model of Route Aggregation in IGP.

We have shown in the previous section how route aggregation can be modelled with a prebimonoid at the eBGP level without consideration as to the internal structure of the domains involved. However, route aggregation can also be used within the internal network of a domain within distance-vectoring protocols. The routes which are discovered by border routers can be redistributed into the network of the domain by using the advertisements from the IGP to carry the routes to external destinations. This has the effect of introducing entries pertaining to external destinations into the routing tables of internal routers. We will be focusing on those external routes as they are injected into the routing tables.

In this section, we will use the same approach as for eBGP to model the behaviour of route aggregation within a simplified IGP which uses the distance as its metric and includes the next-hops. While there are several options to configure aggregation within an IGP, the one we will focus on is the interface-based approach where aggregation rules are configured on individual interfaces by each router. Upon advertising routes, the aggregation rules of an interface are applied to the routes before they are sent. One of the unexpected behaviours of route aggregation within RIPv2 presented [36] is that the configuration on one interface can interfere with the advertisements of other interfaces.

We will focus on the network of ISP 2 on Figure 6.3 where some routers are configured to use aggregation on the same prefix. The scenario under which the forwarding loops arise, along with the sequence of advertisements leading to it, is based on one of those appearing in [36]. Within an IGP, we describe routes as the association of a `NLRI` together with an interior cost and a next-hop.

$$\mathcal{R}outes_{IGP} = \mathcal{P} \times \mathbb{N}^{\infty} \times \mathcal{V}$$

We will be using the subprefix relation $\sqsubseteq$ as well as the total order which was used on advertising peers $\leqslant_{\mathcal{V}}$. For the distance metric, we introduce $\leqslant_{\infty}$

$$a \leqslant_{\infty} b \equiv \begin{cases} \texttt{false} & a = \infty \ \wedge \ b \in Nats \\ \texttt{true} & b = \infty \\ a \leqslant b & a, b \in \mathbb{N} \end{cases}$$

which behaves in the same way as the less-than order for natural numbers with the exception that ∞ is greater than any other element in $\mathbb{N}$. The preference on routes is reminiscent of the one in eBGP with the exception for the use of the total ordering on the distance. The $\lesssim_{BR}$ relation favours routes having a lower interior cost and further breaks ties on the next-hop.

$$\lesssim_{BR} \ \equiv \ = \ \vec{\times} \ \leqslant_{\infty} \ \vec{\times} \ \leqslant_{\mathcal{V}}$$

The ground set we will be using is obtained by reducing all the subsets of $\mathcal{R}outes_{IGP}$ to their minimal elements, including an element $\mathcal{I}$ for which we will make additional assumptions. In particular, we assume that $\mathcal{I} = \min_{\lesssim_{BR}}(\mathcal{I})$.

$$\mathcal{S}_I = \left\{ \min_{\lesssim_{BR}}(R) \ \mid R \subseteq \mathcal{R}outes_{IGP} \right\} \cup \{\mathcal{I}\},$$

Whenever a router receives an advertisement for some destination from a neighbor, it performs a lookup in its routing table. When there is no route known for the destination prefix, the route is introduced in the routing table. In the case there is a route already known, the costs are compared and the route with the least cost is kept in the routing table. A router can further be configured to perform route aggregation on certain links. If a given aggregate route is configured on a specific interface and multiple matching routes are known with different interior costs, the router will only advertise the aggregate route and pick as its interior cost the least one among all child routes. Formally, we can express this behaviour as

$$\mathbf{eroutes}_i = \min_{\lesssim_{BR}} \left(\bigcup_q f_{i,q}\left(\mathbf{eroutes}_q\right) \right)$$

The functions $f_{i,j}$ implement the transformation of routes from the set E according to whether j is a neighbor of i, the aggregation rules $R_{i,j}$ configured on the link from j to i.

$$f_{i,j}(E) = \begin{cases} \varnothing & \text{i is not a neighbor of j} \\ \min_{\lesssim_{BR}} \left(\mathbf{aggregate} - \mathbf{routes}_{i,j}(E) \cup \mathbf{remaining} - \mathbf{routes}_{i,j}(E) \right) & \text{i is a neighbor of j} \end{cases} \tag{6.19}$$

Given a set of aggregation rules $R_{i,j}$ configured at router j with respect to i, the weight of the link from i to j and a set of routes E, the transformation that the routes undergo produces a set where

- All the routes (n_e, c_e, v_e) from E that fall under the prefix of an aggregation rule, $n_e \sqsubset n_a \in R_{i,j}$, are replaced by a unique route with prefix n_a, with its cost equal to the c_e to which the weight of the link traversed is added and j as the next-hop

  $$\mathbf{aggregate} - \mathbf{routes}_{i,j}(E) = \{(n_a, \omega(i,j) + c_b, j) \mid n_a \in R_{i,j} \ \wedge \ (n_b, c_b, h_b) \in B \ \wedge \ n_b \ \sqsubset \ n_a\}$$

- All the routes (n_e, l_e, v_e) from E for which there is no aggregation rule with a covering prefix, $\nexists\, n_a \ \in \ R_{i,j} \ : \ n_e \sqsubset n_a$, have their cost increased by the weight of the link and their next-hop set to j.

  $$\mathbf{remaining} - \mathbf{routes}_{i,j}(E) = \{(n_b, \omega(i,j) + c_b, j) \mid (n_b, c_b, h_b) \in B \ : \ \forall\, n_a \ \in \ R_{i,j} \ : \ \neg(n_b \ \sqsubset \ n_a)\}$$

The result conceals all the child routes behind a general prefix but increases the interior cost of each resulting route with that of the link traversed and only the best resulting routes are advertised. These functions can be expressed in terms of the multiplicative law. Given a set of routes B, the left term of the union represents the aggregation rules of A resulting from the presence of one or more child routes in B matching those aggregation rules.

$$\begin{aligned}\textbf{aggregate}\,(A,B) &= \{(n_a, c_a + c_b, h_a) \mid (n_a, c_a, h_a) \in A \;\wedge\; (n_b, c_b, h_b) \in B \;\wedge\; n_b \sqsubseteq n_a\}\\ \textbf{remaining}\,(A,B) &= \{(n_b, c_a + c_b, h_a) \mid (n_a, c_a, h_a) \in A \;\wedge\; (n_b, c_b, h_b) \in B \;:\; \forall\,(n_a, c_a, h_a) \in A \;:\; \neg(n_b \sqsubseteq n_a)\}\end{aligned}$$

The first set contains the aggregate routes of A for which one or more matching routes are found in B while the second is the set of routes in B for which no corresponding aggregate rule exists in A with their advertising router updated. Whether a route matches an aggregation rule or not, its associated interior cost is increased by the distance associated with the link used. Given a set of aggregation rules A and a set of routes B, we have

$$A \otimes_I B = \begin{cases} A & B = \mathcal{I} \\ B & A = \mathcal{I} \\ \min_{\lesssim_{BR}} (\textbf{aggregate}\,(A,B) \cup \textbf{remaining}\,(A,B)) & \text{otherwise} \end{cases} \tag{6.20}$$

The multiplicative law defined incorporates the same behaviour as the functions $f_{i,j}$. The application of the minimum operation in the last case has an interesting consequence. When aggregation is performed in RIPv2, a choice must be made for the interior cost associated to the aggregate route. In this case, the rule is to pick the least interior cost across all the child routes and use it as the value for the aggregate route. If two routes with interior costs 1 and 2 get aggregated under a common prefix, the aggregate route will have an interior cost of 1. As it is advertised, the interior cost will be updated based on the link used.

$$\begin{aligned} A &= \{(\texttt{10.2.0.0/16}, 1, \mathbf{R0})\}\\ B &= \{(\texttt{10.2.6.0/24}, 1, \mathbf{B3}), (\texttt{10.2.7.0/24}, 2, \mathbf{B2})\}\\ A \otimes_I B &= \min_{\lesssim_{BR}} (\{(\texttt{10.2.0.0/16}, 2, \mathbf{R1}), (\texttt{10.2.0.0/16}, 3, \mathbf{R1})\})\\ &= \{(\texttt{10.2.0.0/16}, 2, \mathbf{R1})\}\end{aligned}$$

We provide a construction to express a given configuration of the internal routers in terms of sets of routes. If i and j are two routers with the interior cost of the link given by $\omega(i,j)$ and the aggregation rules $R_{i,j}$ are configured on the interface connected to the link that goes to j, we can construct the corresponding multiplicative operand A according to

$$A = \begin{cases} \varnothing & \omega(i,j) = \infty \\ \{(\bar{l}, \omega(i,j), j)\} & \omega(i,j) \neq \infty \;\wedge\; R_{i,j} = \varnothing \\ \{(n_r, \omega(i,j), j) \mid n_r \in R_{i,j}\} & \text{otherwise} \end{cases} \tag{6.21}$$

Suppose that two routes are known at router **R1**, respectively for prefixes `10.2.6.0/24` and `10.2.7.0/24`, with different costs and **R1** is configured to aggregate under the prefix `10.2.0.0/16` for the interface leading to **R0**. The resulting aggregate routes that **R0** receives from **R1** would be given by

$$\{(\texttt{10.2.0.0/16}, 2, \mathbf{R1}), (\texttt{10.2.0.0/16}, 3, \mathbf{R1})\} = \{(\texttt{10.2.0.0/16}, 1, \mathbf{R1})\} \otimes_I \{(\texttt{10.2.6.0/24}, 1, \mathbf{B3}), (\texttt{10.2.7.0/24}, 2, \mathbf{B2})\}$$

We define $\oplus_I$ to keep only routes that are minimal with respect to the $\lesssim_{IGP}$ relation. In effect, given two sets of routes A and B, only the ones with the most specific prefixes will be kept. Furthermore, when multiple routes exist for one `NLRI`, the one with the least interior cost is kept.

$$A \oplus_I B = \begin{cases} \min_{\lesssim_{BR}} (A \cup B) & A \neq \mathcal{I} \neq B \\ A \cup B & \text{otherwise} \end{cases} \tag{6.22}$$

The structure we use to model this simplified IGP has a ground set defined . The various proofs of the properties are given as the two subsequent lemmas.

$$\text{IGP} = \left(\left\{ \min_{\lesssim_{BR}} (R) \mid R \subseteq \mathcal{R}outes_{IGP} \right\} \cup \{\mathcal{I}\}, \oplus_I, \otimes_I \right) \tag{6.23}$$

Lemma 6.3.1. *The additive law $\oplus_I$ is* ASSOCIATIVE, COMMUTATIVE, IDEMPOTENT *and admits $\varnothing$ as an* IDENTITY.

Proof. We recall that given that $\lesssim_{BR}$ is a preorder, for any two sets A and B we have

$$\min_{\lesssim_{BR}}(A \cup B) = \min_{\lesssim_{BR}}\left(\min_{\lesssim_{BR}}(A) \cup B\right) \tag{6.24}$$

ASSOCIATIVITY. The proof directly follows from Statement 6.24.

$$\begin{aligned}
(A \oplus_I B) \oplus_I C &= \min_{\lesssim_{BR}}\left(\min_{\lesssim_{BR}}(A \cup B) \cup C\right) \\
&= \min_{\lesssim_{BR}}((A \cup B) \cup C) && \text{(By 6.24)} \\
&= \min_{\lesssim_{BR}}(A \cup (B \cup C)) && (\text{ASSOCIATIVITY of } \cup) \\
&= \min_{\lesssim_{BR}}\left(A \cup \min_{\lesssim_{BR}}(B \cup C)\right) && \text{(By 6.24)} \\
&= A \oplus_I (B \oplus_I C)
\end{aligned}$$

- COMMUTATIVITY of $\oplus_I$ follows from the COMMUTATIVITY of $\cup$

$$A \oplus_I B = \min_{\lesssim_{BR}}(A \cup B) = \min_{\lesssim_{BR}}(B \cup A) = B \oplus_I A$$

- IDEMPOTENCY holds given the IDEMPOTENCY of $\cup$ and the fact that $A = \min_{\lesssim_{BR}}(A)$ for all elements A of the ground set

$$\begin{aligned}
A \oplus_I A &= \min_{\lesssim_{BR}}(A \cup A) \\
&= \min_{\lesssim_{BR}}(A) \\
&= A
\end{aligned}$$

- The empty set is an IDENTITY for $\oplus_I$ given that it is an identity for the underlying $\cup$ operation.

$$A \oplus_I \varnothing = \min_{\lesssim_{BR}}(A \cup \varnothing) = \min_{\lesssim_{BR}}(A) = \min_{\lesssim_{BR}}(\varnothing \cup A) = \varnothing \oplus_I A$$

□

Lemma 6.3.2. *The multiplicative law $\otimes_I$ admits the empty set as an* ANNIHILATOR, *$\mathcal{I}$ as an* IDENTITY.

Proof.
- The proof that $\varnothing$ is an ANNIHILATOR for $\otimes_I$ works by assuming either operand to be equal to it followed by simplification. In either case, both terms of the union can be shown to reduce to the empty set which leaves us with the empty set.
- The proof that $\mathcal{I}$ is an IDENTITY follows directly from Definition (6.20).

□

We conclude that the structure IGP is an IDEMPOTENT prebimonoid. By using Construction (6.21), we can construct an adjacency matrix based on a given input network. Let $\mathcal{G} = (\mathcal{V}, \mathcal{A})$ be a directed graph with a weight function $\omega : \mathcal{A} \mapsto \mathbb{N}^{\infty}$ and a function providing the set of aggregation rules configured on each link from i to j, $R_{i,j}$. The adjacency matrix $\mathbf{A}$ can be produced by applying Construction (6.21 for each entry i, j to be produced.

Note that the adjacency matrix obtained in this way does not correctly represent the reality of the aggregation behaviour observed in [36]. In particular, the configuration of aggregation rules on some interfaces impact the outgoing advertisements on other interfaces of the same router depending on the activated rules. Due to the lack of thorough understanding of this behaviour, we are unable to define the exact method by which the adjacency matrix can be constructed in every case.

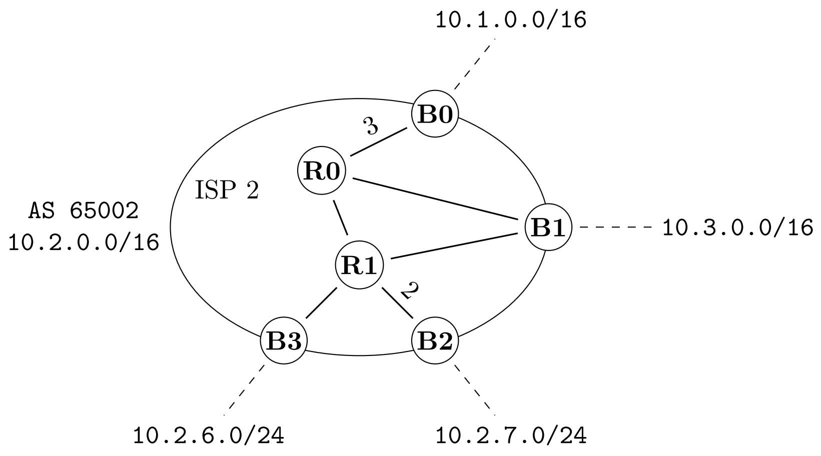


Figure 6.15: Detailed infrastructure of ISP 2 giving rise to reachability issues. All the unlabelled links are assumed to have a cost of 1 associated with them.

As was the case in the model of eBGP, the matrices produced according to this construct do not satisfy LEFT-INFLATIONARITY.

$$\begin{aligned} A &= \{(\mathtt{10.0.0.0/8}, [\mathtt{1}], \mathbf{B0})\} \\ B &= \{(\mathtt{10.1.0.0/16}, [\mathtt{0}], \mathbf{I1}), (\mathtt{10.4.0.0/16}, [\mathtt{0}], \mathbf{I4})\} \\ A \otimes_E B &= \{(\mathtt{10.0.0.0/8}, [\mathtt{1}], \mathbf{B0})\} \\ B \oplus_E (A \otimes_E B) &= \{(\mathtt{10.0.0.0/8}, [\mathtt{1}], \mathbf{B0}), (\mathtt{10.1.0.0/16}, [\mathtt{0}], \mathbf{I1}), \{(\mathtt{10.4.0.0/16}, [\mathtt{0}], \mathbf{I4})\}\} \end{aligned}$$

As a consequence, we cannot make any claims regarding the convergence of Algorithm 4. However, the matrices produced according to this construct satisfy LEFT-DISTRIBUTIVITY. The proof of this is very similar to that of Lemma 6.2.4. Furthermore, we can define the reduction function ϕ^{IGP} which relates the contents of the matrices to the effective routes.

$$\phi_i^{IGP}(\mathbf{L}) = \min_{\lesssim_{BR}} \left(\bigcup_j \mathbf{L}_{i,j} \right)$$

It is possible to prove that at each iteration, the effective routes can be obtained from the contents of the corresponding matrix. This allows us to claim the applicability of our approach to route aggregation performed within the network as long as the metrics involved are LEFT-DISTRIBUTIVE.

$$\phi_i^{IGP}(\mathbf{L}^k) = \mathbf{eroutes}_i^k$$

Let us now apply this contraption to a scenario involving a forwarding loop. Its occurence within the network of ISP 2, Figure 6.15 comes from the fact that **B1** accepts as the best route for `10.0.0.0/8` a route that it originally injected into the network and passed through routers **R0** and **R1** before coming back.

In Figure 6.15, routers **B0** and **B1** are assumed to have an aggregation rule for prefix `10.0.0.0/8` configured on their respective links to **R0**. Given that both routers receive route advertisements for `10.1.0.0./16` and `10.3.0.0/16` respectively from their external peers, they will both start advertising routes for the prefix `10.0.0.0/8`. However the configuration of the rule at **B1** has the consequence that it will not advertise routes falling under the prefix `10.0.0.0/8` to **R1**. The sequence of events which leads to a forwarding loop can be summarized as follows

1 **R0** receives two routes for `10.0.0.0/8`, one from **B0** with an interior cost of 3 and another one from **B1** with interior cost 1. As a consequence, **R0** installs an entry for this new destination in its routing table with **B1** as the next-hop

2 **R1** receives a route advertisement from **R0** for `10.0.0.0/8` with interior cost 2 and installs an entry for this previously unknown destination in its routing table with **R0** as the next-hop

Router	Destination	Interior cost	`NEXT_HOP`
B0	`10.0.0.0/8`	4	**R0**
B1	`10.0.0.0/8`	3	**R1**
R0	`10.0.0.0/8`	1	**B1**
R1	`10.0.0.0/8`	2	**R0**

Figure 6.16: External routes in the routing tables after stabilisation of the distance-vectoring method.

3 **B1** receives a route advertisement from **R1** for `10.0.0.0/8` with interior cost 3 and installs an entry for this previously unknown destination in its routing table with **R1** as the next-hop

The resulting entries for the prefix `10.0.0.0/8` in the routing tables exhibit a forwarding loop involving the cycle $\mathbf{B1} \rightarrow \mathbf{R1} \rightarrow \mathbf{R0} \rightarrow \mathbf{B1}$. Note that **B0** would have an entry for the prefix `10.1.0.0/16` while the routing table of **B1** would contain an entry for the `10.3.0.0/16`. The consequence is that any traffic bound for `10.3.0.0/16` from any router will reach **B1** who will be able to get it out of the loop towards its intended destination. On the other hand, all traffic bound for any address in `10.0.0.0/8` but not in `10.3.0.0/16` will remain trapped in the forwarding loop. In practice, this problem is solved by having **B1** install a *sink-route* for `10.0.0.0/8`. Any traffic whose destination address matches the aggregate route `10.0.0.0/8` but not any more specific route would be dropped by the router. Given the four vertices $\{\mathbf{B0}, \mathbf{B1}, \mathbf{R0}, \mathbf{R1}\}$ on Figure 6.15, the weights on the arcs connecting them and the only aggregation rule for the prefix `10.0.0.0/8` configured on the arcs $\mathbf{B0} \rightarrow \mathbf{R0}$ and $\mathbf{B1} \rightarrow \mathbf{R0}$.

The entries of the adjacency matrix associate the aggregation rules with the distance of the link used and the next-hop. In order to mitigate the size of the matrices involved, we restrict ourselves to the routers **B0**, **B1**, **R0** and **R1**. The unexpected behaviour that **R1** does not receive any route advertisements from **B1** for `10.0.0.0/8` is expressed by setting the corresponding entry $\mathbf{A}_{\mathbf{R1},\mathbf{B1}}$ to the multiplicative annihilator.

$$\mathbf{A} = \begin{array}{c} \\ \mathbf{B0} \\ \mathbf{B1} \\ \mathbf{R0} \\ \mathbf{R1} \end{array} \begin{array}{c} \begin{array}{cccc} \mathbf{B0} & \mathbf{B1} & \mathbf{R0} & \mathbf{R1} \end{array} \\ \left[\begin{array}{cccc} \varnothing & \varnothing & (\bar{l}, 3, \mathbf{R0}) & \varnothing \\ \varnothing & \varnothing & (\bar{l}, 1, \mathbf{R0}) & (\bar{l}, 1, \mathbf{R1}) \\ (\texttt{10.0.0.0/8}, 3, \mathbf{B0}) & (\texttt{10.0.0.0/8}, 1, \mathbf{B1}) & \varnothing & (\bar{l}, 1, \mathbf{R1}) \\ \varnothing & \varnothing & (\bar{l}, 1, \mathbf{R0}) & \varnothing \end{array}\right] \end{array}$$

We only give the fixpoint limited to the columns **B0** and **B1** from which the forwarding loop arises. We can relate the entries of the fixpoint to those of the forwarding table by applying the ϕ_i^{IGP} function to each row of **L**. The result includes the forwarding loop $\mathbf{B1} \rightarrow \mathbf{R1} \rightarrow \mathbf{R0} \rightarrow \mathbf{B1}$.

$$\mathbf{L} = \begin{array}{c} \\ \mathbf{B0} \\ \mathbf{B1} \\ \mathbf{R0} \\ \mathbf{R1} \end{array} \begin{array}{c} \begin{array}{cc} \mathbf{B0} & \mathbf{B1} \end{array} \\ \left[\begin{array}{cc} \mathcal{I} \cup \{(\texttt{10.0.0.0/8}, 6, \mathbf{R0})\} & (\texttt{10.0.0.0/8}, 4, \mathbf{R0}) \\ (\texttt{10.0.0.0/8}, 4, \mathbf{R1}) & \mathcal{I} \cup \{(\texttt{10.0.0.0/8}, 3, \mathbf{R0})\} \\ (\texttt{10.0.0.0/8}, 3, \mathbf{B0}) & (\texttt{10.0.0.0/8}, 1, \mathbf{B1}) \\ (\texttt{10.0.0.0/8}, 4, \mathbf{R0}) & (\texttt{10.0.0.0/8}, 2, \mathbf{R0}) \end{array}\right] \end{array}$$

The application of the reduction function to the fixpoint produces the following sets of effective routes introduced in the routing table. The set of initial routes here could be replaced by any routes known prior to the injection process. For instance, the only initial route assumed for **B0** in the injection process is for `10.1.0.0/16` while **B1** has an effective route for `10.3.0.0/16`. We can see that routers **R1**, **R0** and **B1** are involved in a forwarding loop.

$$\begin{aligned} \phi^{IGP}(\mathbf{L}_{\mathbf{B0}}) &= \{(\texttt{10.0.0.0/8}, 4, \mathbf{R0})\} \cup \mathcal{I} \\ \phi^{IGP}(\mathbf{L}_{\mathbf{B1}}) &= \{(\texttt{10.0.0.0/8}, 3, \mathbf{R1})\} \cup \mathcal{I} \\ \phi^{IGP}(\mathbf{L}_{\mathbf{R0}}) &= \{(\texttt{10.0.0.0/8}, 1, \mathbf{B1})\} \cup \mathcal{I} \\ \phi^{IGP}(\mathbf{L}_{\mathbf{R1}}) &= \{(\texttt{10.0.0.0/8}, 2, \mathbf{R0})\} \cup \mathcal{I} \end{aligned}$$

We have shown how the model can be used to detect forwarding loops by using the exact aggregation matrix **A** which results from the configured rules along with the unexpected behaviour pointed out in [36]. At this stage, we are unable to provide a sufficient condition for their absence. Even in the presence of some form of LEFT-INFLATIONARITY to guarantee the convergence of Algorithm 4 to a fixpoint, this property does not suffice to exclude the formation of forwarding loops.

Chapter 7

Conclusion and future work

Route aggregation is a mecanism that is well integrated into the way routers discover and construct their forwarding and routing tables. In this work, we proposed to model the behavior of aggregation at two different levels with a unified approach involving IDEMPOTENT prebimonoids. We deliberately chose to limit ourselves to a subset of the metrics that describe routes in real-world implementation to simplify the model while keeping it easily extensible to encompass the complete set of metrics. This allowed us to study only the way prefixes and route aggregation can be modelled within an algebra framework. While we limited ourselves to a subset of metrics involved in route selection and left aside some considerations, the result hints at the adequacy of the algebraic approach to model the way information pertaining to network prefixes is exchanged among routers to construct the tables they use for forwarding purposes.

The conclusions of this attempt is that while the resulting algebra satisfies the properties of an IDEMPOTENT prebimonoid, they lack LEFT-INFLATIONARITY, be it in general or in the specific adjacency matrices that we use to encode aggregation scenarios. As such, the known results regarding the convergence of the algorithms to compute a fixpoint to a left equation cannot be applied to give a positive answer. On the other hand, in our attempts to model IGP, we identified the need for new results given that even when the computation does converge, forwarding loops may arise. By expanding the amount of information placed into the entries of the matrices used, we effectively propose an expression of the computation of the routes that exist between any two routers of a given infrastructure and relate this computation to the one a path-vectoring or distance-vectoring method relies on.

While these limitations do limit the applicability of the model at the moment, it seems to be a viable way to approach route aggregation problems in order to study the root causes for the formation of forwarding loops. We identify several shortcomings that need addressing in order to bring it even closer to reality.

7.1 Computation of effective routes

In this thesis, we introduced a formalization of the way BGP speakers compute the effective routes they use for forwarding purposes. This model captured the general approach underlying path-vectoring and distance-vectoring whereby each router performs some transformation on its effective routes to produce the routes that it advertises to its peers. The distance between this model and the reality could be reduced by introducing more refinement into the functions $f_{i,j}$ underlying the computation. In particular, the separation between origination and aggregation requires more study before obtaining a satisfying representation of the way the path-vectoring method works. On the other hand, while some attributes are determined by the process of aggregation such as `AS_SEQUENCE` or the internal cost, others may influence it. Specifically, we have the `MULTI_EXIT_DISC` attribute in mind which can prevent the aggregation of certain routes. While the extension of the attributes used beyond the `AS_SEQUENCE` and advertising peer, the attributes that influence aggregation should receive a careful treatment to properly reflect the results encoded in the fixpoint.

7.2 Applicability to non-distributive metrics

In our proof of the consistency of the results obtained by the computation of a fixpoint and the computation of the effective routes, we relied on the LEFT-DISTRIBUTIVITY and MONOTONICITY of the underlying metric involved. Recent work showed that non-DISTRIBUTIVE metrics are often used within a network. One best known example is when the bandwidth is part of the process which ranks paths, an information that is often involved in QoS and other traffic engineering methods. The question of whether MONOTONICITY of the metric is necessary for the consistency of the fixpoint computation and the path-vectoring process is an important point that requires addressing.

7.3 Conditions for the absence of forwarding loops

It is customary to consider that strict INFLATIONARITY is a sufficient condition for the algorithm to converge to loop-free paths. In our approach, we expanded the information contained within the matrices with the consequence that this condition is not sufficient enough in the context of a multipath problem where network prefixes are part of the entries of the matrices. Beside the consequence that INFLATIONARITY does not hold, we showed in the context of IGP that even under convergence, forwarding loops may arise. These results agree with those presented in [36] as to why forwarding loops form in the first place. This discovery hints at the need for new results which would shed light on the convergence when matrices that are not strictly INFLATIONARY are used with an algorithm from the Bellman-Ford family.

7.4 Full set of BGP metrics

We chose to limit ourselves to three pieces of information pertaining to routes by only considering the impact of the `AS_SEQUENCE` and the advertising peer in the Decision Process. In practice, BGP speakers consider a whole range of additional metrics to make its choice of best routes. Some of these metrics have been studied and formalized in the past as algebraic structure related to semirings. Given that they all admit some form of preference relation along with rules to transform them under exchange between routers, they could be naturally included within our approach but the consequences in terms of the resulting algebraic structure would require further study.

7.5 Simple loop detection

The standard version of BGP incorporates restrictions on the routes that are allowed to be considered for Phase II of its Decision Process. One of those restriction is on the loop-freedom of the `AS_SEQUENCE` of received routes. In practice, upon assigning a preference, a router will scan the `AS_SEQUENCE` and consider the route ineligible if it contains its own AS number. At the level of IGP, the protocols used include mecanism some form of loop-detection in the form of the split horizon or poisoned reverse mecanism. The loop-freedom on `AS_SEQUENCE` and split horizon could be integrated into the prebimonoid to bring the results encoded in the fixpoints much closer to those observed in the real-world under a given configuration.

7.6 Unexpected behavior of RIPv2

Le & al. showed that when route aggregation is configured in RIPv2 on some interface of a router has the consequence that some routes are no longer advertised out of some other interfaces. This peculiar behavior depends on the specific aggregate routes which are activated at some point due to the reception of a matching child route. This masking behavior is not well expressible in the current form of our model which only reduces the sets of routes based on the aggregation rules that are configured. At this stage, the prebimonoid we propose is not easily extensible to incorporate such a dropping of routes. Another issue is the process by which an adjacency matrix is produced given a certain configuration of the internal routers. The adjacency matrix used to illustrate the formation of the forwarding loop had to be produced by hand. The question of whether the adjacency matrix can be consistently produced from the configurations of the internal routers requires further inquiry. In [36], the authors argued that the problem of identifying whether a set of configured aggregation rules results in forwarding loops is NP-hard. While this conclusion does not bode well for practical applications of our model to detect forwarding loops, a sufficient condition for their absence would be enough for its usefulness.

Bibliography

[1] Abley, J., Lindqvist, K., Davies, E., Black, B., and Gill, V. IPv4 Multihoming Practices and Limitations. RFC 4116 (Informational), July 2005.

[2] Alim, M. A., and Griffin, T. G. On the interaction of multiple routing algorithms. In *Proceedings of the Seventh COnference on Emerging Networking EXperiments and Technologies* (New York, NY, USA, 2011), CoNEXT '11, ACM, pp. 7:1–7:12.

[3] Awduche, D., Chiu, A., Elwalid, A., Widjaja, I., and Xiao, X. Overview and Principles of Internet Traffic Engineering. RFC 3272 (Informational), May 2002. Updated by RFC 5462.

[4] Backhouse, R. C., and Carré, B. A. Regular Algebra Applied to Path-finding Problems. *Journal of Applied Mathematics 15* (1975), 161–186.

[5] Bellman, R. On a Routing Problem. *Quarterly of Applied Mathematics 16* (1958), 87–90.

[6] Bertsekas, D., and Gallager, R. *Data Networks (1st Ed.).* Prentice-Hall, Inc., 1987.

[7] Bertsekas, D. P. Distributed asynchronous computation of fixed points*. *Laboratory for Information and Decision Systems Technical Report Series* (1981).

[8] Billings, J. N., and Griffin, T. G. A Model of Internet Routing Using Semi-modules. *Lecture Notes in Computer Science 5827* (2009), 29–+.

[9] Carré, B. A. An algebra for network routing problems. *IMA Journal of Applied Mathematics 7*, 3 (1971), 273–294.

[10] Coltun, R., Ferguson, D., Moy, J., and Lindem, A. OSPF for IPv6. RFC 5340 (Proposed Standard), July 2008. Updated by RFCs 6845, 6860.

[11] Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. *Introduction to Algorithms*, 3rd ed. MIT Press, Cambridge, Massachusetts, 2009.

[12] Davey, B. A., and Priestley, H. A. *Introduction to lattices and order (2nd ed.).* Cambridge university press, 2002.

[13] Day, J. *Patterns in Network Architecture: A Return to Fundamentals.* Prentice Hall, 2008.

[14] de Almeida, V. T., and Güting, R. H. Using dijkstra's algorithm to incrementally find the k-nearest neighbors in spatial network databases. In *Proceedings of the 2006 ACM Symposium on Applied Computing* (New York, NY, USA, 2006), SAC '06, ACM, pp. 58–62.

[15] Deering, S., and Hinden, R. Internet Protocol, Version 6 (IPv6) Specification. RFC 2460 (Draft Standard), Dec. 1998. Updated by RFCs 5095, 5722, 5871, 6437, 6564, 6935, 6946, 7045, 7112.

[16] Dijkstra, E. W. A note on two problems in connexion with graphs. *Numerische Mathematik 1* (1959), 269–271.

[17] Dijkstra, E. W. On the role of scientific thought, August 1974.

[18] Dynerowicz, S., Colin, J.-N., and Schumacher, L. A distributed local-optimum distance vector algorithm for the metarouting approach. In *Communications (ICC), 2011 IEEE International Conference on* (June 2011), pp. 1–6.

[19] Dynerowicz, S., and Griffin, T. G. On the forwarding paths produced by internet routing algorithms. In *ICNP* (2013), IEEE, pp. 1–10.

[20] Eppstein, D. Finding the k-shortest paths. In *Foundations of Computer Science, 1994 Proceedings., 35th Annual Symposium on* (Nov 1994), pp. 154–165.

[21] Ford, L. R., and Fulkerson, D. R. *Flows in Networks.* Princeton University Press, 1962.

[22] Fredman, M. L., and Tarjan, R. E. Fibonacci heaps and their uses in improved network optimization algorithms. *J. ACM 34*, 3 (July 1987), 596–615.

[23] Fuller, V., and Li, T. Classless Inter-domain Routing (CIDR): The Internet Address Assignment and Aggregation Plan. RFC 4632 (Best Current Practice), Aug. 2006.

[24] Gondran, M. Algèbre linéaire et cheminement dans un graphe. *RAIRO - Operations Research - Recherche Opérationnelle 9*, V1 (1975), 77–99.

[25] Gondran, M., and Minoux, M. *Graphs, Dioids and Semirings: New Models and Algorithms (Operations Research/Computer Science Interfaces Series).* Springer Publishing Company, Incorporated, 2008.

[26] Griffin, T. G., and Gurney, A. J. T. Increasing bisemigroups and algebraic routing. In *RelMiCS* (2008), pp. 123–137.

[27] Griffin, T. G., and Sobrinho, J. L. Metarouting. In *SIGCOMM* (2005), pp. 1–12.

[28] Gurney, A. J. T. *Construction and verification of routing algebras.* PhD thesis, University of Cambridge, Computer Laboratory, April 2009.

[29] Gurney, A. J. T., and Griffin, T. G. Lexicographic products in metarouting. In *ICNP* (2007), pp. 113–122.

[30] Housley, R., Curran, J., Huston, G., and Conrad, D. The Internet Numbers Registry System. RFC 7020 (Informational), Aug. 2013.

[31] Kernighan, B. W. *The C Programming Language*, 2nd ed. Prentice Hall Professional Technical Reference, 1988.

[32] Khanna, A., and Zinky, J. The revised arpanet routing metric. *ACM SIGCOMM Computer Communication Review 19*, 4 (1989), 45–56.

[33] Knuth, D. E. A generalization of dijkstra's algorithm. *Inf. Process. Lett. 6*, 1 (1977), 1–5.

[34] Krioukov, D., claffy, k. c., Fall, K., and Brady, A. On compact routing for the internet. *SIGCOMM Comput. Commun. Rev. 37*, 3 (July 2007), 41–52.

[35] Kurose, J. F., and Ross, K. W. *Computer Networking: A Top-Down Approach*, 5th ed. Addison-Wesley Publishing Company, USA, 2009.

[36] Le, F., Xie, G. G., and Zhang, H. On route aggregation. In *Proceedings of the Seventh COnference on Emerging Networking EXperiments and Technologies* (New York, NY, USA, 2011), CoNEXT '11, ACM, pp. 6:1–6:12.

[37] Malkin, G. RIP Version 2. RFC 2453 (INTERNET STANDARD), Nov. 1998. Updated by RFC 4822.

[38] Minoux, M. Structures algébriques généralisées des problèmes de cheminement dans les graphes. *RAIRO - Operations Research - Recherche Opérationnelle 10*, V2 (1976), 33–62.

[39] Mondou, J.-F., Crainic, T. G., and Nguyen, S. Shortest path algorithms: A computational study with the c programming language. *Computers & Operations Research 18*, 8 (1991), 767 – 786.

[40] Moy, J. OSPF Version 2. RFC 2328 (INTERNET STANDARD), Apr. 1998. Updated by RFCs 5709, 6549, 6845, 6860.

[41] Nithin, M., Ao, T., and Dahai, X. Optimal Link-state Hop-by-hop Routing, 2013.

[42] ORAN, D. OSI IS-IS Intra-domain Routing Protocol. RFC 1142 (Historic), Feb. 1990. Obsoleted by RFC 7142.

[43] PEYER, S., RAUTENBACH, D., AND VYGEN, J. A generalization of dijkstra's shortest path algorithm with applications to {VLSI} routing. *Journal of Discrete Algorithms 7*, 4 (2009), 377 – 390.

[44] PEYER, S., RAUTENBACH, D., AND VYGEN, J. A generalization of dijkstra's shortest path algorithm with applications to vlsi routing. *J. of Discrete Algorithms 7*, 4 (Dec. 2009), 377–390.

[45] POSTEL, J. Internet Protocol. RFC 791 (INTERNET STANDARD), Sept. 1981. Updated by RFCs 1349, 2474, 6864.

[46] RAKHEJA, P., KAUR, P., GUPTA, A., AND SHARMA, A. Performance analysis of rip, ospf, igrp and eigrp routing protocols in a network. *International Journal on Computer Applications 48*, 18 (June 2012).

[47] REKHTER, Y., LI, T., AND HARES, S. A Border Gateway Protocol 4 (BGP-4). RFC 4271 (Draft Standard), Jan. 2006. Updated by RFCs 6286, 6608, 6793.

[48] ROMAN, S. *Lattices and ordered sets*. Springer, 2008.

[49] ROSEN, E. Exterior Gateway Protocol (EGP). RFC 827, Oct. 1982. Updated by RFC 904.

[50] SAUCEZ, D., IANNONE, L., BONAVENTURE, O., AND FARINACCI, D. Designing a deployable future internet: the locator/identifier separation protocol (lisp) case. *IEEE Internet Computing* (to appear 2012). http://www.computer.org/portal/web/csdl/doi/10.1109/MIC.2012.98.

[51] SAVAGE, D., SLICE, D., NG, J., MOORE, S., AND WHITE, R. Enhanced Interior Gateway Routing Protocol. Internet-Draft draft-savage-eigrp-02.txt, IETF Secretariat, April 2014.

[52] SOBRINHO, J. Algebra and algorithms for qos path computation and hop-by-hop routing in the internet. *Networking, IEEE/ACM Transactions on 10*, 4 (Aug 2002), 541–550.

[53] SOBRINHO, J. L. Network routing with path vector protocols: Theory and applications. In *SIGCOMM '03: Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications* (New York, NY, USA, 2003), ACM, pp. 49–60.

[54] SOBRINHO, J. L., AND GRIFFIN, T. G. Routing in equilibrium. In *Proceedings of the 19th International Symposium on Mathematical Theory of Networks and Systems (MTNS)* (2010).

[55] TARJAN, R. E. A Unified Approach to Path Problems. *Journal of the ACM 28*, 3 (1981), 577–593.

[56] TEIXEIRA, R., SHAIKH, A., GRIFFIN, T., AND REXFORD, J. Dynamics of hot-potato routing in ip networks. *SIGMETRICS Perform. Eval. Rev. 32*, 1 (June 2004), 307–319.

[57] TEIXEIRA, R., SHAIKH, A., GRIFFIN, T., AND REXFORD, J. Hot potatoes heat up bgp routing, 2004.

[58] THORUP, M., AND ZWICK, U. Compact routing schemes. In *Proceedings of the Thirteenth Annual ACM Symposium on Parallel Algorithms and Architectures* (New York, NY, USA, 2001), SPAA '01, ACM, pp. 1–10.

[59] VILLAMIZAR, C., CHANDRA, R., AND GOVINDAN, R. BGP Route Flap Damping. RFC 2439 (Proposed Standard), Nov. 1998.

[60] VOHRA, Q., AND CHEN, E. BGP Support for Four-Octet Autonomous System (AS) Number Space. RFC 6793 (Proposed Standard), Dec. 2012.

[61] YEN, J. Y. Finding the k-shortest loopless paths in a network. *Management Science 17*, 11 (1971), 712–716.