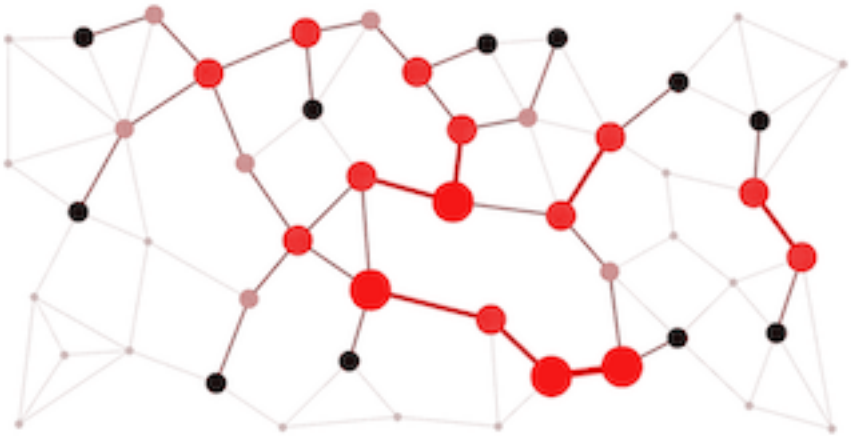
TÉCNICO LISBOA



Routing on Multiple Optimality Criteria: Theory and Protocols

Miguel Alves Ferreira

Thesis to obtain the Master of Science Degree in

Electrical and Computer Engineering

Supervisor: Prof. João Luís Costa Campos Gonçalves Sobrinho

Examination Committee

Chairperson: Prof. Teresa Maria Sá Ferreira Vazão Vasques
Supervisor: Prof. João Luís Costa Campos Gonçalves Sobrinho
Member of the Committee: Prof. João Manuel de Freitas Xavier

October 2020

Declaration

I declare that this document is an original work of my own authorship and that it fulfills all the requirements of the Code of Conduct and Good Practices of the Universidade de Lisboa.

Acknowledgments

A most grateful thank you to Beatriz, to my parents and to Professor João Luís Sobrinho.

I gratefully recognize the importance of *Instituto de Telecomunições*, where this research was carried out, and of *Fundação para a Ciência e Tecnologia*, for the scholarship offered under the scope of the Project “UID/EEA/50008/2019”.

Abstract

The concept of optimal path in a network is bound to: (a) a set of path attributes; (b) a binary extension operation on attributes, which calculates path attributes from link attributes; and (c) a total order on attributes, which defines an optimality criterion by establishing the relative preference among path attributes. Standard vectoring protocols, such as EIGRP, BGP, DSDV or Babel, only solve the problem of routing on optimal paths if the binary extension operation is isotone for the total order, thus leaving out many optimality criteria of practical interest. Isotonicity expresses that the relative preference between two attributes is preserved when they are extended with a third attribute.

We present a general solution to routing on optimal paths and, more broadly, to routing on multiple optimality criteria. A fundamental idea is the derivation of partial orders on attributes that satisfy isotonicity and respect every optimality criterion of a designated collection of such criteria. We design new routing protocols that compute on partial orders and have each node elect a set of attributes, rather than a single attribute, as standard vectoring protocols do. Our evaluation over realistic networks shows that the protocols devised require only a few elected attributes per destination and converge fast. The concepts and protocols introduced pave the way for solutions to other routing problems, not necessarily related to optimality.

Keywords

Routing Protocols; Optimality Criteria; Optimal Path Routing; Routing Algebras; Partial Orders.

Resumo

O conceito de caminho ótimo numa rede está vinculado a: (a) um conjunto de atributos de caminhos; (b) uma operação binária de extensão sobre atributos, que calcula atributos de caminhos a partir de atributos de ligações; e (c) uma ordem total sobre atributos, que define um critério de otimalidade estabelecendo a preferência relativa entre atributos de caminhos. Os protocolos vetoriais padrão, tais como EIGRP, BGP, DSDV ou Babel, só resolvem o problema do encaminhamento em caminhos ótimos se a operação binária de extensão for isótona para a ordem total, deixando assim de parte muitos critérios de otimalidade de interesse prático. Isotonicidade expressa que a preferência relativa entre quaisquer dois atributos é preservada quando estes são estendidos com qualquer terceiro atributo.

Apresentamos uma solução geral para o encaminhamento em caminhos ótimos e, de forma mais abrangente, para o encaminhamento em múltiplos critérios de otimalidade. Uma ideia fundamental é a derivação de ordens parciais sobre atributos que satisfaçam isotonicidade e respeitem todas as ordens totais que definem um critério de otimalidade numa designada colecção de tais critérios. Concebemos novos protocolos de encaminhamento que operam sobre ordens parciais e que em cada nó elege um conjunto de atributos, em vez de um único, como fazem os protocolos vetoriais padrão. A nossa avaliação sobre redes realistas mostra que os protocolos concebidos requerem apenas alguns atributos eleitos por destino e convergem rapidamente. Os conceitos e protocolos introduzidos abrem caminho a soluções para outros problemas de encaminhamento, não necessariamente relacionados com otimalidade.

Palavras Chave

Protocolos de Encaminhamento; Critérios de Otimalidade; Encaminhamento em Caminhos Ótimos; Álgebras do Encaminhamento; Ordens Parciais.

Contents

List of Figures

List of Tables

Acronyms

AS	Autonomous System
ASN	Autonomous System Number
BGP	Border Gateway Protocol
DSDV	Destination Sequenced Distance Vector
ECMP	Equal-Cost Multi-Path Routing
EIGRP	Enhanced Interior Gateway Routing Protocol
ISP	Internet Service Provider
OSPF	Open Shortest Path First
RIP	Routing Information Protocol
SS-BGP	Self-Stable BGP

1

Introduction

Contents

A routing problem states how data-packets should be guided from its sources to its destinations in a network. A routing protocol is a distributed algorithm for solving a given routing problem. In the problem of routing on optimal paths, every path in a network is characterized by a performance-related metric. Depending on the context, this metric may be any combination of hop-count [1], delay [2], transmission count [3], capacity [4], available bandwidth [4], and loss probability [5]. An optimality criterion is a relative valuation of a performance metric; an optimal path from a source to a destination is a path whose metric has as much or more value than the metrics of the remaining paths from source to destination. The goal in this problem is to route data-packets from its source to its destinations along optimal paths for a designated optimality criterion.

The problem of routing on optimal paths for an arbitrary optimality criterion does not yet have a solution. In this work, we present a solution to this problem - as a matter of fact, to a more general one, that of routing on optimal paths for multiple optimality criteria. In this new problem, the goal is to allow every flow of data-packets to be routed from its source to its destination on an optimal path for the optimality criterion that is most appropriate for that particular flow. My work on routing on multiple optimality criteria was developed in collaboration with my advisor, Professor João Luís Sobrinho, and resulted in the publication of a paper in the ACM SIGCOMM 2020 conference [1], which was awarded a "Best Paper Award". In this thesis, I offer my view of this joint work and emphasize the aspects in which I was most involved.

This introductory chapter is structured as follows. Section 1.1 characterizes the problem of routing on optimal paths. Section 1.2 outlines our solution to routing on multiple optimality criteria. Section 1.3 reviews previous work. Section 1.4 describes the organization of the thesis.

1.1 Problem Statement

Optimality. The optimal path to route a given flow of data-packets from its source to its destination depends on a combination of the source and the destination of the data-packets and on the type of information carried in them. For example, to transfer a file across the network, a quickest path is desired, which is a path that minimizes a linear combination of inverse capacity and delay [7]. To stream a video across the network, a wide-shortest path is desired, which is a path of minimum delay if there is one of sufficiently high bandwidth and a path of maximum bandwidth otherwise [8].

The concept of optimal path in a network is defined in general terms within an algebraic framework [9–13]. This framework is premised on:

(a) A set of attributes of links and paths in a network, which represents arbitrary performance metrics;

[1] "The ACM SIGCOMM conference is the premier venue to publish novel and significant research results in the field of computer networking" [6].

(b) A binary extension operation on attributes, which calculates path attributes from link attributes;

(c) A total order on attributes, which establishes the relative preference among path attributes and, thus, models an optimality criterion.

In a total order [14], two attributes are always comparable, one of them being preferred to the other. The optimal attribute from a source to a destination in a network is the most preferred from among the path attributes from the source to the destination and an optimal path is one whose attribute is optimal. For instance, if attributes are non-negative real numbers, the extension operation is addition and the total order is the less-than-or-equal order, then an optimal path is a shortest path.

Standard vectoring protocols. Standard vectoring protocols, such as RIP [1], EIGRP [15], BGP [16], DSDV [17] and Babel [18], compute path attributes at nodes to reach destinations separately per destination and expedite data-packets based exclusively on their destinations. These protocols have each node of a network iterate:

(a) A binary extension operation on attributes, which calculates path attributes to reach the destination from those advertised by neighbors;

(b) A selection operation in accordance with a total order on attributes, which elects a single attribute from among the path attributes to reach the destination via its neighbors.

Data-packets travel along the paths found having each intermediate node forwarding them to the neighbor from which the elected attribute was learned.

Standard vectoring protocols route on optimal paths only if the binary extension operation is isotone for the total order. Isotonicity means that the relative preference between any two attributes is preserved when they are extended with any third attribute [9–11]. Without isotonicity, standard vectoring protocols do not route on optimal paths, in general [11, 19, 20]. This property is often not met by concrete performance metrics. In fact, neither the total order pertaining to quickest paths nor the one pertaining to wide-shortest paths satisfy isotonicity.

Contributions. The main contribution of this work is a solution to routing on optimal paths for multiple optimality criteria, which entails a solution to routing on optimal paths for an arbitrary optimality criterion as a special case. We break down our contributions as follows:

(a) We formulate algebraically the problem of routing on multiple optimality criteria and introduce new routing concepts and constructions that promote a solution to this problem;

(b) We construct new routing protocols that support routing on multiple optimality criteria and show that these protocols satisfy desirable properties, such as termination and optimality;

(c) We evaluate by simulation of the practical feasibility and efficiency of the protocols proposed.

1.2 Solution Overview

We summarize our solution to optimal path routing for multiple optimality criteria and, as a special case, to optimal path routing for an arbitrary optimality criterion.

Optimal path routing for an arbitrary optimality criterion. The problem of routing on optimal paths has a solution only if the extension operation is isotone for the total order. What if isotonicity is not satisfied? We reduce the total order so that isotonicity is satisfied in the reduced order [21]. Reducing means letting some pairs of attributes that were comparable become now incomparable, none of the two being preferred to the other. This process leads to a partial order. In a partial order [14], one attribute is preferred to the other or the two are incomparable. The set of dominant attributes from a source to a destination in a network consists of the path attributes from the source to the destination that are not less preferred than any path attribute. This is a set of mutually incomparable attributes, in general, and it includes the optimal attribute for the original optimality criterion.

We design partial-order vectoring protocols, which generalize standard vectoring protocols to operate according to partial orders on attributes. These protocols compute path attributes at nodes to reach destinations separately per destination, as standard vectoring protocols do. However, they have each node elect a set of incomparable attributes from among the path attributes to the reach the destination via its neighbors, as opposed to a single attribute. Since each node possibly elects multiple attributes per destination, data-packets can no longer be forwarded exclusively based on destination. Therefore, the node differentiates its elected attributes to reach the destination by assigning them unique identifiers, which are advertised alongside the respective attributes [22]. Data-packets are guided along computed paths through label-switching at each intermediate node. If the extension operation is isotone for the partial order, then partial-order vectoring protocols route on dominant paths, namely on the original optimal paths.

Optimal path routing for multiple optimality criteria. The problem of routing on multiple optimality criteria is a new one. A naive solution to this problem would be to run a separate standard vectoring protocol per criterion. However, standard vectoring protocols route on optimal paths only if isotonicity is satisfied. Furthermore, even if isotonicity was satisfied for all criteria, this solution would not scale well: the number of criteria and, hence, the number of protocols, is possibly very large. How can we proceed? Our idea is further based on the reduction of total orders to partial orders. First, the total orders of a designated collection of optimality criteria are intersected. The intersection is a partial order that reduces each of the total orders. Second, the resulting partial order is once more reduced to satisfy

isotonicity. The optimal attributes for all optimality criteria are included in the respective sets of dominant attributes. Then, a partial-order vectoring protocol provides routing optimally and concurrently for all optimality criteria in the collection.

1.3 Previous Work

We discuss previous work on the algebraic conceptualization of routing, multi-objective path problems, and multi-path routing protocols.

Algebraic conceptualization of routing. The algebraic framework proposed in References [10, 11] provides means to reason about routing problems and protocols with generality, by abstracting away the specificity of performance metrics and protocol parameters. The framework is predicated on a set of attributes, a binary extension operation on attributes, and a total order on attributes.

The present work generalizes the framework by accepting partial orders on attributes and proposing routing protocols that compute on them. Furthermore, it presents a procedure that reduces a collection of total orders on attributes to a common partial order that respects all of them and satisfies isotonicity.

Multi-objective path problem. Multi-objective path problems are well-known in the operations research community. These problems can be formulated within the same algebraic framework considered in this work. Attributes are tuples of elementary metrics, each of which either extends with addition and is totally ordered by the less-than-or-equal order or extends with minimum and is totally ordered by the greater-than-or-equal order. Tuples extend coordinate-wise and are partially ordered by the product order of their coordinate-wise total orders. Because each coordinate extends with addition or minimum and is, therefore, individually isotone, the extension of tuples is isotone for the product order. The goal of multi-objective path problems is to find sets of dominant attributes [23–26].

In most cases, multi-objective path problems are solved by extensions of Dijkstra's algorithm [23, 24, 27] and Bellman-Ford algorithm [25–28] that compute on sets of incomparable attributes, rather than on a single attribute [2]. If attributes are tuples of several additive components, then the number of dominant attributes from source to destination can be exponential in the number of nodes in the network [23]. Approximate algorithms have been devised to deal with the explosion in the number of dominant attributes [30].

[2] Dijkstra's and Bellman-Ford algorithm are sequential algorithms that discover shortest paths from sources to destination [29].

The setting considered in work is more general and the problem addressed is different. Attributes are assumed arbitrary, not necessarily tuples of minimal or additive components. Even when they are, a partial order on them is derived, rather than assumed a priori, and does not necessarily coincide with the product order. The sets of dominant attributes are to be found by a routing protocol, that is, a distributed algorithm, rather than a sequential one, that can expedite data-packets along the paths computed.

Multi-path routing protocols. Multi-path routing protocols select multiple paths from a source to a destination. They have been mostly suggested as extensions to BGP [16], the prevailing standard vectoring protocol for routing on the Internet, with one of the following goals in mind. A first goal is to ensure the termination of BGP [31–35]. A second goal is to improve the capability of BGP to deliver data-packets during periods of convergence of the protocol following link failures [36–39]. And a third goal is to increase the path diversity available [33, 40, 41].

Since partial-order vectoring protocols possibly find multiple paths from a source to a destination, they are a type of multipath routing protocol. However, these protocols target a different goal. We seek to route data-packets on optimal paths for a variety of optimality criteria, some of which do not lend themselves to a solution by a standard vectoring protocol. Besides, partial-order vectoring protocols are formulated with generality rather than being specific to BGP.

1.4 Thesis Organization

We detail the organization of this thesis. Chapter 2 illustrates the optimal path routing problem for an arbitrary optimality criterion and for multiple optimality criteria. Chapter 3 reviews the algebraic theory for optimal path routing. Chapter 4 develops a procedure that starts with a generic collection of optimality criteria and ends with a partial order that respects each criterion and satisfies isotonicity. Furthermore, it designs partial-order vectoring protocols. Chapter 5 presents an evaluation of our solution to optimal path routing for multiple optimality criteria. Chapter 6 concludes this thesis and discusses future work. Appendix A presents a sequential algorithm that computes sets of dominant attributes.

2

Routing based on Widths and Lengths

Contents

We illustrate how partial-order vectoring protocols support routing on multiple optimality criteria. In the upcoming examples, every link and path in a network is characterized by a pair *width-length* from the *Cartesian product* [14] between positive widths and finite lengths. A *width* is a non-negative real number or infinity. Widths extend with minimum: the width of a path is the minimum of the widths of its links. A width stands for a metric like capacity [4] or available bandwidth [4]. A *length* is a non-negative real number or infinity. Lengths extended with addition: the length of a path is the sum of the lengths of its links. A length stands for a metric such as delay [2] or transmission count [3]. Consequently, the *extension* of a width-length (w, l) with a width-length (w', l') is $(\min\{w, w'\}, l + l')$.

The present chapter is organized as follows. Section 2.1 explores routing on shortest-widest paths. Section 2.2 goes on to explore routing on optimal paths concurrently for several optimality criteria premised on widths and lengths.

2.1 Routing on Shortest-Widest Paths

We show that a standard vectoring protocol that computes on the shortest-widest order fails in routing on shortest-widest paths. Contrastingly, we illustrate that a partial-order vectoring protocol that computes on an appropriate partial order on width-lengths succeeds in routing on shortest-widest paths.

Shortest-widest paths. A *shortest-widest path* from a source to a destination in a network is one of minimum length among those of maximum width [42]. Widths take precedence over lengths is the choice of optimal paths. The shortest-widest order is the total order underlying the selection of shortest-widest paths. In the *shortest-widest order* (lexicographic order) on width-lengths, a width-length (w, l) is *preferred* to a width-length (w', l') if its width is greater, $w > w'$, or if its width is the same and its length is smaller, $w = w'$ and $l < l'$. For example, width-length $(10, 5)$ is preferred to $(5, 2)$, because it has greater width, and $(10, 5)$ is preferred to $(10, 12)$, since it has the same width and smaller length. Figure 2.1 represents the shortest-widest order in the width-length plane. The set of width-lengths that are less preferred than width-length (w, l) is in green and the set of width-lengths that are preferred to (w, l) is in blue.

The *shortest-widest width-length* in a set of width-lengths is the most preferred width-length in the set. In particular, the shortest-widest width-length in set $\{(5, 2), (10, 12), (10, 5)\}$ is $(10, 5)$, on the account of $(10, 5)$ being preferred to both $(5, 2)$ and $(10, 12)$. A *shortest-widest path* from a source to a destination is one whose width-length is the shortest-widest in the set of all path width-lengths from the source to the destination. In the network of Figure 2.2, each link is annotated with a width-length and nodes want to route data-packets to destination x on shortest-widest paths. This network will serve as the running example throughout this chapter. The shortest-widest path from u to x is uwx, as the width-length of

uvx is $(5, 2) = (\min\{5, 20\}, 1 + 1)$, the width-length of $uwvx$ is $(10, 12) = (\min\{10, 20, 20\}, 3 + 5 + 1)$, the width-length of uwx is $(10, 5) = (\min\{10, 20\}, 3 + 2)$ and the shortest-widest width-length in set $\{(5, 2), (10, 12), (10, 5)\}$ is $(10, 5)$.

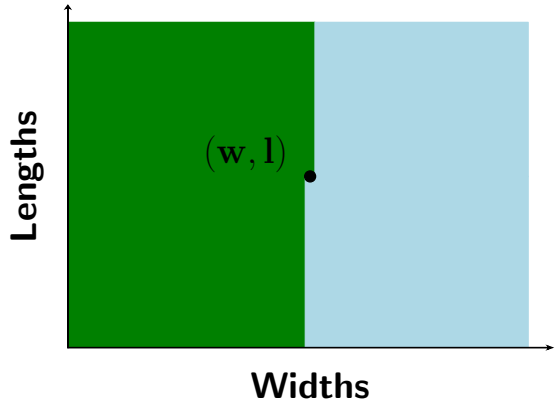


Figure 2.1: Shortest-widest order. The green area contains the width-lengths that are less preferred than (w, l) and the blue area contains those that are preferred to (w, l).

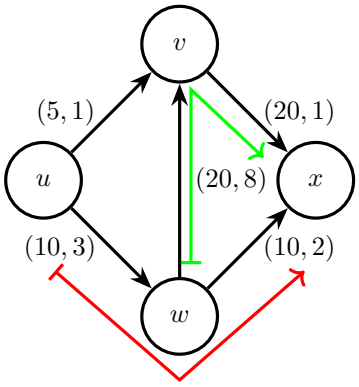


Figure 2.2: Shortest-widest paths from nodes u and w to destination x. Each link is annotated with a width-length.

With a *standard vectoring protocol*, nodes continually extend, elect and advertise to in-neighbors path width-lengths to reach the destination. The destination initiates the computation by advertising the width-length comprising infinite width and zero length to its in-neighbors. When a node receives a width-length from an out-neighbor, it first extends the width-length of the link to the out-neighbor with the width-length received from the out-neighbor, thus learning a candidate width-length to reach the destination via the out-neighbor. Then, the node elects the shortest-widest width-length from among its candidate width-lengths to reach the destination.

Returning to the network of Figure 2.2, destination x elects $(\infty, 0)$. Node v elects the candidate $(20, 1)$ via x, the extension of $(20, 1)$ with $(\infty, 0)$. Node w has a candidate $(20, 9)$ via v, the extension of $(20, 8)$ with $(20, 1)$, and a candidate $(10, 2)$ via x, the extension of $(10, 2)$ with $(\infty, 0)$. It elects $(20, 9)$, the shortest-widest width-length in the set $\{(20, 9), (10, 1)\}$. Node u has a candidate $(5, 2)$ via v, the extension of $(5, 1)$ with $(20, 1)$, and a candidate $(10, 12)$ via w, the extension of $(10, 3)$ with $(20, 9)$. It elects $(10, 12)$, the shortest-widest width-length in the set $\{(5, 2), (10, 12)\}$.

Figure 2.3 shows the stable state of the protocol. Each line in the table next to each node consists of a candidate width-length and the out-neighbor from which this width-length was learned. Data-packets are guided along the paths computed with each intermediate node forwarding them to the out-neighbor from which the elected attribute was learned. Node u forwards data-packets to w, which forwards them to v, which delivers them to x. Data-packets sourced at u and destined for x travel along path $uwvx$, whereas the shortest-widest path from u to x is uwx. The extension with $(10, 3)$ inverts the relative preference

between $(20, 9)$ and $(10, 2)$: width-length $(20, 9)$ is preferred to $(10, 2)$, but $(10, 12)$, the extension of $(10, 3)$ with $(20, 9)$, is less preferred than $(10, 5)$, the extension of $(10, 3)$ with $(10, 2)$. Node w hides $(10, 2)$ from u, although it would produce a width-length that is preferred to $(10, 12)$ if advertised to u.

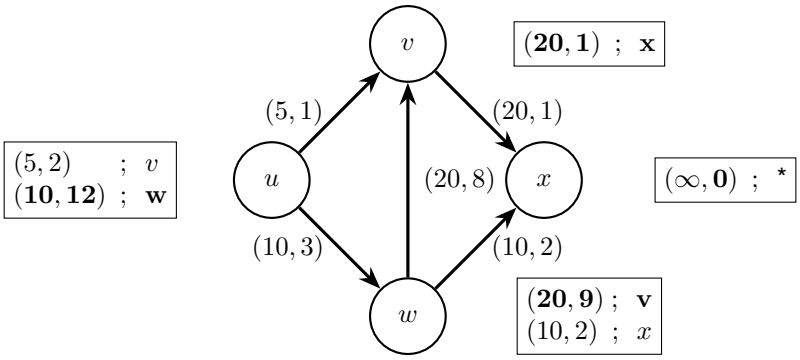


Figure 2.3: Stable state of a standard vectoring protocol operating on the shortest-widest order for destination x. Elected width-lengths are in bold.

The key property for the optimality of standard vectoring protocols is isotonicity. The extension of width-lengths is *isotone* for a total order on width-lengths if the relative preference between any two width-lengths is preserved when both are extended by any third width-length. With isotonicity, a candidate width-length at a node that is hidden would not yield a width-length that is preferred to the elected width-length at an in-neighbor of its if advertised. If isotonicity holds, then a standard vectoring protocol routes on optimal paths; if it does not hold, then a standard vectoring protocol does not route on optimal paths, in general. This result has been known for some time [11, 19, 20]. The shortest-widest order does not satisfy isotonicity. Therefore, routing on shortest-widest paths is not possible with a standard vectoring protocol.

Dominant paths. In the *product order* [14] on width-lengths, a width-length (w, l) is preferred to a width-length (w', l') if it is different, $(w, l) \neq (w', l')$, and both its width is greater or equal, $w \geq w'$, and its length is smaller or equal, $l \leq l'$. Two width-lengths are *incomparable*, none of them being preferred to the other, if one has greater width but the other smaller length. The product order is a partial order on width-lengths. For instance, width-lengths $(10, 5)$ and $(5, 2)$ are incomparable, on the account of $(10, 5)$ having greater width but $(5, 2)$ having smaller length, and yet $(10, 5)$ is preferred to $(5, 2)$ in the shortest-widest order. Width-length $(10, 5)$ is preferred to $(10, 12)$ in both the product order and the shortest-widest order, due to its equal width and smaller length. Figure 2.4 represents the product order in the width-length plane. The set of width-lengths that are incomparable with (w, l) is in white.

It is easy to show that the product order on width-lengths *is contained* in the shortest-widest order: a width-length that is preferred to another in the product order is also preferred to that other in the shortest-widest order. Note that the representation of the shortest-widest order, in Figure 2.1, contains

the representation of the product order, in Figure 2.4. In addition, the product order satisfies isotonicity. At length, suppose that a width-length (w, l) is preferred or equal to a width-length (w', l') or, in other words, that both $w \geq w'$ and $l \leq l'$. Let (x, m) be any width-length. From $w \geq w'$, we write $\min\{x, w\} \geq \min\{x, w'\}$, and from $l \leq l'$, we obtain $m + l \leq m + l'$. We conclude that $(\min\{x, w\}, m + l)$ is preferred or equal to $(\min\{x, w'\}, m + l')$. The product order is, thus, an *isotonic reduction* of the shortest-widest order. The concept of isotonic reduction is a fundamental idea of this work and is further explored in Chapter 4.

A width-length in a set of width-lengths is *dominant* if no width-length in the set is preferred to it. Concretely, the set of dominant width-lengths in set $\{(5, 2), (10, 12), (10, 5)\}$ is $\{(5, 2), (10, 5)\}$, since $(5, 2)$ and $(10, 5)$ are incomparable and $(10, 5)$ is preferred to $(10, 12)$. A *dominant path* from a source to a destination in a network is one whose width-length is dominant in the set of all path width-lengths from the source to the destination. In the network of Figure 2.5, the dominant paths from u to x are uwx and uvx, since the set of dominant width-lengths in set $\{(5, 2), (10, 5), (10, 12)\}$ is $\{(5, 2), (10, 5)\}$. The product order on width-lengths respects the shortest-widest order and so the shortest-widest path from u to x is found in the set of dominant paths from u to x.

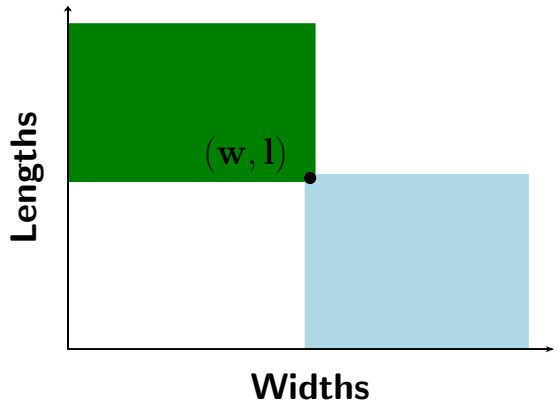


Figure 2.4: Product order on width-lengths. The white area contains the width-lengths that are incomparable with (w, l).

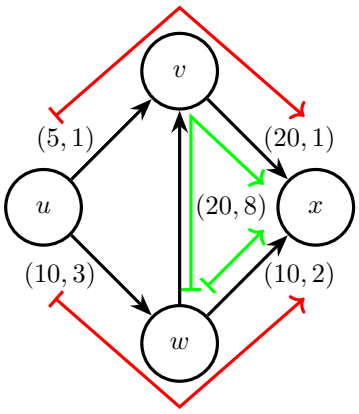


Figure 2.5: Dominant paths from nodes u and w to destination x.

With a *partial-order vectoring protocol*, nodes repeatedly extend, elect and advertise sets of incomparable width-lengths to reach the destination. When a node receives a set of width-lengths from an out-neighbor, it first extends the width-length of the link to the out-neighbor with every width-length received from the out-neighbor, thus obtaining a set of candidate width-lengths to reach the destination via the out-neighbor. Then, the node elects the set of dominant width-lengths from among its candidate width-lengths to reach the destination. It assigns a distinct label to each elected attribute to reach the destination. These labels are advertised to its in-neighbors alongside the respective attributes to enable the expedition of data-packets along the paths computed by switching labels at every intermediate

node [22]. The design of partial-order vectoring protocols is another fundamental idea of this work and is further discussed in Chapter 4.

Coming back to the network of Figure 2.5, destination x elects $(\infty, 0)$ and assigns it label 1. Node v elects the candidate $(20, 1)$ learned from x with label 1 and assigns it label 2. Node w has a candidate $(20, 5)$ learned from v with label 2 and a candidate $(10, 2)$ learned from x with label 1. It elects both width-lengths, because the set of dominant width-lengths in set $\{(20, 9), (10, 2)\}$ is itself, assigning label 4 to $(20, 9)$ and label 6 to $(10, 2)$. Node u has a candidate $(5, 2)$ learned from v with label 3 and candidates $(10, 12)$ and $(10, 5)$ learned from w with labels 4 and 6, respectively. It elects $(5, 2)$ and $(10, 5)$, since the set of dominant width-lengths in set $\{(5, 2), (10, 5), (10, 12)\}$ is $\{(5, 2), (10, 5)\}$, assigning label 3 to $(5, 2)$ and label 5 to $(10, 5)$.

Figure 2.6 shows the stable state of the protocol. Each line in the table next to each node is of the form:

(width, length) **,** *label* **;** *next.hop* **,** *next.label*,

and reads as follows:

(1) A data-packet originated at the node and meant to travel along a path with width-length *(width, length)* is forwarded to the out-neighbor *next.hop* carrying label *next.label*;

(2) A data-packet arriving at the node from an in-neighbor carrying label *label* is forwarded to the out-neighbor *next.hop* now carrying label *next.label*.

The product order on width-lengths satisfies isotonicity and so the partial-order vectoring protocol has the necessary routing and forwarding information to guide data-packets on any dominant path from u to x, namely on the shortest-widest path. Width-length $(10, 5)$ is the shortest-widest width-length from among the elected width-lengths at u to reach x. Therefore, node u forwards the data-packets to w carrying label 6. At w, incoming label 6 matches out-neighbor x and outgoing label 1. As a result, node w replaces label 6 with label 1 and delivers the data-packets to x.

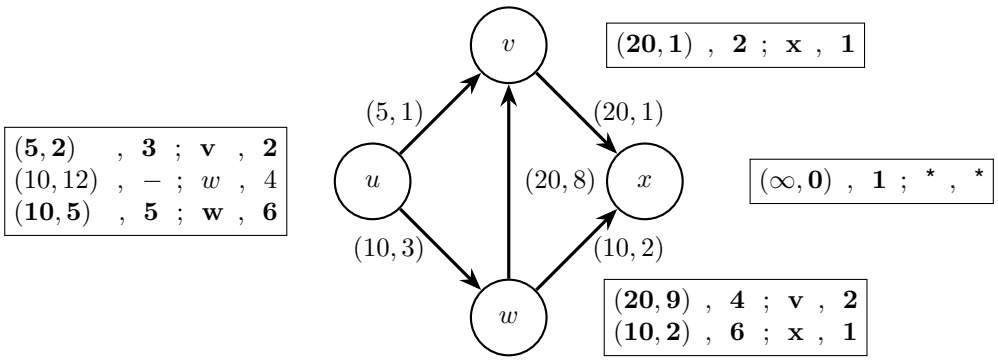


Figure 2.6: Stable state of a partial-order vectoring protocol operating on the product order on width-lengths for destination x.

To sum up, we showed that a standard vectoring protocol fails in routing on optimal paths when the total order underlying the choice of the optimal paths does not satisfy isotonicity. However, we suggested that a partial-order vectoring protocol that computes on a partial order on attributes that is contained in the original total order and satisfy isotonicity succeeds in routing on the original optimal paths.

2.2 Routing on Several Width-Length Criteria

We show that a partial-order vectoring protocol that operates on the product order on width-lengths allows routing concurrently on shortest-widest and widest-shortest paths. Then, we introduce two parameterized families of optimality criteria premised on widths and lengths for which this partial-order vectoring protocol also allows routing optimally and concurrently.

Widest-shortest paths. A widest-shortest path from a source to a destination in a network is one of maximum width among those of minimum length [42]. As opposed to shortest-widest paths, lengths take precedence over widths in the choice of optimal paths. The widest-shortest order is the total order underlying the selection of widest-shortest paths. In the *widest-shortest order* (co-lexicographic order) on width-lengths, a width-length (w, l) is preferred to a width-length (w', l') if its length is smaller, $l < l'$, or if its length is the same and its width is greater, $l = l'$ and $w > w'$. To illustrate, width-length $(5, 2)$ is preferred to $(10, 5)$ and to $(10, 12)$, on the account of its smaller length, whereas $(10, 5)$ was preferred to both $(5, 2)$ and $(10, 12)$ in the shortest-widest order. Figure 2.7 represents the widest-shortest order in the width-length plane. In the network of Figure 2.8, the widest-shortest path from u to x is uvx, because the widest-shortest width-length in set $\{(5, 2), (10, 5), (10, 12)\}$ is $(5, 2)$.

The product order on width-lengths respects the widest-shortest order as well as the shortest-widest order. Actually, the product order is the *intersection* between the shortest-widest and the widest-shortest order: a width-length is preferred to another in the product order precisely when it is preferred to that other in both the shortest-widest and the widest-shortest order. Observe that the intersection between representations of the shortest-widest order, in Figure 2.1, and the widest-shortest order, in Figure 2.7, corresponds to the representation of the product order, in Figure 2.4. The set of dominant paths from u to x consists of the widest-shortest path and the shortest-widest path from u to x.

Let us go back to Figure 2.6. With a partial-order vectoring protocol that operates on the product order, node u is able to route flows of data-packets to destination x on shortest-widest or widest-shortest paths, as it is more convenient for each specific flow. Suppose that u wants to route a flow of data-packets to x along the widest-shortest path. Width-length $(5, 2)$ is the widest-shortest width-length from among the elected width-lengths at u to reach x. Therefore, node u marks the data-packets with label 2 and forwards them to v. At v, label 2 is swapped for label 1 and the data-packets are delivered to x.

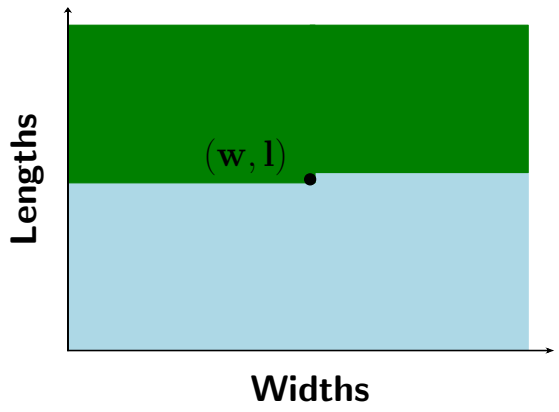


Figure 2.7: Widest-shortest order.

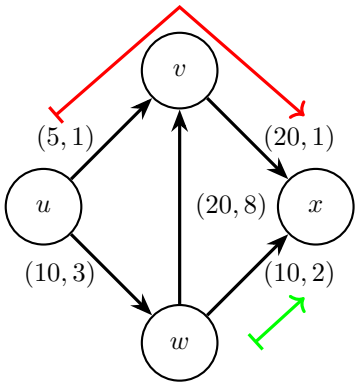


Figure 2.8: Widest-shortest paths from nodes u and w to destination x.

K-quickest and W-wide-shortest paths. The time to transfer a file of size K along a path of capacity w and delay l is $K/w+l$. A *K-quickest path* from a source to a destination is one that minimizes the time to transfer such a file [7]. Accordingly, in the *K-quickest order*, a width-length (w, l) is preferred to a width-length (w', l') if:

$$K/w + l < K/w' + l',$$
$$\text{or } K/w + l = K/w' + l' \text{ and } w > w'.$$

To exemplify, when K takes the value of 100, width-length $(20, 5)$ is preferred to $(10, 2)$, because $100/20+5 = 10 < 12 = 100/10 + 2$, and $(10, 5)$ is preferred to $(10, 8)$, since $100/10 + 5 = 15 < 18 = 100/10 + 8$. As the value of K grows, widths prevail and the K-quickest order approaches the shortest-widest order; on the other hand, as the value of K shrinks, lengths prevail and the K-quickest order approaches the widest-shortest order. Figure 2.9 represents the K-quickest order in the width-length plane.

To stream a video of maximum rate W from a source to a destination, a W-wide-shortest path is desired. A *W-wide-shortest path* from a source to a destination is one of minimum number of links among those of available bandwidth higher than W; if the available bandwidth of every path from the source to the destination is lower than or equal to W, then a W-wide-shortest path is one of maximum available bandwidth [8]. In the *W-wide-shortest order*, a width-length (w, l) is preferred to a width-length (w', l') if:

$$w > W \text{ and } (w' \leq W \text{ or } (w, l) \prec_{\text{ws}} (w', l')),$$
$$\text{or } w \leq W \text{ and } (w, l) \prec_{\text{sw}} (w', l'),$$

where $\prec_{\mathsf{ws}}$ and $\prec_{\mathsf{sw}}$ read as (w, l) is preferred to (w', l') in the widest-order order and the shortest-widest order, respectively. If there is a path from the source to the destination with width greater than W, then a W-wide-shortest path is a widest-shortest path; otherwise, it is a shortest-widest path. For example, when W takes the value of 15, width-length $(20, 5)$ is preferred to $(10, 2)$, because it is the only one of the two with width greater than 15, and $(10, 5)$ is preferred to $(10, 8)$, since $(10, 5)$ is preferred to $(10, 8)$ in the shortest-widest order. Figure 2.10 represents the W-wide-shortest order in the width-length plane.

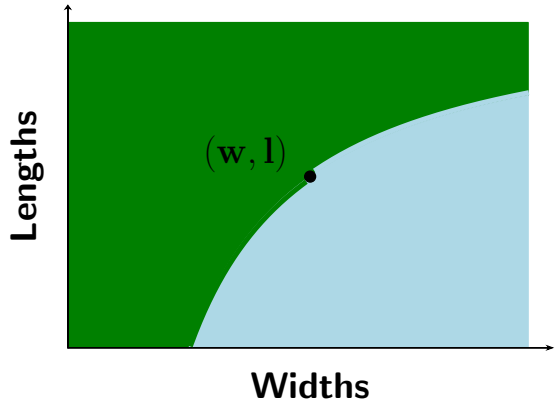


Figure 2.9: K-quickest order.

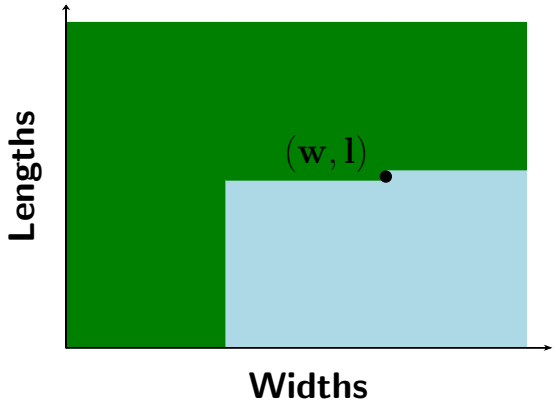


Figure 2.10: W-wide shortest order.

Neither the K-quickest order nor the W-wide-shortest order satisfy isotonicity, except when K and W equal 0, in which case it is easy to see that the K-quickest and W-wide-shortest order amount to the widest-shortest order. In fact, when K takes the value of 100, width-length $(20, 5)$ is preferred to $(10, 2)$, but $(10, 8)$, the extension of $(10, 3)$ with $(20, 5)$, is less preferred than $(10, 5)$, the extension of $(10, 2)$ with $(10, 3)$. When W takes the value of 15, the relative preferences between $(20, 5)$ and $(10, 2)$ is also inverted when both width-lengths are extended with $(10, 3)$. Routing optimally on any of these optimality criteria is not possible with vectoring protocols.

However, it is easy to verify that the product order on width-lengths is contained in the K-quickest and the W-wide shortest order, for all values of K and W. Therefore, a partial-order vectoring protocol that operates on the product order allows routing optimally and concurrently on K-quickest paths and W-wide-shortest paths, for all values of K and W.

To conclude, we further suggested that a partial-order vectoring protocol that computes on an isotone partial order supports routing optimally and concurrently for any optimality criterion defined by a total order that contains the partial order.

3

A Review of the Algebraic Theory for Routing on Optimal Paths

Contents

We revisit the algebraic theory for routing on optimal paths. The theory provides a unified treatment of optimal path routing by abstracting away the specificity of performance metrics to focus on a few algebraic properties that impact the behavior of standard vectoring protocols. The key insight of this theory is that these protocols solve the problem of routing on optimal paths only if the algebraic property of isotonicity is satisfied. The present chapter is structured in the following way. Section 3.1 introduces the basic algebraic routing concepts. Section 3.2 reviews standard vectoring protocols.

3.1 Optimality

We describe an algebraic framework that allows us to reason about optimal path problems for arbitrary performance metrics that meet a handful of algebraic properties. This framework is premised on the following abstractions:

(a) A set S of *attributes*. Attributes represent arbitrary performance metrics. Every link and path in a network is associated with an attribute.

(b) A *binary extension operation* on attributes. The binary extension operation, which we denote by $\oplus$, allows the calculation of the attribute of a path from the attributes of its constituent links.

(c) A *total order* on attributes. The total order, which we denote by $\preceq_i$, defines an optimality criteria i and, therefore, models the relative preference among attributes.

Properties of extension and path attributes. We state the properties assumed for a binary extension operation and show how path attributes are obtained from link attributes. A binary extension operation $\oplus$ on attributes satisfies *associativity* and *commutativity* and has an identity element, which we designate by *neutral attribute* and denote by ϵ:

$$(a \oplus b) \oplus c = a \oplus (b \oplus c), \text{ for all attributes } a, b, c; \quad \text{(associativity)} \tag{3.1}$$

$$a \oplus b = b \oplus a, \text{ for all attributes } a, b; \quad \text{(commutativity)} \tag{3.2}$$

$$\epsilon \oplus a = a, \text{ for all attributes } a. \quad \text{(existence of identity)} \tag{3.3}$$

Given a network, we denote the attribute of a link uv by $a(uv)$. The attribute of a path $P = u_0 u_1 \dots u_n$ is symbolized by $a(P)$. A path is *trivial* if it is composed of a single node and *non-trivial* otherwise. If P is a trivial path, then its attribute is ϵ. Otherwise, in case P is non-trivial, its attribute is the extension among the attributes of its constituent links:

$$a(P) = a(u_0 u_1) \oplus a(u_1 u_2) \oplus \dots \oplus a(u_{n-1} u_n). \tag{3.4}$$

Total order and optimal paths. We recall the notions of total order and optimal path. A *total order* $\preceq_i$ on attributes is a binary relation over S [1] that satisfies *antisymmetry*, *transitivity* and *connexity*:

$$a \preceq_i b \text{ and } b \preceq_i a \text{ imply } a = b, \text{ for all attributes } a, b; \quad \text{(antisymmetry)} \tag{3.5}$$

$$a \preceq_i b \text{ and } b \preceq_i c \text{ imply } a \preceq_i c, \text{ for all attributes } a, b, c; \quad \text{(transitivity)} \tag{3.6}$$

$$a \preceq_i b \text{ or } b \preceq_i a, \text{ for all attributes } a, b. \quad \text{(connexity)} \tag{3.7}$$

We write $a \prec_i b$ for $a \preceq_i b$ and $a \neq b$. When $a \prec_i b$, we say that a is *i-preferred* to b and that b is *less i-preferred* than a. Connexity means that an attribute is always is i-preferred to another and, therefore, two attributes are *comparable* every time. We assume there is a least i-preferred of all attributes. The antisymmetry of $\preceq_i$ implies a least i-preferred of all attributes is unique. We designate it by *null attribute* and denoted by $\bullet$:

$$a \preceq_i \bullet, \text{ for all attributes } a. \tag{3.8}$$

The *i-optimal* attribute in a set A of attributes, which we denote by $O_i(A)$, is the attribute of A that is i-preferred to every other attribute of the set. Given a network, the *i-optimal* attribute from source s to destination t in the network, which we denote by $a_i^*(s,t)$, is the i-optimal attribute in the set of all path attributes from s to t. A path from s to t is i-optimal if its attribute is $a_i^*(s,t)$.

Inflation and isotonicity. We introduce two fundamental algebraic properties that entangle the binary extension operation on attributes and the total order. These algebraic properties influence the ability of standard vectoring protocols to meet desirable properties, such as termination and optimality.

Definition 3.1. Binary extension operation $\oplus$ on attributes is *inflationary* for total order $\preceq_i$ if:

$$b \preceq_i a \oplus b, \text{ for all attributes } a, b. \tag{3.9}$$

Inflation expresses that the attribute of a path does not become i-preferred when this path is prefixed by another path [10, 43]. This property is related to the termination of standard vectoring protocols to stable states that guide data-packets from sources to destinations. The exact nature of this relationship depends on the class of standard vectoring protocols and is explored further in Section 3.2. Inflation is usually satisfied by concrete performance metrics and, consequently, it is assumed throughout the text.

We show how inflation causes the binary extension operation on attributes and the total order to become intertwined.

[1] A binary relation R over a set A is a subset of the Cartesian product between A and itself or, put differently, it is a set of pairs of elements of A. When $(a, b) \in R$, we write $a \, R \, b$.

Proposition 3.1. *Attribute ϵ is the most i-preferred of all attributes:*

$$\epsilon \preceq_i a, \text{ for all attributes } a. \tag{3.10}$$

Proof. Let $a \in S$. Because of inflation and commutativity of $\oplus$, we obtain $\epsilon \preceq_i a \oplus \epsilon = \epsilon \oplus a$. And because ϵ is the identity element of $\oplus$, it follows that $\epsilon \preceq_i \epsilon \oplus a = a$. □

Proposition 3.2. *Binary extension operation $\oplus$ on attributes has an absorbing element given by $\bullet$:*

$$\bullet \oplus a = \bullet, \text{ for all attributes } a. \tag{3.11}$$

Proof. Let $a \in S$. Using inflation and commutativity of $\oplus$, we write $\bullet \preceq_i a \oplus \bullet = \bullet \oplus a$. Since $\bullet$ is the least i-preferred of all attributes, we have $\bullet \oplus a \preceq_i \bullet$. The antisymmetry of $\preceq_i$ together with $\bullet \preceq_i \bullet \oplus a$ and $\bullet \oplus a \preceq_i \bullet$ imply $\bullet \oplus a = \bullet$. □

Definition 3.2. Binary extension operation $\oplus$ is *isotone* for total order $\preceq_i$ on attributes if:

$$b \preceq_i c \text{ implies } a \oplus b \preceq_i a \oplus c, \text{ for all attributes } a, b, c. \tag{3.12}$$

Isotonicity conveys that the relative preference between the attributes of two paths is preserved when these paths are prefixed by a third path [9,10,19]. This property is associated with the optimality of standard vectoring protocols. With isotonicity, standard vectoring protocols find the i-optimal attributes from sources to destination; without isotonicity, these attributes are not found, in general. In contrast to inflation, isotonicity is not met by most concrete performance metrics and, consequently, is not assumed throughout the text. (See Table 3.1.)

Examples. In summary, we detailed an algebraic framework that enables us to formulate problems of optimal paths and to devise algorithms for these problems for any performance metrics that satisfy the algebraic properties of associativity, commutativity and inflation. To conclude this section, Table 3.1 shows concrete examples of the algebraic framework from the context of routing based on widths and lengths, as introduced in Chapter 2. The usual name of the optimal path is given in the first column. In the table, the examples marked with ‡ satisfy isotonicity.

3.2 Protocols for Optimal Paths

In the problem of routing on optimal paths, the goal is to route data-packets from sources to destinations along optimal paths for a designated optimality criterion. Links fail and are added over time.

Table 3.1: Examples of the algebraic framework. Those marked with ‡ satisfy the algebraic property of isotonicity.

Optimal Paths	$\mathbf{S}$	$\oplus$	ϵ	$\preceq_{\mathbf{i}}$	$\bullet$
Shortest paths ‡	$\mathbb{R}_0^+ \cup \{\infty\}$	$+$	0	$\leq$	∞
Widest paths ‡	$\mathbb{R}_0^+ \cup \{\infty\}$	$\min$	∞	$\geq$	0
Shortest-widest paths (SW)	$\{(w,l) \mid w \in \mathbb{R}^+ \cup \{\infty\}, l \in \mathbb{R}_0^+\} \cup \{(0,\infty)\}$	$(\min\{w', w\}, l' + l)$	$(\infty, 0)$	$w > w'$; or $w = w'$ and $l \leq l'$	$(0, \infty)$
Widest-shortest paths (SW) ‡				$l < l'$; or $l = l'$ and $w \geq w'$	
K-quickest paths (K-Q)				$K/w + l < K/w' + l'$; or $K/w + l = K/w' + l'$ and $w > w'$	
W-wide-shortest paths (W-W*S)				$w > W$ and $(w' \leq W$ or $(w,l))$; or $w \leq W$ and $(w,l) \prec_{\mathsf{sw}} (w',l')$	

Standard vectoring protocols, such as RIP [1], EIGRP [15], BGP [16], DSDV [17] and Babel [18], have a destination initiate a computation instance by advertising to its in-neighbors the neutral attribute. They have each node elect and advertise to its in-neighbors the best from among the attributes learned from its out-neighbors.

A standard vectoring protocol belongs to one of two classes, the class of standard non-restarting vectoring protocols or the class of standard restarting vectoring protocols [20]. They differ in the amount of routing state held at each node and in their behavior in the absence of isotonicity. In the remainder of this chapter, we formalize the operation of these classes of protocols and discuss the algebraic requirements that guarantee they terminate to stable states that guide data-packets along optimal paths from sources to destinations. Section 3.2.1 concerns the class of standard non-restarting vectoring protocols and Section 3.2.2 concerns the class of standard restarting vectoring protocols.

3.2.1 Non-Restarting Protocol

With a *standard non-restarting vectoring protocol*, a destination maintains a single computation instance all through time. A node stores a candidate attribute to reach the destination via each of its out-neighbors; at all times, it elects the most preferred from among its candidates attributes to reach the destination. The elected attribute improves and worsens throughout the computation instance. The election of a new attribute that is worse than the old one triggers the process of some nodes settling on worse attributes than those they started out with by persistently electing and advertising to in-neighbors the best from among the candidates learned from out-neighbors. This process is possibly long-lasting and, in the meantime, may see data-packets trapped in forwarding-loops. The stable state of the protocol delivers data-packets from sources to destinations, although not necessarily along optimal paths.

Non-restarting protocols are traditional from wired networks, with RIP [1], EIGRP [15] and BGP [16] as prototypical examples.

3.2.1.A Distributed Path Computation

We describe how non-restarting protocols carry out the distributed computation of paths from sources to destinations and how data-packets are expedited along the paths computed.

In the canonical standard non-restarting vectoring protocol, destination t initiates the computation by advertising attribute ϵ to its all in-neighbors. This event is designated by the network-wide announcement of destination t. Algorithm 1 gives the pseudo-code of the protocol for when node u receives the advertisement of attribute b from out-neighbor v pertaining to destination t, $u \neq t$ [11]. Variable $optTab_u[t, v]$ stores the candidate attribute at u to t via v and variable $opt_u[t]$ stores the elected attribute at u to reach t.

Upon receiving b from v, node u updates its candidate attribute via v to the extension of the attribute of its link to v with b (line 1). Then, it finds its elected attribute as the i-optimal attribute in the set of all its candidate attributes (line 2). If the elected attribute has changed, then u advertises this attribute to all its in-neighbors (line 3-5). We assume advertisements are subject to a variable propagation delay and are delivered in first-in, first-out order.

Algorithm 1 Canonical standard non-restarting vectoring protocol. When node u receives the advertisement of attribute b from out-neighbor v pertaining to t, $u \neq t$.

1: $optTab_u[t, v] \coloneqq a(uv) \oplus b$
2: $opt_u[t] \coloneqq O_i(\{\, optTab_u[t, v] \mid v \text{ out-neighbor of } u \,\})$
3: **if** $opt_u[t]$ has changed **then**
4: **for** all r in-neighbor of u **do**
5: Send $opt_u[t]$ to r

With the failure of link uv, for all destinations t, $u \neq t$, node u withdraws its candidate attribute via v, re-elects a new attribute from among its remaining candidate attributes and advertises this attribute to all its in-neighbors in case it has changed.

Each node maintains a *forwarding table* with entries of the form:

destination **;** *next.hop*,

where *destination* is a destination and *next.hop* is the out-neighbor of the node from which the elected attribute was learned. Data-packets meant for *destination* are forwarded to *next.hop* regardless of whether

they have been originated at the node or they have arrived at the node from an in-neighbor of its, a strategy named *destination-based forwarding*. A node may install multiple entries with common values of *destination* and different values of *next.hop*. This option allows data-packets to travel along multiple elected paths with a common attribute, a possibility known as ECMP.

3.2.1.B Termination and Optimality

We explore the algebraic conditions that ensure the termination and optimality in stable state of non-restarting protocols. Then, on a different vein, we assess the relationship between isotonicity and the speed of convergence of these protocols.

We first discuss termination. The *state* of a non-restarting protocol at a given instant corresponds to the candidate and elected attributes at every node, plus the advertisements of attributes in transit across links. This state is *stable* if there are no advertisements in transit, thus implying the distributed path computation has stopped. We have the following definition.

Definition 3.3. Given a network, consider an instant in time from which there are no changes in the network. A non-restarting protocol *terminates* in the network if there is a future instant in time from which the state of the protocol is stable, whatever the initial state of the protocol and whatever the delays of advertisements across links.

The key property for the termination of a non-restarting protocol is inflation. However, this property alone does not guarantee the protocol terminates in a given network. In addition, the network must meet two algebraic properties. First, the set of all path attributes in the network must be finite. This property can be enforced by including a *hop-count* component in the attributes that is incremented with every advertisement and invalidating attributes with hop-count greater than some maximum value, as in RIP [1]. Second, all circuits in the network must be strictly inflationary. An attribute a is *strictly inflationary* if $b \prec_i a \oplus b$, for all non-null attributes b, and a circuit in a network is strictly inflationary if its attribute is strictly inflationary. [2] Returning Table 3.1, for each of the examples presented, except widest paths, all attributes other than the neutral attribute are strictly inflationary.

Theorem 3.1 ([11])**.** *If the set of all path attributes is finite and all circuits are strictly inflationary in a given network, then a standard non-restarting vectoring protocol terminates in the network.*

We have the following two examples.

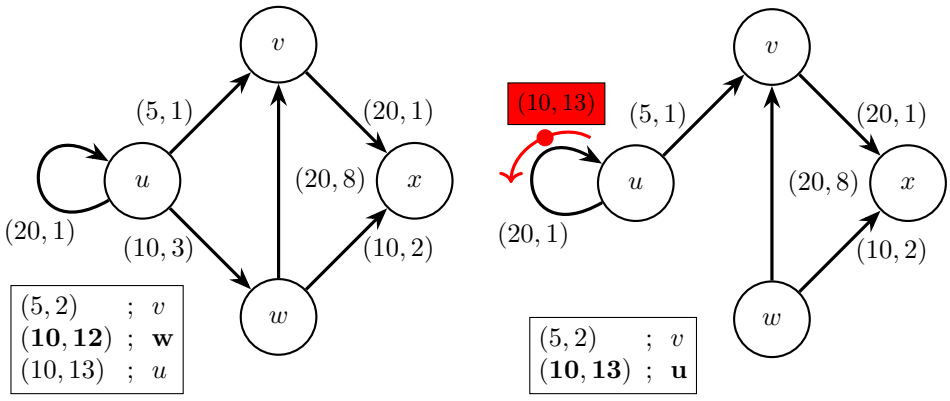
Example 3.1. In Figure 3.1, we show that if there are infinitely many paths attributes in a given network, then some nodes may be involved in counts-to-infinity. In the network of the figure, each link is annotated

[2] A *circuit* in a network is a path in which at least one node is repeated.

with a width-length. The self-loop at u is a synthetic representation of a circuit containing u. This network will serve as a running example to illustrate the interplay between algebraic properties and protocol behaviors. We assume that the goal is to route data-packets on shortest-widest paths to destination x. Figure 3.1(a) shows the stable state of the protocol at u. Node u elects $(10, 12)$ learned from w. In addition, it has a candidate $(5, 2)$ via v and a candidate $(10, 13)$ via around the self-loop.

With the failure of link uw, node u elects the candidate $(10, 13)$ via around the self-loop. Figure 3.1(b) shows the state of the protocol at u after the failure. Thereafter, node u repeatedly elects the width-length learned from around the self-loop, in detriment of the candidate $(5, 2)$ via v, and advertises this width-length back around the self-loop. With each advertisement, the width-length learned from around the self-loop retains its width and increments its length by one. Because lengths are not upper bounded, the advertisements perpetuate and the protocol does not reach a new stable state.

The failure of a link may imply that in a new stable state some nodes elect worse attributes than they did in the previous stable state. Trying to settle on worse attributes by always electing and advertising to in-neighbors the best attribute from among those learned from out-neighbors takes many iterations, with as many paths being explored [44]. If nodes have to range over infinitely many paths before stabilizing on a new attribute, then the protocol fails to terminate. With a non-restarting protocol for shortest-paths, or a *distance-vector protocol*, this behaviour is known as *count-to-infinity* [45].

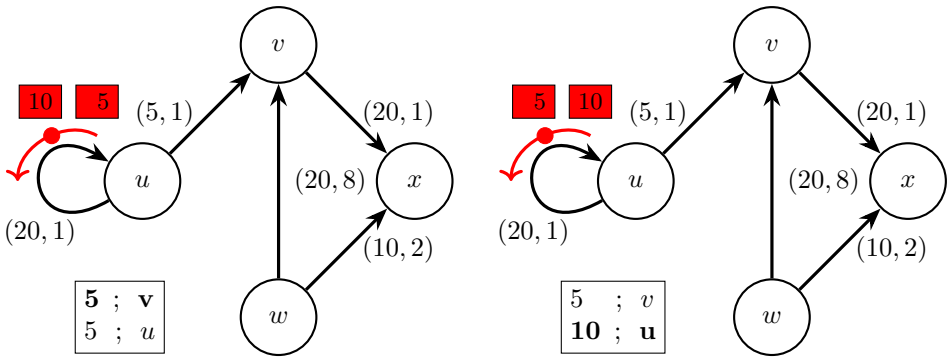


(a) Stable state. **(b)** State after the failure of uw.

Figure 3.1: Count-to-infinity with a non-restarting protocol for shortest-widest paths for destination x. Links are annotated with width-lengths. Elected width-lengths are in bold. Advertisements in transit are in red.

Example 3.2. In Figure 3.2, we show that if there is a circuit in a given network that is not strictly inflationary, then the protocol may oscillate perpetually. The goal is to route data-packets on widest paths to destination x. The width of the self-loop at u is not strictly inflationary: it is not the case that $w > \min\{20, w\}$ for all widths w; in particular, $w > \min\{20, w\}$ is false for $w \leq 20$. Destination x initiates the computation by advertising ∞ to v and w. Later, node u learns 5 from v and 10 from w. It elects 10 from w and advertises this width around the self-loop.

Suppose that link uw fails before u receives 10 from around the self-loop. Node u elects the candidate 5 via v and advertises this width around the self-loop. Figure 3.2(a) shows the current state of the protocol at u. Widths 10 and 5 are both in transit across the self-loop such that 10 precedes 5. With the reception of 10, node u elects the width 10 just learned from around the self-loop and advertises this width back around the self-loop. Figure 3.2(b) shows the new state of the protocol at u. The same two widths are in transit across the self-loop, but now 5 precedes 10. With the reception of 5, node u re-elects the candidate 5 via v and advertises this width around the self-loop. Thus, the protocol returns to the state shown in Figure 3.2(a). The oscillation between these two states continues indefinitely.



(a) State after the reception of 5 at u. **(b)** State after the reception of 10 at u.

Figure 3.2: Oscillatory behaviour with a non-restarting protocol for widest paths for destination x.

We proceed to examine optimality in stable state. We have the following definition.

Definition 3.4. Given a network, a non-restarting protocol is *optimal in stable state* in the network if, for all source-destination pairs (s, t):

(a) the elected attribute at s to reach t in stable state, which we denote by $e(s, t)$, is the i-optimal attribute from s to t, $e(s, t) = a_i^*(s, t)$;

(b) data-packets sourced at s and destined for t are guided along simple paths from s to t. [3]

The key property for the optimality in stable state of a non-restarting protocol is isotonicity. However, this property by itself does not ensure its optimality. In addition, all circuits in the network must be strictly inflationary. The strict inflation of all circuits implies that the stable state delivers data-packets from sources to destinations. With isotonicity, it does so along i-optimal paths; without isotonicity, these paths are not i-optimal, in general.

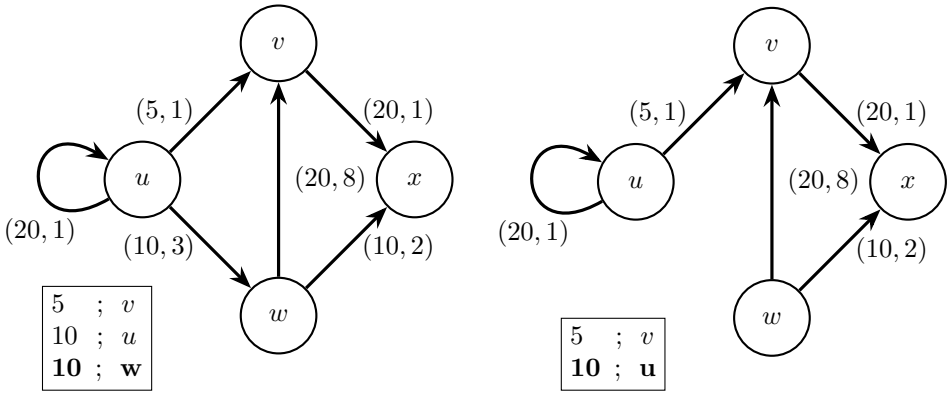
Theorem 3.2 ([11]). *If all circuits are strictly inflationary in a given network and isotonicity holds, then a standard non-restarting vectoring protocol is optimal in stable state in the network.*

[3] A *simple path* in a network is a path with all nodes distinct.

We have the following example.

Example 3.3. In Figure 3.3, we show that, when there is a circuit in a given network that is not strictly inflationary, the protocol may not route data-packets on loop-free paths. The goal is again to route data-packets on widest paths to destination x. Figure 3.3(a) show the stable state of the protocol at u. Node u elects 10 via w. Plus, it has a candidate 5 via v and a candidate 10 via around the self-loop.

With the failure of link uw fails, node u elects the candidate 10 via around the self-loop. Since its elected attribute has not changed, the protocol terminates at once. Figure 3.3(b) show the new stable state of the protocol at u. Node u forwards data-packets back to itself around the self-loop. Data-packets sourced at u and destined for x are trapped in a forwarding-loop around the self-loop and, therefore, they are not delivered to x.



(a) Stable state. **(b)** New stable state after the failure of uw.

Figure 3.3: No loop-freedom with a non-restarting protocol for destination x.

We discuss the problem solved by non-restarting protocols. With a non-restarting protocol, destination t elects attribute ϵ at all times. In stable state, the elected attribute at node u is the extension of the elected attribute at one of its out-neighbors. At length, the last attribute an out-neighbor v of u advertises to all its in-neighbors is its elected attribute in stable state, $e(v,t)$. Because advertisements are delivered in first in, first out order, the last attribute u receives from v is $e(v,t)$, which it extends by the attribute of its link to v to its candidate attribute via v, $a(uv) \oplus e(v,t)$. Since u elects the i-optimal attribute in the set of all its candidate attributes, its elected attribute in stable state satisfies the following fixed-point equation:

$$e(u,t) = O_i(\{a(uv) \oplus e(v,t) \mid v \text{ out-neighbor of } u\}). \tag{3.13}$$

A non-restarting protocol is, thus, a distributed algorithm to solve the fixed-point equation at every node. Conversely, the fixed-point equation at every node expresses the problem of routing data-packets on i-optimal paths constrained by destination-based forwarding [20]. This is a well-defined problem in-

dependently of routing protocols, which can be solved by a generalization of Dijkstra's algorithm that operates on a total order on attributes [46]. If, and only if, isotonicity holds, the problem of optimal path routing constrained by destination-based forwarding corresponds to that of optimal path routing [9, 13].

At last, we inspect the relationship between isotonicity and the time it takes for non-restarting protocols to converge following the network-wide announcement of a destination. When isotonicity is satisfied, a non-restarting protocol converges fast with the network-wide announcement of a destination. This is because the elected attributes improves every time a node sends an advertisement. In detail, suppose that node v elects and advertises to its in-neighbor u a new attribute b' that is i-preferred to its old elected attribute b, $b' \prec_i b$. Accordingly, node u replaces its candidate attribute $a(uv) \oplus b$ via v with $a(uv) \oplus b'$. Isotonicity and $b' \prec_i b'$ implies $a(uv) \oplus b' \preceq_i a(uv) \oplus b$. If u elects a new attribute, then this must be $a(uv) \oplus b'$, since no other candidate attribute has changed, and we have $a(uv) \oplus b' \prec_i a(uv) \oplus b$.

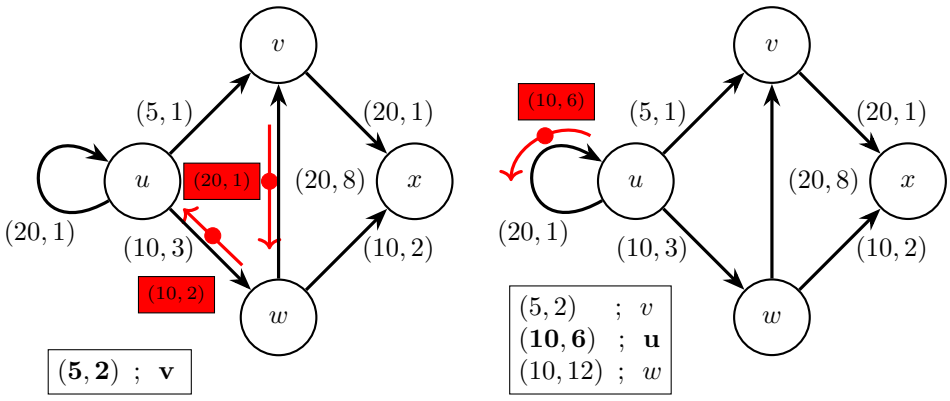
Example 3.4. In Figure 3.4, we illustrate that, when isotonicity is not satisfied, a non-restarting protocol may converge slowly with the network-wide announcement of a destination. The goal is to route data-packets on shortest-widest paths to destination x. We recall that the shortest-widest order does not met isotonicity. Initially, destination x advertises $(\infty, 0)$ to v and w. Node v elects $(20, 1)$ learned from x and advertises this width-length to u and w.

Suppose that w receives $(\infty, 0)$ from x before it receives $(20, 1)$ from v. First, node w elects $(10, 2)$ and advertises this width-length to u. Figure 3.4(a) shows the current state of the protocol. Node u elects $(10, 5)$ and advertises this width-length around the self-loop. Then, node w improves its elected width-length from $(10, 2)$ to $(20, 9)$ and advertises this width-length to u. Node u worsens its candidate width-length via w from $(10, 5)$ to $(10, 12)$. However, it does not elect this width-length, but the candidate width-length $(10, 6)$ via the self-loop. Figure 3.4(b) shows the new state of the protocol. Thereafter, node u repeatedly elects the width-length learned from around the self-loop and advertises this width-length back around the self-loop. The protocol terminates when the length of candidate via the self-loop becomes greater than 12, at which point u elects its candidate width-length via w.

In the absence of isotonicity, the improvement of the elected attribute at a node may imply the worsen of the elected attribute at some of its in-neighbors. As described in Example 3.1, the process whereby a non-restarting protocol has some nodes settle on worse attributes than those they started out with may take a long time, leaving data-packets trapped in forwarding-loops in the meantime.

3.2.2 Restarting Protocol

With a *standard restarting vectoring protocol*, a destination regularly initiates fresh computations instances, such that newer instances supersede older ones. A node holds only the elected attribute to reach the destination, which is the best attribute learned so far in the current computation instance; it



(a) State after the reception of $(\infty, 0)$ at w. **(b)** State after the reception of $(20, 9)$ at u.

Figure 3.4: Slow convergence with a non-restarting vectoring protocol for shortest-widest paths for destination x.

does not hold a candidate attribute to reach the destination via each of its out-neighbors. Within the same computation instance, the elected attribute can only improve. The election of a new attribute that is worse than the old one entails that the node is *black-holed* for the destination, which means it drops every data-packet intended for the destination. Furthermore, the node propagates the information that it cannot reach the destination upstream so that other nodes whose path to the destination includes this node are also black-holed. Nodes resume forwarding upon learning an attribute from a newer computation instance, which the destination initiates periodically. This action prevents processes akin to count-to-infinity and data-packets from being trapped in forwarding-loops. However, it entails that isotonicity is required for the basic delivery of data-packets [20]. Restarting protocols are traditional from wireless networks, with DSDV [17] and Babel [18] being prototypical examples.

3.2.2.A Distributed Path Computation

We explain how the distributed path computation performed by restarting protocols differs from that performed by non-restarting protocols.

In the canonical standard restarting vectoring protocol, a destination t initiates a fresh computation instance by advertising attribute ϵ to all its in-neighbors. Algorithm 2 presents the pseudo-code of the protocol for when node u receives the advertisement of pair (b, n) from out-neighbor v concerning destination t, $u \neq t$. Pair (b, n) comprises attribute b and the sequence number n of the instance that produces this attribute. Variable $opt_u[t]$ holds the elected attribute at u to reach t and variable $Seq_u[t]$ holds the sequence number of the computation instance that produces $opt_u[t]$.

Upon receiving (b, n) from v, node u first calculates the extension of the attribute of its link to v with b, $a(uv) \oplus b$ (line 1). If b is produced by a computation instance of higher sequence number than the

current elected attribute, then u replaces this attribute with $a(uv) \oplus b$ and the current sequence number with n (lines 2-4). Otherwise, in case b and the current elected attribute are produced by a computation instance sharing the same sequence number, the current elected attribute is updated to the most i-preferred between itself and $a(uv) \oplus b$ (lines 5-6). If there has been a change in the elected attribute or in the sequence number, then u advertises both to all its in-neighbors (lines 7-9).

Algorithm 2 Canonical standard restarting vectoring protocol. When node u receives pair (b, n) of attribute b and sequence number n from out-neighbor v concerning a destination t, $u \neq t$.

1: $a \coloneqq a(uv) \oplus b$
2: **if** $Seq_u[t] < n$ **then**
3: $\quad opt_u[t] \coloneqq a$
4: $\quad Seq_u[t] \coloneqq n$
5: **else if** $Seq_u[t] = n$ **then**
6: $\quad opt_u[t] \coloneqq O_i(\{opt_u[t], a\})$
7: **if** $opt_u[t]$ has changed or $Seq_u[t]$ has changed **then**
8: $\quad$ **for** all r in-neighbor of u **do**
9: $\quad\quad$ Send $(opt_u[t], Seq_u[t])$ to r.

The pseudo-code of Algorithm 2 presumes isotonicity in that a node advertises to all its in-neighbors the new elected attribute, but does not explicitly withdraw the old one. This fact merits an explanation. Consider that node v elects attribute b and that its in-neighbor u elects attribute $a(uv) \oplus b$ learned from v. Suppose that v presently elects an attribute b' that i-preferred to b, $b' \prec_i b$. Accordingly, node u learns attribute $a(uv) \oplus b'$ from v and considers it for election. At the same time, node u no longer considers $a(uv) \oplus b$, because v does not elect b. From isotonicity and $b' \prec_i b$, we write $a(uv) \oplus b' \preceq_i a(uv) \oplus b$. Hence, if $a(uv) \oplus b' \neq a(uv) \oplus b$, then $a(uv) \oplus b' \prec_i a(uv) \oplus b$, which translates the new elected attribute is i-preferred to the old one and, thus, the explicit withdraw of b is not required.

With the failure of link uv, for all destinations t, $u \neq t$, if v is the out-neighbor from which the elected attribute at u to reach t was learned, then u is black-holed for t and it advertises attribute $\bullet$ to all its in-neighbors. Subsequently, when a node receives $\bullet$ from the out-neighbor from which the elected attribute was learned, the node is itself black-holed for t and it advertises $\bullet$ to all its in-neighbors.

As in a non-restarting protocol, each node maintains a forwarding table with entries of the form:

destination **;** *next.hop*.

These entries have the same meaning and usage as before. A version of the protocol with a single entry per destination, whose value of *next.hop* corresponds to the out-neighbor from which the elected attribute was learned for the first time, prevents data-packets from traveling along multiple i-optimal

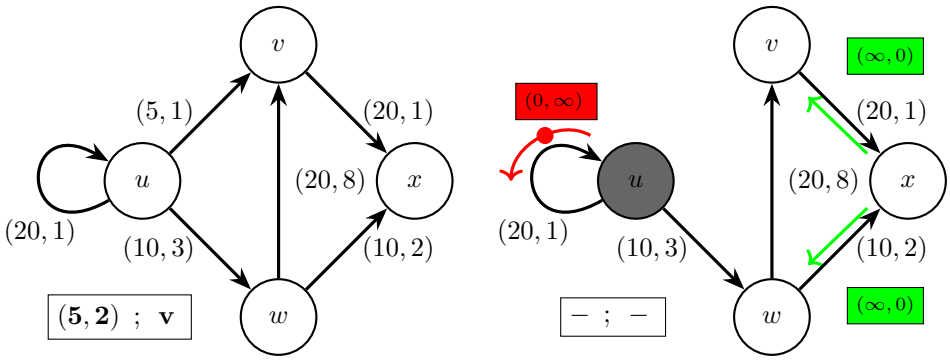
paths, but, on the other hand, it ensures that the algebraic conditions for the termination and optimality in stable state are weaker in comparison to non-restarting protocols.

We have the following example.

Example 3.5. In Figure 3.5, we exemplify the operation of a restarting protocol. The goal is to route data-packets on widest-shortest paths to destination x. Destination x initiates a computation instance by advertising $(\infty, 0)$ to v and w. Node v elects $(20, 1)$ learned from x and advertises this width-length to u and w. Node w learns $(20, 9)$ from v and $(10, 2)$ from x. It elects $(10, 2)$ and advertises this width-length to u, while discarding $(20, 9)$. Node u learns $(5, 2)$ from v and $(10, 5)$ from w. It elects $(5, 2)$ and advertises this width-length around the self-loop, while discarding $(10, 5)$. At last, node u learns $(5, 3)$ from around the self-loop and discard this width-length. Figure 3.5(a) shows the stable state of the protocol at u.

With the failure of link uv, node u is black-holed for x and it advertises $(0, \infty)$ around the self-loop. Subsequently, destination x initiates a fresh computation instance. Figure 3.5(b) shows the current state of the protocol. Width-lengths of the new computation instance replaces those of the older one. In the new computation instance, the elected width-length at u in stable state is $(10, 5)$.

The execution of a computation instance initiated by a destination resembles the network-wide announcement of the destination in a non-restarting protocol. With the failure of a link, the restarting protocol trades the possibility of some nodes being involved in processes comparable to count-to-infinity, during which data packets may be trapped in forwarding-loops, with that of these nodes being black-holed for the destination until they learn an attribute from a newer computation instance.



(a) Stable state

(b) State after the failure of link uv.

Figure 3.5: Example operation of a restarting protocol for widest-shortest paths to destination x. Black-holed nodes are in gray. Advertisements of the new computation instance are in green.

3.2.2.B Termination and Optimality

We investigate the differences between the algebraic conditions that guarantee the termination and optimality in stable state of non-restarting and restarting protocols.

The concept of termination, as described in Definition 3.3, applies here to a single computation instance, which is initiated by a destination when it advertises attribute ϵ to all its in-neighbors. This concept is operationally relevant when the period with which the destination initiates fresh computation instances is large compared to the time it takes for them to terminate.

Contrary to a non-restarting protocol, in a restarting protocol, inflation by itself implies the termination of the protocol to a stable state. Strict inflation of circuits and finiteness of path attributes is not necessary for termination and optimality in stable state; on the other hand, if isotonicity does not hold, then the stable state does not deliver data-packets from sources to destinations, in general. If isotonicity holds, data-packets are delivered from sources to destinations and travel along i-optimal paths.

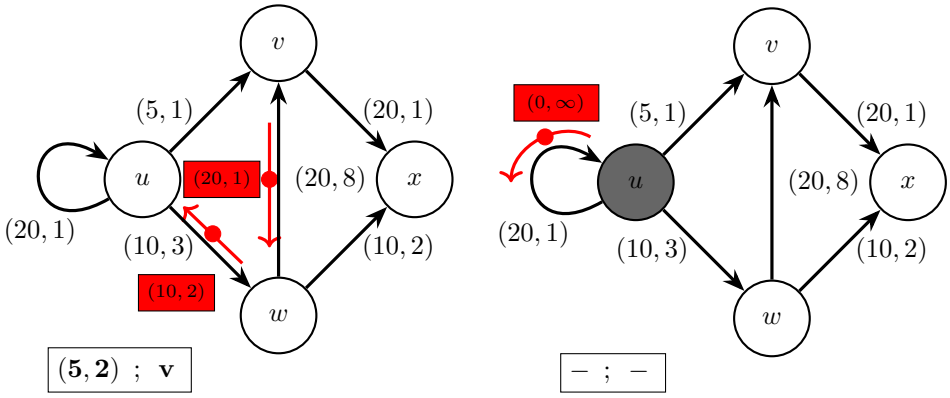
We have the following example.

Example 3.6. In Figure 3.6, we show that, in the absence of isotonicity, some nodes may be permanently black-holed. The goal is now to route data-packets on shortest-widest paths to destination x. Destination x initiates a computation instance by advertising $(\infty, 0)$ to v and w. Node v elects $(20, 1)$ learned from x and advertises this width-length to u and w.

Suppose that w receives $(\infty, 0)$ from x before it receives $(20, 1)$ from v. This way, node w elects $(10, 2)$ and, accordingly, node u elects $(10, 6)$. Subsequently, node w improves its elected width-length from $(10, 2)$ to $(20, 9)$. As a result, node u considers $(10, 12)$ for election, while not considering $(10, 6)$, because v no longer elects $(10, 2)$. Therefore, the elected width-length at u worsens and u is black-holed for x. If the advertisements are received in the same order in subsequent computation instances, then u can not escape its black-hole condition.

In the case isotonicity is not met, in a restarting protocol, some nodes may be left permanently black-holed during a period of convergence and, therefore, the stable state does not deliver data-packets from sources to destinations, in general. Contrastingly, in a non-restarting protocol, some nodes may be involved in a process similar to count-to-infinity during a period of convergence, but the stable state delivered data-packets from sources to destinations, although not necessarily on optimal paths.

To summarize, we illustrated the differences between non-restarting and restarting protocols in terms of the interaction between the algebraic properties and the behaviors of the protocols. To end to section, Table 3.2 compares the stable state and transient behaviour of non-restarting and restarting protocols and the algebraic requirements that imply their termination and optimality.



(a) State after the election of $(\infty, 0)$ at w. **(b)** State after the reception of $(20, 9)$ at u.

Figure 3.6: Permanent black-holes with a restarting protocol for shortest-widest paths to destination x.

Table 3.2: Comparison between the stable state and the transient behaviour of non-restarting and restarting protocols and the algebraic requirements for their termination and optimality.

	Non-restarting vectoring protocol	**Restarting vectoring protocol**
Stable state	With or without isotonicity, data-packets are delivered from sources to destinations.	Without isotonicity, data-packets are not delivered from sources to destinations, in general.
Transient behaviour	Nodes may be involved in processes comparable to count-to-infinity, during which data-packets may be trapped in forwarding loops.	Nodes are not involved in processes comparable to count-to-infinity, but may be black holed for destinations.
Termination	• Finiteness of the set of all path attributes. • Strictly inflation of all circuits.	(Inflation alone ensures termination.)
Optimality	• Strictly inflation of all circuits. • Isotonicity.	• Isotonicity.

4

A New Algebraic Theory for Routing on Multiple Optimality Criteria

Contents

We develop a new algebraic theory that promotes a general solution to routing on multiple optimality criteria by introducing new routing concepts, constructions and protocols. The present chapter is organized as follows. Section 4.1 presents an algebraic statement for routing on multiple optimality criteria and establishes the concepts and constructions that promote a solution to this problem. Section 4.2 designs the protocols that support routing on multiple optimality criteria.

4.1 Multiple Optimality and Dominance

The problem of routing on multiple optimality criteria can be formulated within the algebraic framework described in Section 3.1. Let S be a set of attributes that represents arbitrary performance metrics and $\oplus$ be a binary extension operation on attributes that calculates path attributes from link attributes. There is a *collection* I of arbitrary optimality criteria such that total order $\preceq_i$ defines the optimality criterion i, for all $i \in I$. We assume that $\oplus$ is inflationary for $\preceq_i$, for all $i \in I$. However, we do not assume $\oplus$ is isotone for $\preceq_i$, for some $i \in I$. [1] The goal is to allow every flow of data-packets to be routed from source to destination on an i-optimal path according to the optimality criterion $i \in I$ that is most appropriate for that particular flow. The problem of routing on optimal paths is a special case of the previous problem, where the collection comprises a single optimality criterion.

We propose a solution to the problem of routing on multiple optimality criteria, which entails a solution to routing on optimal paths. This solution is premised on the following ideas:

(a) Generalizing the algebraic framework described in Section 3.1 to accept partial orders on attributes and by introducing techniques that reduce total orders to partial orders;

(b) Defining the problem of routing on dominant paths and transforming the problem of routing on multiple optimality criteria into that of routing on dominant paths for partial orders that satisfy isotonicity;

(c) Constructing routing protocols that are able to route on dominant paths for such partial orders and, therefore, on optimal paths concurrently for multiple optimality criteria.

We detail the first two points in this section and last one in upcoming Section 4.2.

Partial orders and dominant paths. We introduce the notions of partial order and dominant path, which generalize those of total order and optimal path, respectively. A partial order $\preceq$ on attributes is a binary relation over S that satisfies antisymmetry, transitivity and *reflexivity*:

[1] Inflation is often met by concrete performance metrics, whereas isotonicity is not.

$$a \preceq_i b \text{ and } b \preceq_i a \text{ imply } a = b, \text{ for all attributes } a, b; \quad \text{(antisymmetry)} \quad (4.1)$$

$$a \preceq_i b \text{ and } b \preceq_i c \text{ imply } a \preceq_i c, \text{ for all attributes } a, b, c; \quad \text{(transitivity)} \quad (4.2)$$

$$a \preceq a, \text{ for all attributes } a. \quad \text{(reflexivity)} \quad (4.3)$$

Connexity implies reflexivity, so that a total order on attributes is a particular case of a partial order.[2] We still write $a \prec b$ for $a \preceq b$ and $a \neq b$. When $a \prec b$, we still say a is preferred to b and b is less preferred than a. If $a \preceq b$ or $b \preceq a$, then a and b are comparable; otherwise, they are incomparable. A partial order $\preceq'$ on attributes is *contained* in a partial order $\preceq$, and $\preceq$ *contains* $\preceq'$, if they are different and all pairs of attributes that are comparable in $\preceq'$ are also comparable in $\preceq$:

$$a \preceq' b \text{ implies } a \preceq b, \text{ for all attributes } a, b. \quad (4.4)$$

The *set of dominant attributes* in a set A of attributes, which we denote by $D(A)$, consists of the attributes of A with no attribute of A preferred to them:

$$D(A) = \{a \in A \mid \text{there is no } b \in A \text{ such that } b \prec a\}. \quad (4.5)$$

Given a network, the *set of dominant attributes* from source s to destination t, which we denote by $A^*(s, t)$, is the set of dominant attributes in the set of all path attributes from the source to the destination. A path from s to t is *dominant* if its attributes belongs to $A^*(s, t)$.

Isotonic reductions. We present the concept of isotonic reduction. A fundamental idea of our approach is the possibility to reduce a partial order on attributes that does not satisfy isotonicity to one that does satisfy this property. Attaining isotonicity entails letting some pairs of attributes that were comparable in the original partial order become incomparable in the reduced one, namely those whose relative preference is inverted when extended with a third attribute. This idea is expressed precisely by the following definition [21].

Definition 4.1. An *isotonic reduction* of a partial order $\preceq$ on attributes for a binary extension operation $\oplus$ is a partial order that is contained in $\preceq$ and for which $\oplus$ is isotone.

The more the pairs of attributes that are comparable, the less the routing state required by the routing protocols devised in Section 4.2 and the more efficient these protocols are. Hence, we aspire to isotonic reductions with as many pairs of comparable attributes as possible. A *largest isotonic reduction* is the one that contains every other isotone reduction. Theorem 4.1 shows that every partial order on

[2] A total order on attributes is binary relation over S that verifies antisymmetry and transitivity, but connexity instead of reflexivity.

attributes has a largest isotonic reduction by explicitly characterizing this reduction. This fact relies on the associativity of the binary extension operation.

Theorem 4.1. *The largest isotonic reduction of a partial order $\preceq$ on attributes for a binary extension operation $\oplus$, which we denote by $\preceq_R$, consists of all pairs of comparable attributes whose relative preference is preserved when both are extended with any common third attribute:*

$$a \preceq_R b \text{ if } x \oplus a \preceq x \oplus b \text{ for all attributes } x, \text{ for all attributes } a, b. \tag{4.6}$$

Proof. [3] We show that the binary relation $\preceq_R$ on attributes defined by $a \preceq_R b$ if $x \oplus a \preceq x \oplus b$ for all $x \in S$ is the largest isotonic reduction of $\preceq$ for $\oplus$. The proof is divided in three parts.

(a) *Binary relation $\preceq_R$ on attributes is a partial order.* We prove that $\preceq_R$ verifies antisymmetry, transitivity and reflexivity.

- *Antisymmetry*. Suppose that $a \preceq_R b$ and $b \preceq_R a$. From $a \preceq_R b$, we write $a = \epsilon \oplus a \preceq \epsilon \oplus b = b$ and, likewise, from $b \preceq_R a$, we write $b \preceq a$. The antisymmetry of $\preceq$ together with $a \preceq b$ and $b \preceq a$ imply $a = b$.
- *Transitivity*. Suppose that $a \preceq_R b$ and that $b \preceq_R c$. The transitivity of $\preceq$ together with $x \oplus a \preceq x \oplus b$ and $x \oplus b \preceq x \oplus c$ imply $x \oplus a \preceq x \oplus c$, for all $x \in S$. We deduce that $a \preceq_R c$.
- *Reflexivity*. The reflexivity of $\preceq$ implies $x \oplus a \preceq x \oplus a$, for all $x \in S$. We infer that $a \preceq_R a$.

(b) *Binary extension operation $\oplus$ is isotone for $\preceq_R$.* Suppose that $a \preceq_R b$. To prove that $c \oplus a \preceq_R c \oplus b$, for all $c \in S$, we need to assert that $x' \oplus (c \oplus a) \preceq_R x' \oplus (c \oplus b)$, for all $x' \in S$. Inequality $x \oplus a \preceq x \oplus b$ becomes $(x' \oplus c) \oplus a \preceq (x' \oplus c) \oplus b$ by choosing $x = x' \oplus c$. Using associativity of $\oplus$, we write $(x' \oplus c) \oplus a = x' \oplus (c \oplus a)$ and $(x' \oplus c) \oplus b = x' \oplus (c \oplus b)$ to deduce that $x' \oplus (c \oplus a) \preceq x' \oplus (c \oplus b)$.

(c) *Partial order $\preceq_R$ is the largest isotonic reduction of $\preceq$.* From Parts (a) and (b), it follows that $\preceq_R$ is an isotonic reduction of $\preceq$ for $\oplus$. We are left with proving that it is the largest such reduction. By way of contradiction, suppose that $a \preceq_R^+ b$ is an isotonic reduction of $\preceq$ for $\oplus$ that contains $\preceq_R$. There is $a \preceq_R^+ b$ such that $a \preceq_R b$ does not hold. Inequality $a \preceq_R^+ b$ implies $x \oplus a \preceq x \oplus b$, for all $x \in S$. On the other hand, the fact that $a \preceq_R b$ does not hold implies there is $x \in S$ for which $x \oplus a \preceq x \oplus b$ does not hold: a contradiction has been arrived at. □

Inflation continues to be a desirable algebraic property, as it is related to the termination of the routing protocols presented in Section 4.2 to stable states that guide data-packets from sources to destinations. Theorem 4.2 shows that the largest isotonic reduction of an inflationary partial order is itself inflationary.

[3] The proofs of Theorems 4.1 and 4.2 are also given in Reference [47]. However, these are rather important results and, therefore, they are repeated here.

This result requires both associativity and commutativity of the binary extension operation. A corollary of this theorem is that the largest isotonic reduction of a partial order is non-trivial, in the sense that it includes other pairs of comparable attributes besides those consisting of an attribute and itself.

Theorem 4.2. *Let $\oplus$ be a binary extension operation on attributes and $\preceq$ be a partial order. If $\oplus$ is inflationary for $\preceq$, then $\oplus$ is also inflationary for the largest isotonic reduction of $\preceq$ for $\oplus$.*

Proof. Let $\preceq_R$ be the largest isotonic reduction of $\preceq$ for $\oplus$. To prove that $b \preceq_R a \oplus b$, we need to assert that $x \oplus b \preceq x \oplus (a \oplus b)$, for all $x \in S$. From inflation of $\oplus$ for $\preceq$, we obtain $x \oplus b \preceq a \oplus (x \oplus b)$, for all $x \in S$. Using associativity and commutativity of $\oplus$, we write

$$\begin{aligned} a \oplus (x \oplus b) &= (a \oplus x) \oplus b \\ &= (x \oplus a) \oplus b \\ &= x \oplus (a \oplus b) \end{aligned}$$

to conclude that $x \oplus b \preceq x \oplus (a \oplus b)$. □

From multiple optimality to dominance. We provide a procedure that starts with a collection of optimality criteria and ends with a partial order on attributes that is contained in each of the total orders of the collection and that satisfies isotonicity.

First, we reduce the total orders that define the optimality criteria in the collection I to a common partial order that is contained in each of them. Define the binary relation $\preceq_I$ on attributes as the *intersection* [14] of all total orders $\preceq_i$, $i \in I$:

$$a \preceq_I b \text{ if } a \preceq_i b \text{ for all } i \in I, \text{ for all attributes } a, b. \tag{4.7}$$

It is easy to show that the intersection of partial orders is a partial order. Therefore, binary relation $\preceq_I$ is a partial order on attributes. Two attributes are comparable in $\preceq_I$ if one is i-preferred to the other, for all $i \in I$. They are incomparable if one is i-preferred to the other but the other is j-preferred to the first, for some distinct $i, j \in I$. In practice, we expect the total orders that model the optimality criteria to share many pairs of comparable attributes in common so that their intersection has many pairs of comparable attributes. Because we assume $\oplus$ is inflationary for $\preceq_i$, for all $i \in I$, $\oplus$ is inflationary for $\preceq_I$: for all attributes a, b, inequality $a \preceq_i b \oplus a$, for all $i \in I$, implies $a \preceq_I b \oplus a$. Since we do not assume $\oplus$ is isotone for $\preceq_i$, for some $i \in I$, $\oplus$ is not isotone for $\preceq_I$, in general.

Second, we further reduce the common partial order to satisfy isotonicity. If $\oplus$ is not isotone for $\preceq_I$, then we retain only the largest isotonic reduction of $\preceq_I$, as detailed in Theorem 4.1, which we denote by $\preceq_{I,R}$. Theorem 4.2 guarantees that $\oplus$ is also inflationary for $\preceq_{I,R}$. In the special case the collection consist of a single optimality criterion, this procedure comes down to the isotonic reduction of the total

order that defines the optimality criterion. This way, we obtain a partial order that expresses a trade-off between the connexity of the total order and isotonicity.

Given a network, let $A^*_{I,R}(s,t)$ be the set of dominant attributes from source s to destination t according to $\preceq_{I,R}$. By construction, partial order $\preceq_{I,R}$ is contained in $\preceq_i$, for all $i \in I$. Therefore, the i-optimal attributes from s to t are found in the set of dominant attributes from s to t, for all $i \in I$:

$$a^*_i(s,t) \in A^*_{I,R}(s,t), \text{ for all } i \in I. \tag{4.8}$$

Thus, the problem of computing i-optimal attributes concurrently for all $i \in I$ has become that of computing sets of dominant attributes according to a partial order that is contained in $\preceq_i$, for all $i \in I$, and for which $\oplus$ is both inflationary and isotone.

The feasibility of the transformation described above depends on the sizes of the sets of dominant attributes from sources to destinations. In some routing contexts, the number of dominant attributes from sources to destinations is polynomially bounded on the size of the network. The evaluation presented in Chapter 5 shows that the number of dominant attributes in such contexts is substantially smaller than predicted by the worst case bounds.

Examples revisited. In summary, we reduced the problem of routing on multiple optimality criteria to that of routing on dominant paths for an isotone partial order. The linchpins of this transformation were the ideas of intersecting collections of total orders and reducing partial orders to satisfy isotonicity. To conclude this section, we obtain the largest isotonic reductions and intersections of several total orders on pairs width-length. (See Section 3.1.)

The product order on width-lengths, which we denote by $\preceq_{\mathsf{WL}}$, is defined by:

$$(w,l) \preceq_{\mathsf{WL}} (w',l') \text{ if } w \geq w' \text{ and } l \leq l', \text{ for all width-lengths } (w,l),(w',l').$$

The product order is a partial order on width-lengths for which isotonicity is satisfied. The product order on width-lengths was first introduced in Chapter 2. We have the following two propositions.

Proposition 4.1. *The largest isotonic reduction of the shortest-widest order is the product order on width-lengths.*

Proposition 4.2. *The largest isotonic reduction of the K-quickest order is the intersection between the K-quickest order and the widest-shortest order.*

The proofs of these two propositions can be found in Reference [47].

Figure 4.1 represents the shortest-widest order in the width-length plane. The shortest-widest order on width-lengths does not satisfy isotonicity and, therefore, a standard vectoring protocol is not able to route on shortest-widest paths. Figure 4.2 represents the product order in the width-length plane. From Proposition 4.1, we learn that a partial-order vectoring protocol designed in Section 4.2 that operates on the product order on width-lengths is able to route on shortest-widest paths. It is worth noting that whereas in Chapter 2 the product order was a working hypothesis, here was derived as the largest isotonic reduction of the shortest-widest order.

Figure 4.3 shows the K-quickest order in the width-length plane. The K-quickest order does not satisfy isotonicity, in general, and so routing on K-quickest paths is not possible with a vectoring protocol. Figure 4.4 shows the intersection between the K-quickest order and the widest-shortest order. Routing on K-quickest paths is now possible with a partial-order vectoring protocol, for some K. Notice that the largest isotonic reduction of the K-quickest order contains the product order on width-lengths. Larger isotonic reductions are preferred, as they contribute to the efficiency of partial-order vectoring protocols.

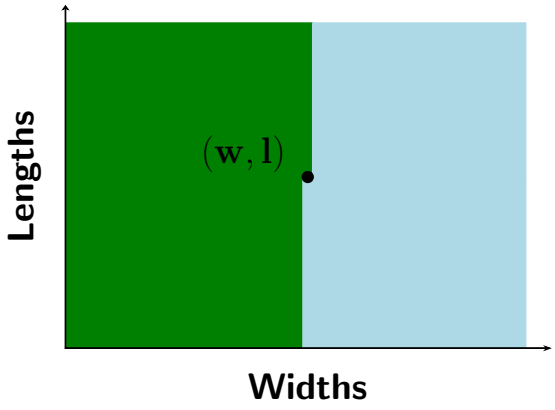


Figure 4.1: Shortest-widest order. The green area contains the width-lengths that are less preferred than $(\mathbf{w}, \mathbf{l})$ and the blue area contains those that are preferred to (w, l).

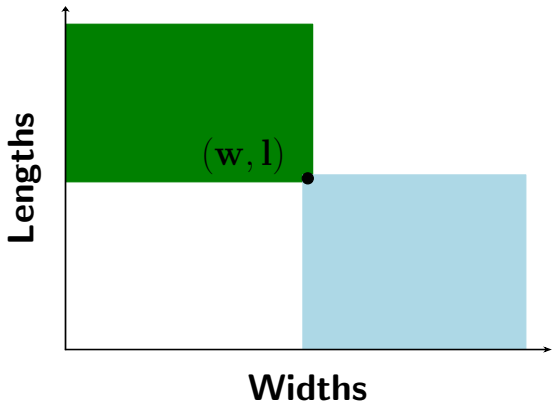


Figure 4.2: Product order on width-lengths. The white area contains the width-lengths that are incomparable with (w, l).

The following two propositions are easy and, thus, given without a proof.

Proposition 4.3. *The intersection of the K-quickest orders, for all $K \geq 0$, is the product order on width-lengths.*

Proposition 4.4. *The intersection of the W-quickest orders, for all $W \geq 0$, is the product order on width-lengths.*

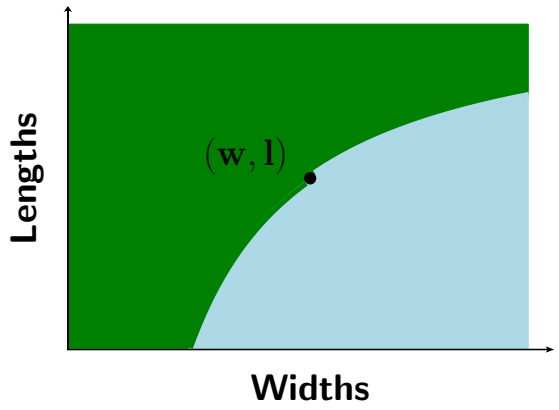


Figure 4.3: K-quickest order.

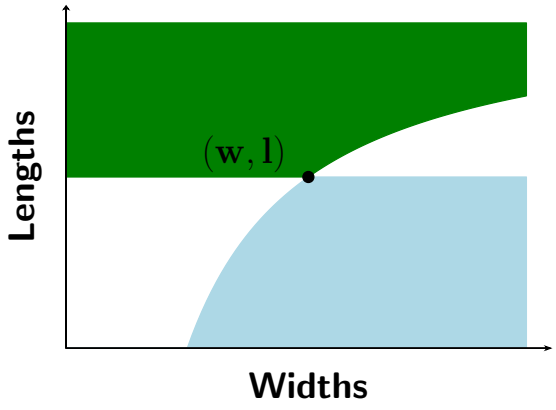


Figure 4.4: Intersection between the K-quickest order and the widest-shortest order.

From Propositions 4.3 and 4.4, we learn that a partial-order vectoring protocol that operates on the product order on width-lengths supports routing optimally and concurrently on K-quickest paths, for all K, and on W-wide-shortest paths, for all W.

4.2 Protocols for Dominant Paths

We design *partial-order vectoring protocols*, which are routing protocols that expand the standard vectoring protocols reviewed in Section 3.2 to operate on partial orders on attributes with the goal of routing on dominant paths. Partial-order vectoring protocols perform a distributed computation of path attributes at nodes to reach destinations separately per destination, as standard vectoring protocols do. However, they have each node elect and advertise to its in-neighbors the set of dominant attributes from among the attributes learned from its out-neighbors, instead of the optimal attribute. If isotonicity is satisfied, then these protocols are able to route on dominant paths, namely on optimal paths for any optimality criterion defined by a total order that contains the partial order.

We introduce two classes of partial-order vectoring protocols, which generalize, respectively, the classes of non-restarting and restarting vectoring protocols. In the remainder of this chapter, we formalize the operation of these classes of protocols and identify the algebraic requirements that guarantee that they terminate to stable states that route data-packets from sources to destinations on dominant paths. Section 4.2.1 presents the class of partial-order non-restarting vectoring protocols and Section 4.2.2 presents the class of partial-order restarting vectoring protocols.

4.2.1 Non-Restarting Protocol

As is the case with a standard non-restarting vectoring protocol, with a *partial-order non-restarting vectoring protocol*, a destination maintains a single computation instance throughout time. A node stores

a set of candidate attributes to reach the destination via each of its out-neighbors; at any given instant, it elects the dominant attributes from among all its candidate attributes to reach the destination. During a period of convergence, the set of elected attributes improves and worsens. Accordingly, nodes may be involved in processes akin to count-to-infinity, with data-packets trapped in forwarding-loops in the meantime. With or without isotonicity, the stable state of the protocol delivers data-packets from sources to destinations, albeit they do not travel along dominant paths, in general.

4.2.1.A Distributed Path Computation

We detail how partial-order non-restarting vectoring protocols perform the distributed path computation from sources to destinations and how data-packets are guided via label-switching along the paths computed.

In the canonical partial-order non-restarting vectoring protocol, destination t initiates the computation instance by advertising singleton $\{\epsilon\}$ to all its in-neighbors. Algorithm 3 presents the pseudo-code of the protocol for when node u receives the advertisement of set B of attributes from out-neighbor v concerning destination t, $u \neq t$. Variable $DomTab_u[t, v]$ stores the set of candidate attributes at u to reach t via v and variable $Dom_u[t]$ stores the set of elected attributes at u to reach t.

Upon receiving set B from out-neighbor v, node u updates its set of candidate attributes via v to the extension of the attribute of its link to v with every attribute of B (line 1). Then, it determines its own set of elected attributes as the set of dominant attributes in the set of all its candidate attributes (line 2). If the set of elected attributes has changed, then u advertises this set to all its in-neighbors (line 4-5).

Algorithm 3 Canonical partial-order non-restarting vectoring protocol. When node u receives the advertisement of set B of attributes from out-neighbor v concerning destination t, $u \neq t$.

1: $DomTab_u[t, v] \coloneqq \{\, a(uv) \oplus b \mid b \in B \,\}$
2: $Dom_u[t] \coloneqq D(\, \{\, DomTab_u[t, v], v \text{ out-neighbor of } u \,\}\,)$
3: **if** $Dom_u[t]$ has changed **then**
4: **for** all r in-neighbor of u **do**
5: Send $Dom_u[t]$ to r

Following the failure of link uv, for all destinations t, $u \neq t$, node u withdraws its set of candidate attribute via v, re-elects a new set of attributes from among its remaining candidate attributes and advertises this set of attributes to all its in-neighbors in case it has changed.

With multiple elected attributes at each node per destination, data-packets can no longer be forwarded exclusively based on their destinations. Each node assigns a different label to each elected attribute to reach the destination, which is advertised alongside the respective attribute to in-neighbors [22].

Therefore, each node maintains two tables:

- A *path selection* table with entries of the form:

  $$\textit{destination} \ ; \ \textit{attribute} \ ; \ \textit{label},$$

  where *destination* is a destination, *attribute* is an elected attribute to reach the destination and *label* is the label assigned to *attribute*.

- A *forwarding* table with entries of the form:

  $$\textit{destination} \ ; \ \textit{label} \ ; \ \textit{next.hop} \ , \ \textit{next.label},$$

  where *next-hop* is the out-neighbor from which *attribute* was learned and *next.label* is the label advertised alongside the attribute that originated *attribute*.

Data-packets sourced at the node and meant to travel along a path with attribute *attribute* are first sent to the path selection table and then sent to the forwarding table. In the path selection table, data-packets are marked with label *label*. In the forwarding table, data-packets have label *label* replaced with label *next.label* and are forwarded to out-neighbor *next.hop*. Data-packets arriving at the node from an in-neighbor carrying label *label* are sent directly to the forwarding table.

A node may install multiple entries in the path selection table with the same value of *destination* and *attribute*, but different values of *label*. This action enables data-packets to travel from sources to destinations along multiple dominant paths with a common attribute.

4.2.1.B Termination and Dominance

We state the algebraic conditions that guarantee the termination and dominance in stable state of partial-order non-restarting vectoring protocols.

We start with termination. As described in Definition 3.3, a stable state of a partial-order non-restarting vectoring protocol is a state without advertisements in transit across links of the network. A partial-order non-restarting vectoring protocol terminates if, in the absence of network changes, it reaches a stable state from any initial state.

As in a standard non-restarting vectoring protocol, the key property for the termination of a partial-order non-restarting vectoring protocol is inflation. In addition to inflation, a partial-order non-restarting vectoring protocol requires that the network on which the protocol is run satisfies two algebraic properties. First, the set of all path attributes in the network must be finite, to avoid count-to-infinity. (See

Example 3.1.) Second, all circuits in the network must be strictly inflationary, to prevent oscillatory behaviors. (See Example 3.2.) We have the following theorem.

Theorem 4.3. *If the set of all path attributes is finite and all circuits are strictly inflationary in a given network, then a partial-order non-restarting vectoring protocol terminates in the network.*

The proof can be found in Reference [47].

We go on to discuss dominance. The property of dominance generalizes that of optimality, as detailed in Definition 3.4. We have the following definition.

Definition 4.2. Given a network, a partial-order non-restarting protocol is *dominant in stable state* in the network if, for all source-destination pairs (s, t):

(a) The set of elected attributes at s to reach t in stable state, which we denote by $E(s, t)$, is the set of dominant attributes from s to t, $E(s, t) = A^*(s, t)$;

(b) Data-packets sourced at s, destined for t and meant to travel along a path with attribute e are guided along simple paths from s to t, for all $e \in E(s, t)$.

In the same way as with a standard non-restarting vectoring protocol, the key property for the dominance of a partial-order non-restarting vectoring is isotonicity. Besides isotonicity, a partial-order non-restarting vectoring protocol demands that all circuits of the network are strictly inflationary. The strict inflation of circuits implies that, whether or not isotonicity holds, the stable state of a partial-order non-restarting protocol delivers data-packets from sources to destinations via label-switching on paths whose attributes are those elected at the nodes. If isotonicity holds, then these attributes are dominant. We have the following theorem.

Theorem 4.4. *If all circuits are strictly inflationary in a given network and isotonicity holds, then the partial-order non-restarting vectoring protocol is dominant in stable state in the network.*

The proof can also be found in Reference [47].

4.2.2 Restarting Protocol

As is the case with a standard restarting vectoring protocol, with a *partial-order restarting vectoring protocol*, a destination regularly initiates fresh computation instances during which newly elected attributes replace those from older instances. Each node does not hold a set of candidate attributes to reach the destination via each of its out-neighbors; rather, it holds exclusively a set of elected attributes to reach the destination, which consists of the best attributes learned so far in the current computation

instance. During a period of convergence, the elected attributes can only improve. This action prevents nodes from being involved in processes similar to count-to-infinity and data-packets from being trapped in forwarding-loops. However, it entails that, without isotonicity, the stable state of the protocol does not deliver data-packets from sources to destinations, in general, as some nodes may be left permanently black-holed.

4.2.2.A Distributed Path Computation

We clarify how the distributed path computation performed by partial-order restarting vectoring protocols is distinct from that performed by partial-order non-restarting vectoring protocols.

In the canonical partial-order restarting vectoring protocol, a destination t initiates a fresh computation instance by advertising attribute ϵ to all its in-neighbors. Algorithm 4 presents the pseudo-code of the protocol for when node u receives the advertisement of pair (b, n) from out-neighbor v concerning destination t, $u \neq t$. Again, pair (b, n) comprises attribute b and the sequence number n of the instance that produces this attribute. Variable $Dom_u[t]$ stores the set of elected attributes at u to reach t and variable $Seq_u[t]$ stores the sequence number of the computation instance that produces $Dom_u[t]$.

Upon receiving (b, n) from v, node u first calculates the extension of the attribute of its link to v with b, $a(uv) \oplus b$ (line 1). If b is from a more recent computation instance than the current set of elected attributes at u, then this set is replaced with the singleton consisting of $a(uv) \oplus b$ and the current sequence number at u is substituted by n (line 2-4). Otherwise, in case $a(uv) \oplus b$ is from the same computation instance as the current set of elected attributes at u, this set is updated to account for $a(uv) \oplus b$, that is, if $a(uv) \oplus b$ is not less preferred than any attribute of the set, then $a(uv) \oplus b$ is included in the set and all attributes less preferred than it are withdrawn (line 5-6). If there has been a change in the set of elected attributes or in the sequence number, then u send both to all its in-neighbors (lines 7-9).

Algorithm 4 Canonical partial-order restarting vectoring protocol. When node u receives pair (b, n) of attribute b and sequence number n from out-neighbor v concerning a destination t, $u \neq t$.

```
1: a ≔ a(uv) ⊕ b
2: if Seq_u[t] < n then
3:     Dom_u[t] ≔ a
4:     Seq_u[t] ≔ n
5: else if Seq_u[t] = n then
6:     Dom_u[t] ≔ D( Dom_u[t] ∪ {a} )
7: if Dom_u[t] has changed or Seq_u[t] has changed then
8:     for all r in-neighbor of u do
9:         Send (a, Seq_u[t]) to r
```

The presence of isotonicity entails that, in the pseudo-code of Algorithm 4, a node advertises to all its in-neighbors the new elected attributes, but does not explicitly withdraw any of the old ones. This fact was justified in Section 3.2.2 and implies that the sets of elected attributes improve every time a node advertises a new elected attribute to its in-neighbors.

The differences between Algorithm 4 and Algorithm 3 are worth noting. Algorithm 4 only holds variable $Dom_u[t]$, whereas Algorithm 3, in addition to $Dom_u[t]$, holds variables $DomTab_u[t, v]$, for all out-neighbors v of u. Algorithm 4 first adds $a(uv) \oplus b$ to $Dom_u[t]$ and then elects the dominant attributes in $Dom_u[t]$, whereas Algorithm 3 replaces what was stored in $DomTab_u[t, v]$ with $\{a(uv) \oplus b \mid b \in B\}$ and elects the dominant attributes from among $DomTab_u[t, v]$, for all out-neighbors v of u. Algorithm 4 advertises a single attribute to in-neighbors of the node, whereas Algorithm 3 advertises sets of attributes.

As in a partial-order non-restarting vectoring protocol, nodes assign distinct labels to elected attributes, with labels and respective attributes being advertised together to in-neighbors. Nodes maintain a path selection table with entries of the form:

destination **;** *attribute* **;** *label*,

and a *forwarding* table with entries of the form:

destination **;** *label* **;** *next.hop* **,** *next.label*.

These entries have the same meaning and usage as before.

4.2.2.B Termination and Dominance

We identify the algebraic requirements that ensure the termination and dominance in stable state of partial-order restarting protocols.

As in a standard restarting vectoring protocol, the concept of termination applies to an isolated computation instance of a partial-order restarting vectoring protocol. If isotonicity holds, then a partial-order restarting vectoring protocol terminates in a stable state where nodes elect dominant attributes to reach destinations. Contrary to a partial-order non-restarting vectoring protocols, in a partial-order restarting vectoring protocol, strict inflation of circuits and finiteness of path attributes are not necessary for termination and dominance, but, on the other hand, isotonicity must hold even for basic delivery of data-packets. Without isotonicity, these protocols may leave some nodes permanently black holed. (See Example 3.6.)

5

Evaluation

Contents

Our solution to routing on multiple optimality criteria required the design of partial-order vectoring protocols that route on dominant paths for partial orders that satisfy isotonicity. We evaluate the practicality and efficiency of these protocols. We built a simulator of partial-order vectoring protocols to answer two questions:

(a) What are the sizes of sets of dominant attributes from sources to destinations?

(b) How do partial-order vectoring protocols behave during periods of convergence following a network event?

We based our evaluation on publicly available topologies and attributes.

The present chapter is structured in the following way. Section 5.1 presents the test networks and the simulator used in this evaluation. Section 5.2 studies the number of dominant attributes from sources to destinations. Sections 5.3 and 5.4 investigate the transient behaviour of partial-order non-restarting and restarting vectoring protocols, respectively.

5.1 Networks and Simulator

We describe the test networks and the simulator of partial-order vectoring protocols that support our evaluation. The networks used for testing are based on the topologies of Internet Service Providers (ISP) inferred by the Rocketfuel project [48]. [1] These topologies have each link annotated with an OSPF weight [2] and with the propagation delay across the link. There is a well-defined structure to the Rocketfuel topologies: nodes are clustered around geographical areas and nodes within each cluster are densely connected to every node in the cluster through links of unit delay. The test networks consist of the largest 2-edge-connected component [29] of each topology. We assign a width and a length to each link of a test network. The width of a link is set to the inverse of the OSPF weight. The length of a link is set to the propagation delay across the link. Table 5.1 characterizes the test networks by their number of nodes and links.

We constructed a simulator of partial-order vectoring protocols, both non-restarting and restarting versions. The source code of the simulator and the accompanying documentation are publicly available on GitHub [3]. The simulator supports four instantiations of attributes: pairs hops-length, width-hops and width-length, and triples width-hops-length. Widths extend with minimum: the width of a path is the minimum of the widths of its links. Hops and lengths extend with addition: the hops of a path in the number of links of the path and the length of a path is the sum of the lengths of its links. The product

[1] The networks of ISP form an Autonomous System (AS) and identified by their Autonomous System Number (ASN).

[2] The Open Shortest Path First (OSPF) routing protocol has each node maintain a representation of the network on which the protocol is run [49]. Each link in the network is assigned a length called an OSPF weight, which is typically set to the inverse of the capacity of the link. Each node runs Dijkstra's algorithm [29] to find the shortest paths to every other node.

[3] See https://github.com/miferrei/rmoc-sigcomm2020-artifact.

Table 5.1: ASN, number of nodes, number of links, number of distinct link widths and average number of dominant width-lengths for all test networks.

ASN	Nodes	Links	Distinct Widths	Dominant Width-Lengths
1221	50	194	8	1.9
1239	284	1882	19	2.5
1755	73	292	18	2.2
3257	113	558	21	3.5
3967	72	180	19	3.7
6461	129	726	19	2.8

orders on the pairs and on the triple were considered, as well as several total orders on width-lengths. (See Table 3.1.)

The simulator executes trials of the protocols that are triggered by network events, such as the network-wide announcement of a new destination or the failure of a link. Advertisements traverse links in first in, first out order and are subject to a random delay taken from a uniform distribution. We set the range of this distribution from 0.01 to 1 ms. With the purpose of avoiding count-to-infinity, advertisements of the non-restarting protocol that travel more than a prespecified maximum number of hops are invalidated. As in RIP, we set that maximum number to 15 (see Section 3.2.1).

5.2 Sets of Dominant Attributes

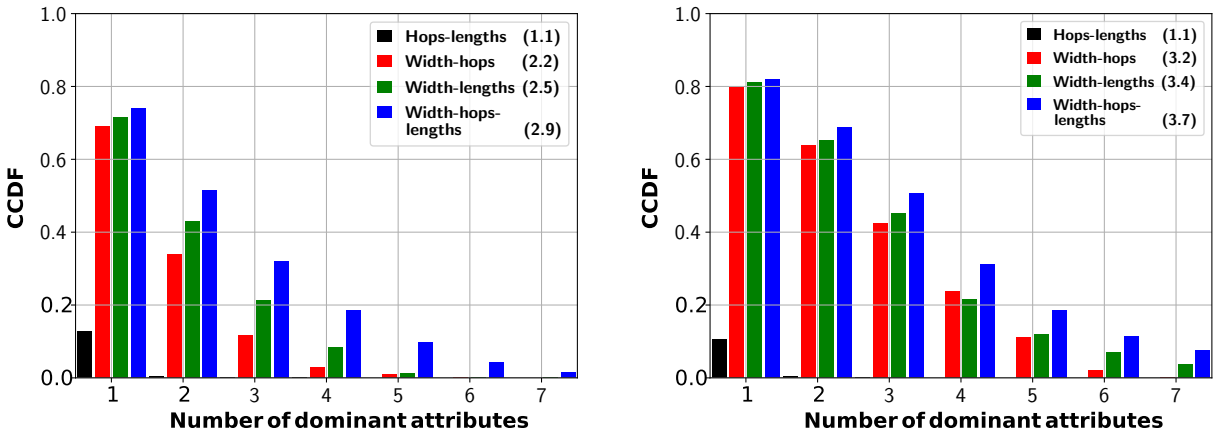
The amount of routing state maintained by partial-order vectoring protocols is proportional to the size of the sets of elected attributes at nodes to reach destinations. With isotonicity, the elected attributes at nodes to reach destinations in stable state are precisely the dominant attributes from nodes to destinations. (See Theorem 4.3.) Partial-order vectoring protocols are feasible to the extent that the sets of dominant attributes are small. Therefore, we assess the number of dominant attributes in the test networks for the product order on the different instantiations of attributes.

The sets of dominant attributes from sources to destinations in a network are read from the stable state of partial-order vectoring protocols. With the simulator, we execute a trial of the non-restarting protocol following the network-wide announcement of all possible destinations. (The restarting protocol could have been used to the same effect.) Alternatively, the sets of dominant attributes can be computed with a generalization of Dijkstra's algorithm that operates according to partial orders on attributes. This algorithm is presented in Appendix A and served to validate the simulator.

Table 5.1 shows the average number of dominant width-lengths over all source-destination pairs for all test networks. The average number of dominant width-lengths ranges from 1.9, in AS 1221, to 3.7, in AS 3967. The number of dominant width-lengths is upper bounded by the number of distinct path widths, which equals the number of distinct link widths, because the width of a path is the width of one of its links. The table also shows the number of distinct link widths for all test networks. The average number of dominant width-lengths is considerably lower than the number of distinct link widths. For example, in AS 1239, the average number of dominant width-lengths is 2.5, while there are 19 distinct link widths. Moreover, the average number of dominant hops-lengths is less than 1.2 for all test networks. (These numbers are not shown in Table 5.1.) Since the hops of every path are smaller than the number of nodes in the network, the size of the set of dominant hops-lengths is at most the number of nodes in the network. The average number of dominant hops-lengths is negligible in comparison to the number of nodes.

Figure 5.1(a) shows the Complementary Cumulative Distribution Function (CCDF) of the number of dominant attributes in the largest test network, AS 1239. The product order on the different instantiations of attributes is presented: (a) hops-lengths, in black; (b) width-hops, in red; (c) width-lengths, in green; and (d) width-hops-lengths, in blue. The average number of dominant hops-lengths is 1.1 and the maximum number of dominant hops-lengths is three. We deduce that path hops and path lengths are highly correlated, that is to say, a path with the minimum number of links from source to destination is very often a shortest path. The average number of dominant attributes for width-hops, width-lengths and width-hops-lengths is 2.2, 2.5 and 2.9, respectively. The percentages of source-destination pairs for which the number of dominant attributes is greater than three are 11.6%, 21.2% and 31.9% respectively for width-hops, width-lengths and width-hops-lengths. As opposed to path hops and path lengths, we infer that path widths and path hops are many times at odds with one another: a widest path from source to destination is usually not a path with the minimum number of links. Furthermore, there is more diversity in path lengths than in path hops, which justifies that there are more dominant width-lengths than dominant width-hops than dominant hops-lengths. Finally, it is easy to show that if a triple width-hops-length is dominant, then the pairs hops-length, width-hops, and width-length are dominant as well. Thus, sets of dominant width-hops-lengths are necessarily larger than the other sets of dominant attributes. Figure 5.1(b) shows the CCDF of the number of dominant attributes in AS 3967. The topologies of AS 1239 and AS 3967 share the same structure, as described in Section 5.1, and the bars of number of dominant attributes are similar. Therefore, the observations made for AS 1239 are also valid for AS 3967.

In summary, we studied the practicality of partial-order vectoring protocols by computing statistics of the number of dominant attributes. We expected that the size of the sets of dominant attributes was linear in the size of the test networks. However, we observed that the number of dominant attributes falls short of this upper bound and is rather small, thus suggesting the feasibility of the protocols proposed.



(a) AS 1239 **(b)** AS 3967

Figure 5.1: CCDF of the number of dominant attributes for the product order on hops-lengths, width-hops, width-lengths and width-hops-length. Averages are given inside parenthesis.

5.3 Transient Behaviour of Non-Restarting Protocols

We assess the transient behaviour of partial-order non-restarting vectoring protocols against two types of network events: the network-wide announcement of a destination and the failure of a link. The metric used is the *termination time*, defined as the time elapsed between the instant the network event occurs and the instant the protocol reaches a stable state. We intend to respond to two questions:

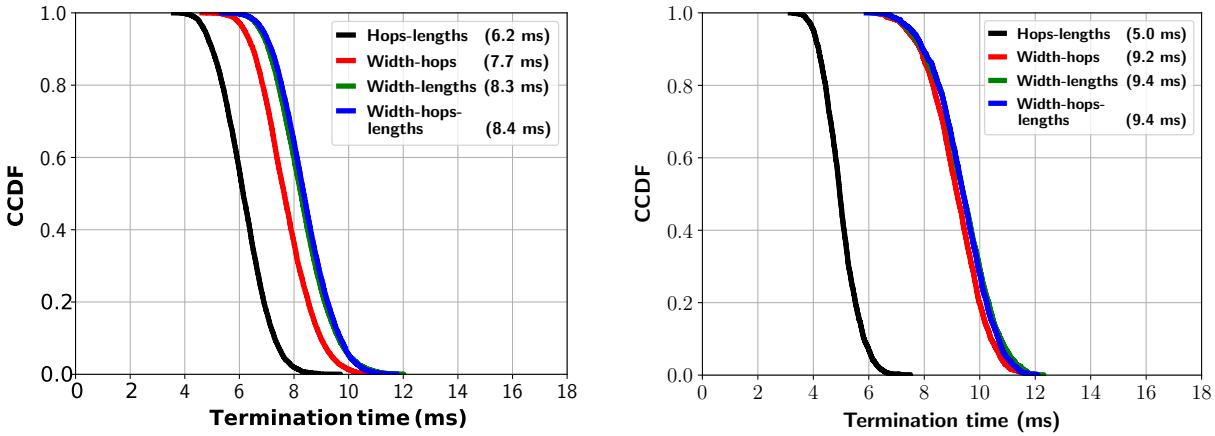
(a) What is the speed of convergence of partial-order non-restarting vectoring protocols?

(b) How does it compare to the speed of convergence of standard non-restarting vectoring protocols?

With the simulator, we run 25 independent trials following the network-wide announcement of a new destination, for each possible destination node. We run 25 independent trials following the failure of a link, for each possible link failures. We report our findings for AS 1239 and AS 3967. However, we carry out the discussion only for AS 1239. As detailed in Section 5.2, the observations made for AS 1239 are also valid for AS 3967.

Network-wide announcement of a destination. Figure 5.2(a) shows the CCDF of the termination times in AS 1239 following the network-wide announcement of a new destination, over all possible destination nodes and all 25 independent trials. The product order on the different instantiations of attributes is presented. The curves of termination times are rather smooth and steep. The average termination times are 6.2 ms, 7.7 ms, 8.3 ms and 8.4 ms for length-hops, width-hops, width-lengths and width-hops-lengths, respectively.

The product order on all four instantiations of attributes satisfies isotonicity. In a partial-order non-restarting vectoring protocol, nodes continually elect and advertise to its neighbors the best from among the attributes learned from its neighbors and isotonicity entails that an elected attribute can only be replaced by a better attribute during a trial. (See Section 3.2.1.) Therefore, the termination time is equal to the maximum delay to propagate an advertisement all the way up a dominant path and is roughly proportional to the number of links in a dominant path with the largest such number. As discussed in the previous section, path widths and path hops are often not correlated, which means that a widest path from source to destination typically traverses more than the minimum number of links required to reach the destination from the source. This fact explains why attributes involving width lead to longer termination times.

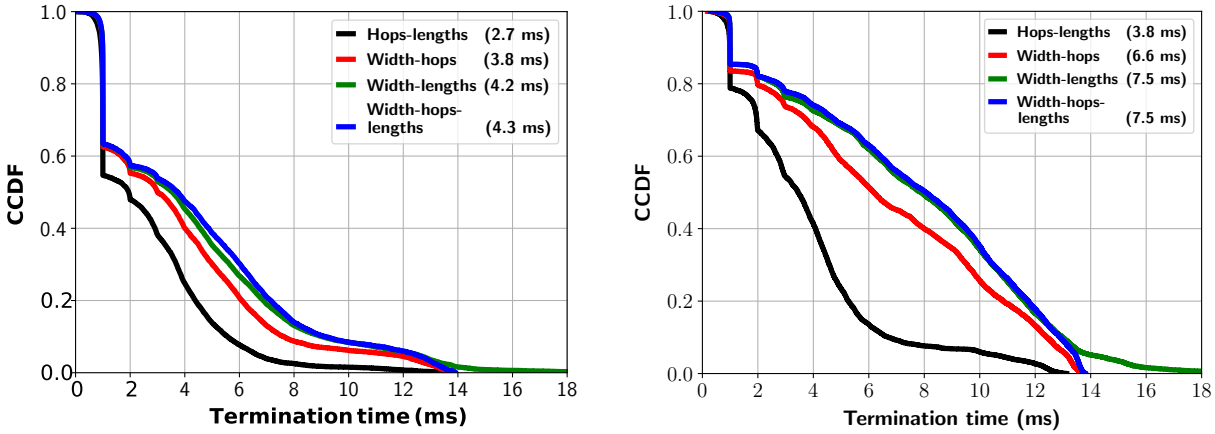


(a) AS 1239 **(b)** AS 3967

Figure 5.2: CCDF of the termination times following the network-wide announcement of a destination for a partial-order vectoring protocol operating on the product order on hops-lengths, width-hops, width-lengths and width-hops-length. Averages are given inside parenthesis.

Failure of a link. Figure 5.3(a) shows the CCDF of the termination times in AS 1239 following the failure of a link, over all possible failures and all 25 independent trials. The product order on the different instantiations of attributes is presented. The curves of termination time have a sharp drop before 1 ms. In particular, 45.3%, 37.4%, 36.7%, and 36.6% of the failures have termination times less than 1 ms, for hops-lengths, width-hops, width-lengths and width-hops-lengths, respectively. As mentioned in Section 5.1, nodes in the test networks are aggregated in clusters and each node inside a cluster has many links to the other nodes in the cluster. Therefore, the failure of a link within the cluster has only a localized impact in the state of the protocol or, put differently, it affects only the elected attributes at the node upstream of the failure.

The curves of termination time also have a heavy tail, which translates that some links failures are associated with long termination times. Concretely, 45.3%, 37.5%, 36.7% and 36.6% of the failures lead to termination times in excess of 10 ms, for width-hops, width-lengths and width-hops-lengths, respectively. The failure of a link may require that some nodes have to settle on worse attributes than those they started out with. Trying to stabilize on worse attributes by always electing the best attributes from among those learned from its neighbors takes many iterations, corresponding to as many paths being explored and, hence, to long termination times. (See Example 3.1)



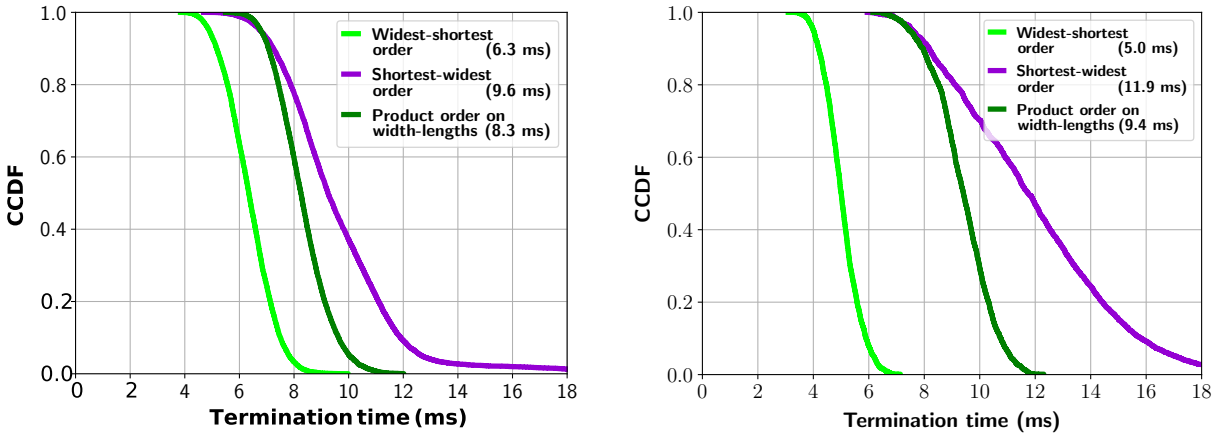
(a) AS 1239 **(b)** AS 3967

Figure 5.3: CCDF of the termination times following the failure of a link for a partial-order vectoring protocol operating on the product order on hops-lengths, width-hops, width-lengths and width-hops-length.

Comparison against various optimality criteria. Figure 5.4(a) shows the CCDF of the termination times in AS 1239 following the network-wide announcement of a destination, over all possible destination nodes and all 25 independent trials. Three orders on width-lengths are presented: (a) the widest-shortest order, in lime; (b) the shortest-widest order, in purple; and (c) the product order, in green. The curves of termination time of the standard vectoring protocol for shortest-widest paths are heavy tailed, with 37.8% of the announcements leading to termination times in excess of 10 ms. The average termination time of the partial-order vectoring protocol is lower than that of the total-order vectoring protocol for shortest-widest paths. The respective averages are 8.3 ms and 9.6 ms. The worse termination time of the standard vectoring protocol is justified by the absence of isotonicity in the shortest-widest order. Without isotonicity, the improvement of an elected attribute at a node may entail the worsen of an elected attribute at an in-neighbor of its. As mentioned before in the case of a link failure, the process whereby a non-restarting vectoring protocol has some nodes settle on worse attributes than those they started out with may take a long time. (See Example 3.4.) We emphasize that the standard non-restarting vectoring protocol for shortest-widest paths not only takes longer to reach a stable state, as the stable state does

not route on shortest-widest paths, in general. Specifically, 30.3% of the source-destination pairs fail to route data-packets on a shortest-widest path from source to destination.

The curves of termination times of standard vectoring protocol for widest-shortest paths and of the partial-order vectoring protocol are both smooth and steep. This is due to the fact that both these orders satisfy isotonicity. The partial-order vectoring protocol has worse termination time than the standard vectoring protocol for widest-shortest paths, the respective averages being 6.3 ms and 8.3 ms. With the partial-order vectoring protocol, nodes have to settle on an average of 2.5 width-lengths per destination, such that 21.2% of the nodes electing more than three width-lengths per destination, whereas, with the standard vectoring protocol, they have to settle on a single width-length. (See Figure 5.1(a)). However, the termination times of the partial-order vectoring protocol are only slightly worse, since the computation of the multiple attributes in a set of dominant attributes is done simultaneously during the execution of the protocol.



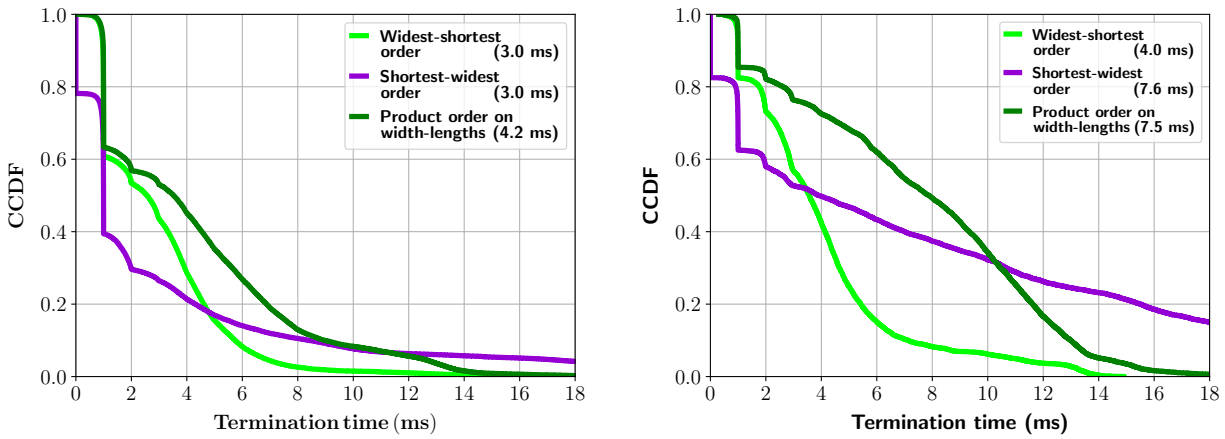
(a) AS 1239 **(b)** AS 3967

Figure 5.4: CCDF of the termination times following the network-wide announcement of a destination for a partial-order vectoring protocol operating on the product order on width-lengths and a standard vectoring protocol operating on the shortest-widest and the widest-shortest order.

Figure 5.5(a) shows the CCDF of the termination times in AS 1239 following the failure of a link, over all possible failures and all 25 independent trials. The stable state of the vectoring protocols for widest-shortest and dominant width-length paths are always affected by the failure of a link. Contrastingly, 21.8% of the failures do not affect the stable state of the vectoring protocol for shortest-widest paths. When the set of dominant paths contains paths with minimum number of links, the failure of a link will prevent the node at its tail from communicating directly with the node at its head. Therefore, at least that node will need to update its stable state. We previously observed that path lengths are strongly correlated with paths hops, which explains why all link failures cause some alteration in the case of widest-shortest paths

and dominant width-length paths. However, shortest-widest paths favor large widths independently of the number of links in a path, implying that some links may not be selected for routing. The failure of such links does not change the state of the protocol.

Last, we observe that, as in the case of a network-wide announcement of a destination, the standard vectoring protocol for shortest-widest paths exhibits the heaviest tail.



(a) AS 1239 **(b)** AS 3967

Figure 5.5: CCDF of the termination times following the failure of a link for a partial-order vectoring protocol operating on the product order on width-lengths and a standard vectoring protocol operating on the shortest-widest and the widest-shortest order.

To sum up, we investigated the efficiency of partial-order non-restarting vectoring protocols by computing statistics of the termination time. Two main conclusions can be drawn.

(a) Following the network-wide announcement of a destination, a partial-order vectoring protocol converges on average faster than a standard vectoring protocol that operates on a total order that does not satisfy isotonicity, and sometimes it converges much faster. The reason is that, when isotonicity does not hold, a node may elect a better attribute that it later has to replace with a worse one, thus prompting a process akin to count-to-infinity, which is possibly long-lasting.

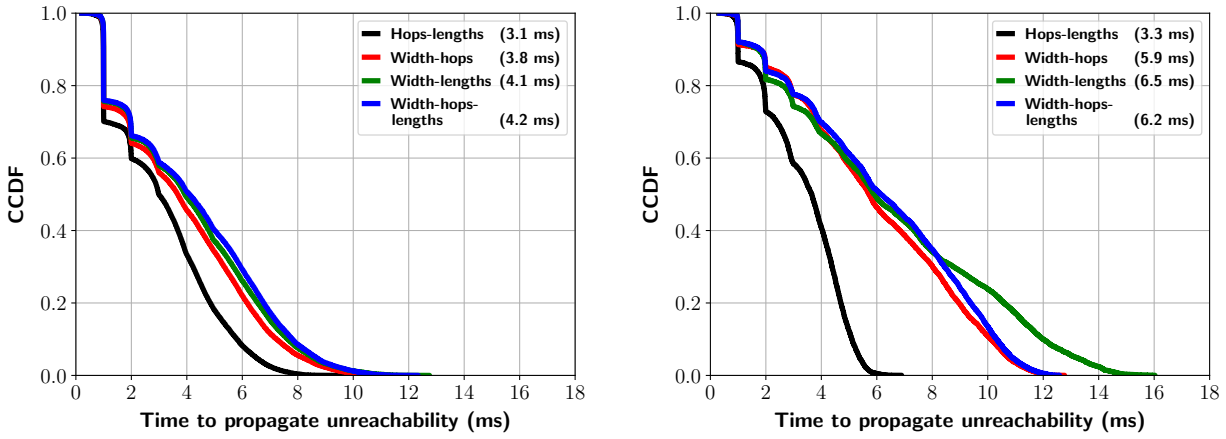
(b) A partial-order vectoring protocol converges only slightly slower than a standard vectoring protocol that computes on an isotone total order. This is because the computation of multiple dominant attributes from sources to destinations is performed in parallel.

5.4 Transient Behaviour of Restarting Protocols

Every computation instance initiated by a destination in a restarting vectoring protocol behaves as the network-wide announcement of the destination in a non-restarting vectoring protocol. (See Exam-

ple 3.5.) Therefore, the transient behaviour of a restarting vectoring protocol can be characterized by the curves of termination time shown in Figure 5.2(a). When a link fails, the node directly affected by the failure propagates unreachability information to its upstream nodes. Figure 5.6(a) shows the CCDF of the time it takes for all nodes to withdraw the elected attributes corresponding to paths that contain the failed link over all links and all trials. This time duration is smaller than the termination time after a network-wide announcement of a destination because only a fraction of nodes are affected by the failure.

While unreachability information propagates upstream the failure, some nodes may be momentarily traffic black holed. Our simulations show that reachability from source to destination is interrupted, on average, on only 0.1% of the cases for width-lengths, width-hops, and width-hops-lengths, and only on 0.2% of the cases for hops-lengths. The duration of the interruptions decreases with the period between consecutive computation instances initiated by the destination.



(a) AS 1239 **(b)** AS 3967

Figure 5.6: CCDF of the time to propagate unreachability for a partial-order vectoring protocol operating on the product order on hops-lengths, width-hops, width-lengths and width-hops-length.

6

Conclusion

Contents

6.1 Summary of Achievements

We presented solutions to two routing problems: (a) optimal path routing problem for an arbitrary optimality criterion; and (b) optimal path routing problem for multiple optimality criteria. These problems can be formulated within the algebraic framework for routing. The first problem is well-known, but it did not have a solution. The second problem is a new one and it is a superset of the first problem.

In the process of developing solutions to these problems, we introduced two new routing concepts: (a) the intersection of a collection of total orders; and (b) the isotonic reduction of a partial order (total order). Equipped with these concepts, we presented a procedure that starts with a collection of total orders and ends with a partial order that is contained in each of the total orders and satisfies isotonicity. Thus, the original optimal path problems are reduced to that computing sets of dominant attributes for an isotone partial order. We devised new partial-order vectoring protocols that find those sets of dominant attributes. The assignment of unique labels to elected attributes enables the expedition of data-packets along any dominant paths. While our running examples emphasized widths and lengths, the solution was developed for arbitrary performance metrics that satisfy the algebraic properties of associativity, commutativity, and inflation.

A preliminary evaluation shows that this approach is promising. The sets of dominant attributes computed on the Rocketfuel networks are rather small. The average number of dominant attributes is below four for networks with hundreds of nodes and thousands of links. Furthermore, our simulations on the Rocketfuel networks revealed that: (a) the computation of the multiple attributes in a set of dominant attributes is done simultaneously; and that (b) isotonicity promotes fast computations. The convergence time of a partial-order vectoring protocol operating on an isotone partial order is only marginally worse than that of a standard vectoring protocol operating on an isotone total order and sometimes much better than that of a standard vectoring protocol operating on a non-isotone total order.

6.2 Future Work

We suspect that other routing problems, not necessarily related to optimality, can be solved with the concepts and protocols introduced. We give two possible examples. The first problem is non-termination of BGP. A routing policy specifies the routing decisions of network administrators and can be formulated within the algebraic framework for routing. It is well-known that BGP fails to terminate for some routing policies [50, 51]. However, there is an extension to BGP, named Self-Stable BGP (SS-BGP), that has been shown to always terminate if the routing policy is isotone [52]. Thus, the problem of the non-termination of BGP can be solved by obtaining isotonic reductions of the total orders underlying the routing policies. A generalization of SS-BGP that operates according to partial orders never fails to terminate and, therefore, allows routing for arbitrary routing policies.

The second problem is that of optimal path routing constrained by a service chain. A service chain is a sequence of services that must be applied to data-packets, each such service being offered by at least one node in the network. In the problem of routing on optimal paths constrained by a service chain, we seek to route data-packets on the optimal paths among those that provide the services of the chain in due sequence [53–57]. This problem can be solved by describing the service chaining constraints in algebraic terms, a process that calls for an isotone partial order. A partial-order vectoring protocol provides optimal path routing constrained by a service chain.

Bibliography

[1] G. Malkin, “RIP version 2,” RFC 2453, 1998.

[2] A. Khanna and J. Zinky, “The revised (arpanet) routing metric,” *ACM SIGCOMM Computer Communication Review*, vol. 19, no. 4, pp. 45–56, 1989.

[3] D. De Couto, D. Aguayo, J. Bicket, and R. Morris, “A high-throughput path metric for multi-hop wireless routing,” in *Proc. of the 9th Annual International Conference on Mobile Computing and Networking (MobiCom 2003)*, 2003, pp. 134–146.

[4] R. Carter and M. Crovella, “Measuring bottleneck link speed in packet-switched networks,” *Performance Evaluation*, vol. 27, pp. 297–318, 2001.

[5] G. Almes, S. Kalidindi, and M. Zekauskas, “A one way packet loss metric for IPPM,” RFC 2680, 1999.

[6] ACM, “ACM SIGCOMM,” 2020. [Online]. Available: http://www.sigcomm.org

[7] Y. Chen and Y. Chin, “The quickest path problem,” *Computers & Operations Research*, vol. 17, no. 2, pp. 153–161, 1990.

[8] T. Ahmed, A. Mehaoua, R. Boutaba, and Y. Iraqi, “Adaptive packet video streaming over ip networks: a cross-layer approach,” *IEEE Journal on Selected Areas in Communications*, vol. 23, no. 2, pp. 385–401, 2005.

[9] B. Carré, “An algebra for network routing problems,” *IMA Journal of Applied Mathematics*, vol. 7, no. 3, pp. 273–294, 1971.

[10] J. L. Sobrinho, “Algebra and algorithms for QoS path computation and hop-by-hop routing in the internet,” in *Proc. of the 16th IEEE International Conference on Computer Communications (INFOCOM 2001)*, vol. 2, 2001, pp. 727–735.

[11] J. L. Sobrinho, “An algebraic theory of dynamic network routing,” *IEEE/ACM Transactions on Networking*, vol. 13, no. 5, pp. 1160–1173, 2005.

[12] T. Griffin and J. L. Sobrinho, “Metarouting,” in *Proc. of the Conference of the ACM Special Interest Group on Data Communication (SIGCOMM 2005)*, 2005, pp. 1–12.

[13] M. Gondran and M. Minoux, *Graphs, Dioids and Semirings*, 1st ed. Springer, 2008.

[14] E. Harzheim, “Ordered sets,” *SIAM Review*, vol. 48, pp. 160–163, 2006.

[15] D. Savage, D. Slice, R. White, J. Ng, P. Paluch, and S. Moore, “Cisco’s enhanced interior gateway routing protocol (EIGRP),” RFC 7868, 2016.

[16] Y. Rehkter, T. Li, and S. Hares, “A border gateway protocol 4,” RFC 4271, 2002.

[17] C. Perkins and P. Bhagwat, “Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers,” *ACM SIGCOMM Computer Communication Review*, vol. 24, no. 4, pp. 234–244, 1994.

[18] J. Chroboczek, “The babel routing protocol,” RFC 6126, 2011.

[19] M. Gouda and M. Schneider, “Maximizable routing metrics,” in *Proc. of the 6th International Conference on Network Protocols (ICNP 1998)*. IEEE, 1998, pp. 71–68.

[20] J. L. Sobrinho, “Fundamental differences among vectoring routing protocols on non-isotonic metrics,” *IEEE Networking Letters*, vol. 1, no. 3, pp. 95–98, 2019.

[21] T. Lengauer and D. Theune, “Efficient algorithms for path problems with general cost criteria,” in *Proc. of the 18th International Colloquium on Automata, Languages, and Programming (ICALP 1991)*, 1991, pp. 314 – 326.

[22] G. Chandranmenon and G. Varghese, “Trading packet headers for packet processing,” *IEEE/ACM Transactions on Networking*, vol. 4, no. 2, pp. 141–152, 1996.

[23] P. Hansen, “Bicriterion path problems,” in *Multiple Criteria Decision Making: Theory and Application*, 1980, pp. 109–127.

[24] E. Martins, “On a multicriteria shortest path problem,” *European Journal of Operational Research*, vol. 16, no. 2, pp. 236–245, 1984.

[25] J. Brumbaugh-Smith and D. Shier, “An empirical investigation of some bicriterion shortest path algorithms,” *European Journal of Operational Research*, vol. 43, no. 2, pp. 216–224, 1989.

[26] F. Guerriero and R. Musmanno, “Label correcting methods to solve multicriteria shortest path problems,” *Journal of Optimization Theory and Applications*, vol. 111, no. 3, pp. 589–613, 2001.

[27] R. Loui, “Optimal paths in graphs with stochastic or multidimensional weights,” *Communications of the ACM*, vol. 26, no. 9, pp. 670–676, 1983.

[28] H. Daellenbach and C. De Kluyver, “Note on multiple objective dynamic programming,” *Journal of the Operational Research Society*, vol. 31, no. 7, pp. 591–594, 1980.

[29] T. Cormen, C. Leiserson, R. Rivest, and C. Stein, *Introduction to Algorithms*, 3rd ed. MIT Press, 2009.

[30] Z. Tarapata, “Selected multicriteria shortest path problems: An analysis of complexity, models and adaptation of standard algorithms,” *International Journal of Applied Mathematics and Computer Science*, vol. 17, no. 2, pp. 269–287, 2007.

[31] A. Basu, C. Ong, A. Rasala, F. Shepherd, and G. Wilfong, “Route oscillations in I-BGP with route reflection,” *ACM SIGCOMM Computer Communication Review*, vol. 32, no. 4, pp. 235–247, 2002.

[32] A. Flavel and M. Roughan, “Stable and flexible iBGP,” *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 4, pp. 183–194, 2009.

[33] Y. Wang, M. Schapira, and J. Rexford, “Neighbor-specific BGP: More flexible routing policies while improving global stability,” *ACM SIGMETRICS Performance Evaluation Review*, vol. 37, pp. 217–228, 2009.

[34] R. Agarwal, V. Jalaparti, M. Caesar, and P. Godfrey, “Guaranteeing BGP stability with a few extra paths,” in *Proc. of the 30th International Conference on Distributed Computing Systems (ICDCS 2010)*. IEEE, 2010, pp. 221–230.

[35] J. Scudder, A. Retana, D. Walton, and E. Chen, “Advertisement of multiple paths in BGP,” RFC 7911, 2016.

[36] N. Kushman, S. Kandula, D. Katabi, and B. Maggs, “R-BGP: Staying connected in a connected world,” in *Proc. of the 4th Symposium on Networked Systems Design and Implementation (NSDI 2007)*, 2007.

[37] F. Wang and L. Gao, “A backup route aware routing protocol - fast recovery from transient routing failures,” in *Proc. of the 27th IEEE International Conference on Computer Communications (INFOCOM 2008)*, 2008, pp. 2333–2341.

[38] Y. Liao, L. Gao, R. Guérin, and Z. Zhang, “Reliable interdomain routing through multiple complementary routing processes,” in *Proc. of the 4th International Conference on Emerging Networking Experiments and Technologies (CoNEXT 2008)*, 2008, pp. 235–247.

[39] I. Ganichev, B. Dai, P. Godfrey, and S. Shenker, “YAMR: Yet another multipath routing protocol,” *ACM SIGCOMM Computer Communication Review*, vol. 40, no. 5, pp. 13–19, 2010.

[40] W. Xu and J. Rexford, “MIRO: Multi-path interdomain routing,” *ACM SIGCOMM Computer Communication Review*, vol. 36, no. 4, pp. 171–182, 2006.

[41] V. Van den Schrieck, P. Francois, and O. Bonaventure, “BGP add-paths: The scaling/performance tradeoffs,” *IEEE Journal on Selected Areas in Communications*, vol. 28, no. 8, pp. 1299–1307, 2010.

[42] Z. Wang and J. Crowcroft, “Quality-of-service routing for supporting multimedia applications,” *IEEE Journal on Selected Areas in Communications*, vol. 14, no. 7, pp. 1228–1234, 1996.

[43] T. Griffin, “The stratified shortest-paths problem,” in *Proc. of the 2nd International Conference on Communication Systems and Networks (COMSNETS 2010)*, 2010, pp. 1–10.

[44] C. Labovitz, A. Ahuja, A. Bose, and F. Jahanian, “Delayed internet routing convergence,” *ACM SIGCOMM Computer Communication Review*, vol. 30, no. 4, pp. 175–187, 2000.

[45] D. Bertsekas and R. Gallager, *Data Networks*, 2nd ed. Prentice Hall, 1991.

[46] J. L. Sobrinho and T. Griffin, “Routing in equilibrium,” in *Proc. of the 19th International Symposium on Mathematical Theory of Networks and Systems (MTNS 2010)*, vol. 9, no. 5, 2010.

[47] J. L. Sobrinho and M. A. Ferreira, “Routing on multiple optimality criteria,” in *Proc. of the Conference of the ACM Special Interest Group on Data Communication (SIGCOMM 2020)*, 2020, pp. 211–225.

[48] N. Spring, R. Mahajan, and D. Wetherall, “Measuring ISP topologies with rocketfuel,” *ACM SIGCOMM Computer Communication Review*, vol. 32, no. 4, pp. 133–145, 2002.

[49] J. Moy, “OSPF version 2,” RFC 2328, 1998.

[50] K. Varadhan, R. Govindan, and D. Estrin, “Persistent route oscillations in inter-domain routing,” *Computer Networks*, vol. 32, no. 1, pp. 1–16, 2000.

[51] T. Griffin, F. Shepherd, and G. Wilfong, “The stable paths problem and interdomain routing,” *IEEE/ACM Transactions On Networking*, vol. 10, no. 2, pp. 232–243, 2002.

[52] J. L. Sobrinho, D. Fialho, and P. Mateus, “Stabilizing BGP through distributed elimination of recurrent routing loops,” in *Proc. of the IEEE 25th International Conference on Network Protocols (ICNP 2017)*, 2017, pp. 1–10.

[53] Sumi Choi, J. Turner, and T. Wolf, “Configuring sessions in programmable networks,” in *Proc. of the 16th IEEE International Conference on Computer Communications (INFOCOM 2001)*, vol. 1, 2001, pp. 60–66.

[54] Z. Cao, M. Kodialam, and T. Lakshman, “Traffic steering in software defined networks: planning and online routing,” *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 4, pp. 65–70, 2014.

[55] P. Quinn and T. Nadeau, “Problem statement for service function chaining,” RFC 7498, 2015.

[56] A. Dwaraki and T. Wolf, “Adaptive service-chain routing for virtual network functions in software-defined networks,” in *Proc. of the 2016 Workshop on Hot Topics in Middleboxes and Network Function Virtualization (HotMiddlebox 2016)*. ACM, 2016, pp. 32–37.

[57] G. Sallam, R. Gupta, B. Li, and B. Ji, “Shortest path and maximum flow problems under service function chaining constraints,” in *Proc. of the 33th IEEE International Conference on Computer Communications (INFOCOM 2018)*, 2018, pp. 2132–2140.

Sequential Computation of Sets of Dominant Attributes

We present a sequential algorithm that computes the sets of dominant attributes from each node to a common destination in a network. We assume that the binary extension operation $\oplus$ on attributes is both inflationary and isotone for the partial order $\preceq$.

Algorithm 5 computes the sets of dominant attributes from a node u to a destination t, for all nodes u. Variable $E[u]$ holds a set of estimates of dominant attributes from u to t, or estimated attributes, and variable $F[u]$ holds a set of attributes from u to t that are already known to be dominant, or final attributes. Attribute a is *dominant with respect to* (w.r.t) set A of attributes if it is not less preferred than any attribute of A and that a is *strictly-dominant* with respect to A if it is dominant with respect to A, but does not belong to A.

In the start of the algorithm, sets $E[v]$ and $F[v]$ are empty, for all nodes v, expect for $E[t]$, which is the singleton $\{\epsilon\}$ (lines 1-4). At each iteration, a dominant attribute from u to t is selected from $E[u]$, for some node u, and the extension of this attribute to each in-neighbor r of u is proposed as a new estimate of a dominant attribute from r to t. In greater detail, the algorithm first selects an attribute a of $E[u]$ that is dominant with respect to all of $E[v]$ (line 6). Attribute a is removed from $E[u]$ and added to $F[u]$ (lines 7-8). Then, for each in-neighbor r of u, the algorithm calculates the extension of the attribute of link ru with a, $a(ru) \oplus b$ (line 10). If $a(ru) \oplus b$ is strictly-dominant with respect to both $E[r]$ and $F[r]$, then $E[r]$ is updated to account for $a(ru) \oplus b$, that is, attribute $a(uv) \oplus b$ is added to the set whereas every attribute that is less preferred than $a(uv) \oplus b$ is removed (lines 11 to 12). The algorithm terminates when all of $E[v]$ are empty.

Algorithm 5 Algorithm that computes the set of dominant attributes from every node to a common destination for an inflationary and isotone partial order.

1: **for** all nodes v **do**
2: $E[v] \coloneqq \emptyset$
3: $F[v] \coloneqq \emptyset$
4: $E[v] \coloneqq \{\epsilon\}$
5: **while** $\cup_{\text{all nodes } v} E[v]$ is not empty **do**
6: Select b from $E[u]$ that is dominant w.r.t. $\cup_{\text{all nodes } v} E[u]$
7: $E[u] \coloneqq E[u] \setminus \{b\}$
8: $F[u] \coloneqq F[u] \cup \{b\}$
9: **for** all r in-neighbor of u **do**
10: $a \coloneqq a(ru) \oplus b$
11: **if** a is strictly-dominant w.r.t. $E[r] \cup F[r]$ **then**
12: $E[u] \coloneqq D(E[u] \cup \{a\})$

In the particular case where attributes are non-negative real numbers, the binary extension operation on attributes is addition ($+$) and the partial order (total order) is the less-than-or-equal-to order ($\leq$), Algorithm 5 is the standard Dijkstra's Algorithm [29].