Algebraic Approaches to Protocol Design

Internet Stability: We Can Do It!

Ashley Flavel and Matthew Roughan
<ashley.flavel@gmail.com>
<matthew.roughan@adelaide.edu.au>

Applied Mathematics
School of Mathematical Sciences
University of Adelaide
A Little Known Fact

The Internet is potentially unstable

- The control plane can fall into oscillatory modes
  - packets loop
  - performance degrades
  - this has been observed
- The worst case: a cascading network failure
  - the whole Internet melts down
- This is not acceptable for critical infrastructure
Can We Fix It

Yes We Can (at least for iBGP)

- **configuration checking for prevention**
  

- **detection, so we can fix problems**
  

- **change control protocols to fix once and for all**
  - but we’d like proof this time, thanks!
Routing Protocols
Packet networks

- The Internet is a packet network
  - data broken into individually addressed packets
  - forwarded hop-by-hop by routers
  - each needs to know where to send each packet

- addresses in the form \(192.168.10.32\)
  - 32 bits (broken into four 8 bit chunks)
  - grouped into blocks by common prefix
    - e.g., \(192.168/16\)

- forwarding done by a lookup in routing table
  - map: destination address \(\rightarrow\) prefix \(\rightarrow\) interface
Forwarding

```
<table>
<thead>
<tr>
<th>dest.</th>
<th>next hop</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>R1</td>
</tr>
<tr>
<td>B</td>
<td>R1</td>
</tr>
<tr>
<td>C</td>
<td>C</td>
</tr>
<tr>
<td>D</td>
<td>R3</td>
</tr>
<tr>
<td>E</td>
<td>R3</td>
</tr>
<tr>
<td>F</td>
<td>R5</td>
</tr>
<tr>
<td>default</td>
<td>R3</td>
</tr>
</tbody>
</table>
```
Forwarding

<table>
<thead>
<tr>
<th>dest.</th>
<th>next hop</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>R1</td>
</tr>
<tr>
<td>B</td>
<td>R1</td>
</tr>
<tr>
<td>C</td>
<td>C</td>
</tr>
<tr>
<td>D</td>
<td>R3</td>
</tr>
<tr>
<td>E</td>
<td>R3</td>
</tr>
<tr>
<td>F</td>
<td>R5</td>
</tr>
<tr>
<td>default</td>
<td>R3</td>
</tr>
</tbody>
</table>
Forwarding

The diagram illustrates a network with routers R1, R2, R3, R4, and R5. Each router is connected to several nodes labeled A, B, C, D, E, and F. The table below shows the routing table for router R2, indicating the next hop for each destination:

<table>
<thead>
<tr>
<th>dest.</th>
<th>next hop</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>R2</td>
</tr>
<tr>
<td>B</td>
<td>R2</td>
</tr>
<tr>
<td>C</td>
<td>R2</td>
</tr>
<tr>
<td>D</td>
<td>R4</td>
</tr>
<tr>
<td>E</td>
<td>R4</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>default</td>
<td>R3</td>
</tr>
</tbody>
</table>

Routing table

The next hop for each destination is indicated by the arrows in the diagram, showing the path data packets should take to reach their destination.
Route Loop

We need to have consistent routing tables!!!
Routing

The Internet is highly dynamic:
How do we maintain a routing table?

- easy for most people
  - just have a default next hop
  - often called the gateway
  - goes to someone who knows more, e.g. your ISP

- but how does an ISP do it?
  - can’t do by hand
    - not robust (to links failures say)
    - doesn’t scale
  - they use **routing protocols**
  - used to build dynamic routing tables
Shortest-Path Routing

- simple approach to routing is to use the shortest path between two locations
  - optimal (in some sense)
- “shortest” with respect to arbitrary distance metric
  - hop count
  - physical distance
  - inverse of bandwidth
  - administratively configured distance
- how do we calculate shortest-paths
  - link-state protocol
  - distance-vector protocol
Distance-Vector Routing

- routers tell each other about the shortest path that they have found out about
  - sometimes called “routing by rumour”
  - more formally we call it the distributed Bellman-Ford algorithm
- Bellman’s principle is at work to guarantee convergence

- there are a few extras we need to deal with, but they don’t matter for this talk
  - loop detection
  - count-to-infinity
Distance-Vector example

<table>
<thead>
<tr>
<th>subnet</th>
<th>10.1.0.0/24</th>
</tr>
</thead>
<tbody>
<tr>
<td>next hop</td>
<td>no route</td>
</tr>
<tr>
<td>distance</td>
<td>infinity</td>
</tr>
</tbody>
</table>

10.1.0.0/24 subnet next hop distance

Ethernet 0

10.1.0.0/24 subnet next hop distance

10.1.0.0/24 subnet next hop distance

Algebraic Approaches to Protocol Design – p.11/53
Distance-Vector example

<table>
<thead>
<tr>
<th>Subnet</th>
<th>10.1.0.0/24</th>
</tr>
</thead>
<tbody>
<tr>
<td>Next Hop</td>
<td>no route</td>
</tr>
<tr>
<td>Distance</td>
<td>infinity</td>
</tr>
</tbody>
</table>

Ethernet 0

R1

10.1.0.0/24

R2

10.1.0.0/24

next hop: R1

distance: 2

R3

10.1.0.0/24

next hop: R1

distance: infinity

R4

10.1.0.0/24

next hop: no route

distance: infinity

R5

10.1.0.0/24

next hop: R1

distance: infinity
Distance-Vector example

<table>
<thead>
<tr>
<th>Subnet</th>
<th>10.1.0.0/24</th>
</tr>
</thead>
<tbody>
<tr>
<td>Next hop</td>
<td>R2</td>
</tr>
<tr>
<td>Distance</td>
<td>3</td>
</tr>
</tbody>
</table>

10.1.0.0/24 subnet next hop distance

10.1.0.0/24 subnet next hop distance 1

10.1.0.0/24 subnet next hop distance 2

10.1.0.0/24 subnet next hop distance 3
Distance-Vector example

- **R1**
  - Subnet: 10.1.0.0/24
  - Next hop: Ethernet 0
  - Distance: 1

- **R2**
  - Subnet: 10.1.0.0/24
  - Next hop: R1
  - Distance: 2

- **R3**
  - Subnet: 10.1.0.0/24
  - Next hop: R2
  - Distance: 3

- **R4**
  - Subnet: 10.1.0.0/24
  - Next hop: R3
  - Distance: 4

- **R5**
  - Subnet: 10.1.0.0/24
  - Next hop: R5
  - Distance: 0
Distance-Vector example

```
<table>
<thead>
<tr>
<th></th>
<th>subnet</th>
<th>next hop</th>
<th>distance</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1</td>
<td>10.1.0.0/24</td>
<td>Ethernet 0</td>
<td>1</td>
</tr>
<tr>
<td>R2</td>
<td>10.1.0.0/24</td>
<td>R1</td>
<td>2</td>
</tr>
<tr>
<td>R3</td>
<td>10.1.0.0/24</td>
<td>R2</td>
<td>3</td>
</tr>
<tr>
<td>R4</td>
<td>10.1.0.0/24</td>
<td>R3</td>
<td>4</td>
</tr>
<tr>
<td>R5</td>
<td>10.1.0.0/24</td>
<td>R2</td>
<td>3</td>
</tr>
</tbody>
</table>
```

R2 3  
R1 2  
R2 3  
R3 4  
Algebraic Approaches to Protocol Design – p.11/53
Distance-Vector Routing

- So that’s distance-vector routing
  - routing protocol
    - automatically builds routing tables
    - adapts to network changes
  - it's a simplification
    - loop detection, count-to-infinity, ...
    - timers, soft-state vs hard-state, ...
- only solves shortest-paths
  - non-negative link weights
  - guaranteed to converge
  - guaranteed to be consistent
- we need a bit more for the Internet
Today’s Internet
Where are we now?

The Internet
- billions of hosts, 100’s of thousands of locations
  - highly dynamic
  - highly heterogeneous
- Simple routing protocols as above don’t cut it
  - don’t scale
  - shortest-paths isn’t what we want to do
- the Internet is a network of networks
  - crude 2-level hierarchy
  - broken into Autonomous Systems
  - each is run separately
Internet Topology

The Internet has \( \sim 30,000 \) Autonomous Systems (ASes)

- Each AS is a separately managed network
- Within an AS may use different routing, technology, management, ...
- May be a LAN, WAN, or combination

- Example ASes:
  - ISP (Internet Service Provider)
  - Campus network
  - Enterprise network
  - Hosting center

- Interior details of an AS are not exposed
the Internet

Tier–3

Tier–2

Backbone

ISP 1

ISP 2

servers

hosting center

campus network

WAN links

LAN links

AS

exchange point

routers

switches
Number of ASes

http://www.cidr-report.org/
Different Flavours of Routing

Routing is different inside an AS from between ASes

- **intra-domain** (inside an AS)
  - called Interior Gateway Routing (IGP) protocols
  - examples: OSPF, RIP, EIGRP, IS-IS, ...
  - shortest-paths is common

- **inter-domain** (between ASes)
  - called Exterior Gateway Routing (EGP) protocols
  - one defacto standard BGPv4
    - Border Gateway Protocol
    - must talk internationally
  - NOT shortest-paths
Routing Policy

- shortest path is not enough for inter-domain routing
  - we need policy routing
- policy is a set of arbitrary rules for routing
  - we may prefer to route the cheapest way
  - we prefer to route to route traffic with $X$
    - maybe $X$ provides better QoS
    - maybe $X$'s network is more secure
- hot-potato routing
  - reduce cost of carrying traffic on our network by dropping it onto someone else's as soon as possible
- all else being equal, maybe we use shortest-paths
A real example from [1]

ABILENE
AS 11537

ASNet
AS 9264

GEANT
AS 20965

Academic Services Network (ASNet) Global Backbone.

REACH
AS 4637

TeleGlobe
AS 6453

JANET
AS 786

The Chinese University of Hong Kong
http://www.cuhk.edu.hk/en/

CUHK
AS 3661

The Chinese University of Hong Kong
http://www.cuhk.edu.hk/en/

Rule: academic networks prefer to use academic networks
Rule: all else being equal use the shortest path
A real example from [1]

Rule: academic networks prefer to use academic networks
BGP
BGP

- Border Gateway Protocol [2]
- BGP has to support all of this stuff
  - generically called policy based routing
  - I will use the term path-vector routing
- incredibly flexible
  - choice of path based on policy, not distance
- large, complex dynamic system
  - hard to understand
  - hard to predict
  - hard to optimize
BGP means

- Standards: IETF RFC 1771
- Optional extensions:
  - RFC 1997 BGP Communities Attribute
  - RFC 2439 BGP Route Flap Damping
  - RFC 2796 BGP Route Reflection
  - RFC 3065 AS Confederations for BGP
- Implementation details
  - timers, proprietary extensions (WEIGHT), ...
- Routing policy configuration languages
  - vendor specific
- Current practises in management of inter-domain routing (e.g. RFC 1772, RFC 2270, ...)
How BGP works

Its complicated

- let's pretend it's like the previous example
- except, we allow the choice of preferred routes to be completely arbitrary
  - assume we can write a list of possible routes for each destination
  - possible routes can be restricted by policy
  - order the list by policy preferences
- it becomes an optimization problem
  - call it the **stable-paths problem** [3, 4]
  - looking for a set of stable paths which match policies
Simple Example 1

- destination is AS 0
- tables show acceptable routes in order of preferences
- result is a shortest-path tree
- change to policy (at nodes 1 & 4)
- no solution
- endless oscillation
Bad Widget

- change to policy (at nodes 1 & 4)
- no solution
- endless oscillation
- change to policy (at nodes 1 & 4)
- no solution
- endless oscillation
Bad Widget

- change to policy (at nodes 1 & 4)
- no solution
- endless oscillation
Bad Widget

- change to policy (at nodes 1 & 4)
- no solution
- endless oscillation
Bad Widget

- change to policy (at nodes 1 & 4)
- no solution
- endless oscillation
Bad Widget

- change to policy (at nodes 1 & 4)
- no solution
- endless oscillation
Bad Widget

- change to policy (at nodes 1 & 4)
- no solution
- endless oscillation
Bad Widget

- change to policy (at nodes 1 & 4)
- no solution
- endless oscillation
Bad Widget

- change to policy (at nodes 1 & 4)
- no solution
- endless oscillation
Bad Widget

- change to policy (at nodes 1 & 4)
- no solution
- endless oscillation
- change to policy (at nodes 1 & 4)
- no solution
- endless oscillation
Bad Widget

- change to policy (at nodes 1 & 4)
- no solution
- endless oscillation
Bad Widget

- change to policy (at nodes 1 & 4)
- no solution
- endless oscillation
Bad Widget

- Change to policy (at nodes 1 & 4)
- No solution
- Endless oscillation
Bad Widget

- Change to policy (at nodes 1 & 4)
- No solution
- Endless oscillation
Bad Widget

- change to policy (at nodes 1 & 4)
- no solution
- endless oscillation
Bad Widget

- change to policy (at nodes 1 & 4)
- no solution
- endless oscillation
Bad Widget

- change to policy (at nodes 1 & 4)
- no solution
- endless oscillation
Is “Bad Widget” really bad?

- route oscillation has been observed in the Internet
  - MED oscillation has been observed
    - more details later

- best case
  - inconsistent routing tables ⇒ route loops
  - packets are dropped
  - extra load on routers and links

- worst case
  - enough routes churn to overload some routers
  - router failures cause more routing churn, so that more routers fail, resulting in a cascade
  - Internet crashes
Proof by movie reference

TERMINATOR 3
iBGP and MED oscillation in detail
Two flavours of BGP
- between ASes: eBGP
- inside AS: iBGP
  - propagate (external) routes to all (internal) routers
  - simple version only sends information one hop
  - so we need a clique for iBGP signalling
Route-reflectors

- fully connected mesh doesn’t scale
  - requires $N(N-1)$ connections
- solution: create hierarchy
  - route-reflectors are parents
  - client routers as children
- common case: 2-layer hierarchy
  - multiple layers are possible
  - not a tree: for redundancy
- RR hierarchies very common in large networks
Route-reflectors

- Simple RR hierarchy

Route Reflectors

1 — 2

3 — 4 — Clients — 5 — 6

- distance-vector-like protocol propagates routes
  - not shortest iBGP path
  - routes go up → over → down
Route-reflectors

- Simple RR hierarchy

Route Reflectors

1 — 2

3 — 4 — Clients — 5 — 6

- distance-vector-like protocol propagates routes
  - not shortest iBGP path
  - routes go up → over → down
Example MED oscillation

Policy preferences: all else being equal
- prefer lower MED score
- prefer closer (IGP) routes
Example MED oscillation

Initially RRs learn clients' routes

- RR 1 chooses route via 4 (lower IGP distance)
- can't compare MEDs between AS X and Y
Example MED oscillation

RRs tell each other about their choice
- RR 1 excludes route via 4 because of higher MED
- RR 1 chooses closest route via 3 (shorter than via 5)
Example MED oscillation

RRs tell each other about their choice
- RR 2 learns of closer route (via 2)
- again can’t compare MEDs between AS X and Y
Example MED oscillation

RR 2 tells RR 1 about the changed choice

- RR 1 no longer learns of the route via 5
- RR 1 switches back to shortest path via 4
Example MED oscillation

RR 1 tells RR 2 about the changed choice
- RR 2 chooses route via 5 (lower MED)
- we are back where we started
Intuition

- what is really going on?
  - RRs hide some information
    - only pass on their “best” route
  - best doesn’t mean shortest iBGP distance
    - prefer lower MED score
    - prefer closer IGP routes (not iBGP distance)
  - route choices are not transitive:
    \[ A > B \quad \text{and} \quad B > C \]
    does not imply
    \[ A > C \]

- combination of non-transitive choices, and information hiding results in oscillation
iBGP oscillation

- there are many other iBGP oscillation examples
- iBGP is controlled by one AS
  - we should never allow oscillation
- how can we fix it
  - check configuration for stability
    - how do you check all cases?
  - detect and fix oscillation
    - may take some time
- design network for stability
  - reduces network flexibility
  - hard to set up (not always possible)
  - doesn’t deal with dynamic nature of network
  - doesn’t stop all types of oscillation
Fix the protocol

A better approach is to fix the protocol so it can’t oscillate:

- small change (preferably)
- incremental deployment
- provably removes the problem
Algebraic abstraction of the problem
Algebras

- an algebra over a field is a vector space with an operation usually called multiplication that satisfies certain axioms:
  - vector space is a set with addition $\oplus$ and scalar multiplication.
  - multiplication $\otimes$ combines two vectors to form a third
    - satisfies certain axioms, e.g., distributivity.

- Example: Min-plus algebra (often look at max-plus)
  - defined over real numbers
  - addition: $x \oplus y = \min(x, y)$
  - multiplication: $x \otimes y = x + y$
Min-plus = shortest-paths

Two operations

- Advertising a route to a neighbour \( \equiv \otimes \)
  - add the distance of the current route, to the link distance

- Selecting the best route \( \equiv \oplus \)
  - choose the shortest route (using min)

We can prove things (Bellman’s principle) in this algebra because it is strictly **monotonic** (for positive link weights, e.g. distances):

\[ x \oplus (x \otimes y) = x \]

i.e. path lengths increase when we advertise them to neighbours
Distance Vector example

R1

- subnet: 10.1.0.0/24
- next hop: Ethernet 0
- distance: 1

R2

- subnet: 10.1.0.0/24
- next hop: no route
- distance: infinity

R3

- subnet: 10.1.0.0/24
- next hop: no route
- distance: infinity

R4

- subnet: 10.1.0.0/24
- next hop: no route
- distance: infinity

R5

- subnet: 10.1.0.0/24
- next hop: no route
- distance: infinity
Distance Vector example

<table>
<thead>
<tr>
<th>subnet</th>
<th>10.1.0.0/24</th>
</tr>
</thead>
<tbody>
<tr>
<td>next hop</td>
<td>no route</td>
</tr>
<tr>
<td>distance</td>
<td>infinity</td>
</tr>
</tbody>
</table>

- **R1**
  - subnet: 10.1.0.0/24
  - next hop: Ethernet 0
  - distance: 1

- **R2**
  - subnet: 10.1.0.0/24
  - next hop: R1
  - distance: 2 = 2 + ∞

- **R3**
  - subnet: 10.1.0.0/24
  - next hop: no route
  - distance: infinity

- **R4**
  - subnet: 10.1.0.0/24
  - next hop: no route
  - distance: infinity

- **R5**
  - subnet: 10.1.0.0/24
  - next hop: no route
  - distance: infinity

1 x 1 = 2

Ethernet 0

10.1.0.0/24
Algebraic description of iBGP

- Min-plus isn’t rich enough to describe iBGP
- But we can do it
  - create algebra of route signatures
  - combine with link labels for route propagation
  - create a new way to compare routes
- Has to implicitly contain iBGP rules
  - edge routers introduce a single route from adjacent ASes
    - we can consider these all to be equal except for MEDs and IGP distances
  - need to implement the rule that routes go up → over → down
A routing algebra consists of an ordered sextet

\[(L, \Sigma, f, W, \preceq, \otimes)\].

where

- labels \(L\) describe links
- signatures \(\Sigma\) describe known routes
- weights \(W\) are used for comparing routes
- a function \(f\) that maps signatures into weights:
  \[f : \Sigma \rightarrow W\]
- a total order \(\preceq\) on \(W\); and
- a binary operation \(\otimes\) that maps pairs of a label and a signature into a signature, i.e.,
  \[\otimes : L \times \Sigma \rightarrow \Sigma.\]
Algebraic description of iBGP

link labels \[ L = \{d,u,o\} \times \mathbb{Z}^+ \]
\[
\uparrow \quad \uparrow
\]
edge type
head node identifier

route signatures \[ \Sigma = \{d,u,o,e\} \times \mathbb{Z}^+ \times \mathbb{Z}^+ \]
\[
\uparrow \quad \uparrow \quad \uparrow
\]
edge current egress
type node node

Note that empty/invalidate route signature \( \phi \) is implicitly included in \( \Sigma \).
Operator $\otimes$

<table>
<thead>
<tr>
<th>$\otimes$</th>
<th>Signature, $\sigma \in \Sigma$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\otimes$</td>
<td>$(e, k, k)$ $(d, i, k)$ $(o, i, k)$ $(u, i, k)$</td>
</tr>
<tr>
<td>$L$</td>
<td>$(d, j)$ $(d, j, k)$ $(d, j, k)$ $(d, j, k)$ $(d, j, k)$</td>
</tr>
<tr>
<td>$(o, j)$</td>
<td>$(o, j, k)$ $\phi$ $\phi$ $(o, j, k)$</td>
</tr>
<tr>
<td>$(u, j)$</td>
<td>$(u, j, k)$ $\phi$ $\phi$ $(u, j, k)$</td>
</tr>
</tbody>
</table>

Operator implicitly expresses the rules of iBGP routing

- a route must go up $\rightarrow$ over $\rightarrow$ down
- invalid combinations result in $\phi$
- head node ID $j$ is set to current node
- origin $k$ doesn’t change
Route selection

Route selection $\oplus$ is defined by $f : \Sigma \rightarrow W$

$$f(\sigma) = \begin{cases} 
(\text{dist}(i,k), k), & \text{if } \sigma = (\ast, i, k), \\
(\infty, \infty), & \text{if } \sigma = \emptyset.
\end{cases}$$

where $\text{dist}(i,k)$ is the IGP distance from node $i$ to node $k$, and the comparison operator $\prec$ implements:

- prefer shorter distances $\text{dist}(i,k)$
- break ties by choosing smaller $k$
Strict Monotonicity

Strict monotonicity ensures the preference of a route strictly decreases when it is propagated, i.e., for all \( \sigma \in \Sigma - \{\emptyset\} \), and \( \lambda \in L \)

\[
f(\sigma) \prec f(\lambda \otimes \sigma)
\]

where the \( \prec \) operator indicates a strict preference.

- Note that in iBGP, the preference operator works on the IGP distance
- that is **NOT** the same as the iBGP distance
- it can decrease when a route is propagated
- iBGP algebra is not monotonic
The Fix

- just make the algebra monotonic
  - just add a hop count into the signatures $\Sigma$
  - perform comparisons first on the hop count (before IGP distance)
  - proof of convergence then takes three lines

- no change to BGP protocol
  - all the require information is already present
  - hop count is contained in “cluster-list” attribute

- doesn’t even require changes to routers, as Cisco have programmable decision stack

- can be deployed incrementally
Conclusion

- oscillation happens today in iBGP
- we can fix it with a very small change to routers
- there is more to this
  - the simple fix can distort routing decisions
  - doesn’t respect MEDs
- but we can fix these as well if we are slightly more sophisticated in our use of the algebra
References


Extra Slides
New Algebra

\[ L = \{d, u, o\} \times \mathbb{Z}^+ \]

- \( \uparrow \) edge type
- \( \uparrow \) head node identifier

\[ \Sigma = \mathbb{Z}^+ \times \{d, u, o, e\} \times \mathbb{Z}^+ \times \mathbb{Z}^+ \]

- \( \uparrow \) hop count
- \( \uparrow \) edge type
- \( \uparrow \) current node egress node

Note that empty/invalidate route signature \( \phi \) is implicitly included in \( \Sigma \).
## Algebraic description

### Operator $\otimes$

<table>
<thead>
<tr>
<th>Operator $\otimes$</th>
<th>Signature, $\sigma \in \Sigma$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$(n, e, k, k)$</td>
</tr>
<tr>
<td>$(d, j)$</td>
<td>$(n + 1, d, j, k)$</td>
</tr>
<tr>
<td>$(o, j)$</td>
<td>$(n + 1, o, j, k)$</td>
</tr>
<tr>
<td>$(u, j)$</td>
<td>$(n + 1, u, j, k)$</td>
</tr>
</tbody>
</table>

- Operator includes same implicit rules as in iBGP previously
- Hop count is incremented as route is propagated
Route selection

Route selection $\oplus$ is defined by $f : \Sigma \rightarrow W$

$$f(\sigma) = \begin{cases} 
(n, \text{dist}(i, k), k), & \text{if } \sigma = (n, *, i, k), \\
(\infty, \infty, \infty), & \text{if } \sigma = \emptyset. 
\end{cases}$$

comparison operator $\prec$ is defined by $<$ in lexical order

- prefer smaller number of iBGP hops
- break ties with shorter distances $\text{dist}(i, k)$
- break ties by choosing smaller $k$

Now algebra is strictly monotonic by construction.