(draft) MRAI Model

Mattia Milani*
*Dept. of Information Engineering and Computer Science, University of Trento, Italy
mattia.milani@studenti.unitn.it

I. Main Idea

This document has to bee take as my flow of consiousness, it will be probably written again, corrected and changed many times.

The main idea behind the Minimum Route Advertisement Interval (MRAI) model that I have in mind is that: In [1] has been shown how to model and algebricaly solve problems with SemiRings (SR) like hop-count shortest-path and even policies. One of the main ideas is to consider the network as a matrix A where each row represent a node i and each column represent a function to reach node j. The adjacency matrix A permits to have a mastrix X that represent the state of the network, this matrix X evolves in time, at each step a routing algebra is applied to each edge solving the local shortest path problem. At some point, a further step would not make the matrix change, so convergence has been reached.

My is an asynchronous approach to the matrix evolution. My Idea is to don't apply the algebra to the whole matrix A but apply it locally to single edges selecting the best route in the knowledge buffer of the node neighbourhood. The algebra would select the best possible path in the vector of possibilities in the nodes and then introudce this coice in the set of possibilities of the edge. The step can be executed indipendently from each other edge in the network. At the end, when no steps produce changes the network is converged.

II. Goals

The main goal is to be able to model MRAI and solve problems caused by it.

III. Semirings, Routing algebra

I should start in order, first of all I have to define what is a routing algebra.

Definition 1. A routing algebra is a tuple $(\mathcal{S}, \oplus, \mathcal{F}, \overline{0}, \overline{\infty})$ where:

- S the set of weights with $\overline{0} \in S$ and $\overline{\infty} \in S$,
- $\oplus$: $SXS \to S$ is the choice operator
- F is a set of functions from $S \to S$

such that: $\oplus$ is Associative, commutative and selective, $\overline{0}$ is an annhilator for $\oplus$, $\overline{\infty}$ is an identitity function for $\oplus$ and a fixed point $\forall f \in F$

It's possible to define a total order between elements.

$$x \leq y \triangleq x \oplus y = x \quad x < y \triangleq x \leq y \wedge x \neq y$$

$$\text{x} \leq y \iff x \oplus y = x \quad (1)$$

each function F represents a procedure to transform a route's weight when that route is extended along and edge. Elements of F will be used to label edges, so we have arc functions.

The algebra is distributive if:

$$\forall f \in F, x, y \in S : f(x \oplus y) = f(x) \oplus f(y) \quad (2)$$

The algebra is increasing if:

$$\forall f \in F, x \in S : x \leq f(x) \quad (3)$$

The algebra is strictly increasing if:

$$\forall f \in F, x \in S : x \neq \overline{\infty} \Rightarrow x < f(x) \quad (4)$$

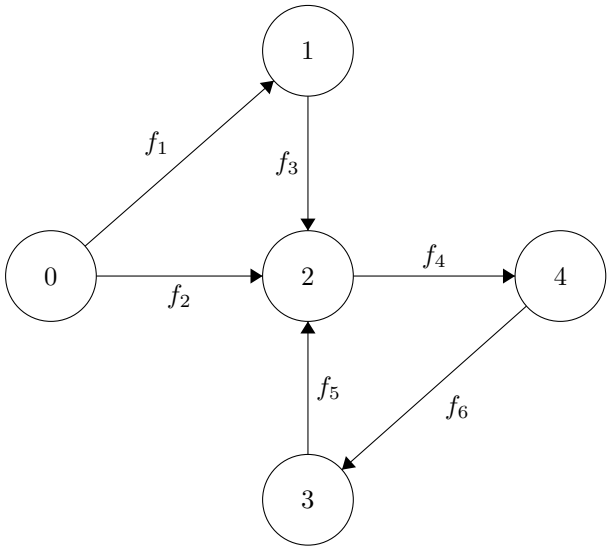
IV. Network

A network is represented by a directed graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ where $\mathcal{V}$ is a set of n nodes $\mathcal{V} = \{0, 1, ..., n-1\}$ and $\mathcal{E}$ is a set of arcs. A configuration of $\mathcal{G}$ with respect to a routing algebra $(\mathcal{S}, \oplus, \mathcal{F}, \overline{0}, \overline{\infty})$ is a mapping from $\mathcal{E}$ to F.

Such mappings will be represented by an $n \times n$ adjacency matrix $\mathcal{X}$ where $\mathcal{X}_{ij} \in F$.

I assume the constant function $f_{\overline{\infty}} \in F$ exists that always returns the invalid weight, function used to represent missing edges.

For example we can have the following graph:



That has as $\mathcal{X}$ the one in Eq. (5)

$$\mathcal{X} = \begin{bmatrix} f_{\overline{\infty}} & f_1 & f_2 & f_{\overline{\infty}} & f_{\overline{\infty}} \\ f_{\overline{\infty}} & f_{\overline{\infty}} & f_3 & f_{\overline{\infty}} & f_{\overline{\infty}} \\ f_{\overline{\infty}} & f_{\overline{\infty}} & f_{\overline{\infty}} & f_{\overline{\infty}} & f_4 \\ f_{\overline{\infty}} & f_{\overline{\infty}} & f_5 & f_{\overline{\infty}} & f_{\overline{\infty}} \\ f_{\overline{\infty}} & f_{\overline{\infty}} & f_{\overline{\infty}} & f_6 & f_{\overline{\infty}} \end{bmatrix} \tag{5}$$

The definition of path must be independent from the definition of the current graph, we can have paths that are not corresponding to the current graph $\mathcal{G}$.

A *path* $p = [(v_1, v_1^{'}), ..., (v_m, v_m^{'})]$ is a (possibly empty) sequence of arcs such that $v_i^{'} = v_{i+1}$ for all $0 \leq i < m$. Arcs are not members of $\mathcal{E}$ but are arbitrary members of $\mathbb{N} \times \mathbb{N}$.

I will assume the same notational convenience on paths as [1]. $p = []$ represent the empty path, $(v_1, v_1^{'}) :: q$ when $0 \leq i < m$ where $q = [(v_2, v_2^{'}), ..., (v_k, v_k^{'})]$, a path is simple if it never visit a vertex mre than once. the $\perp$ simbol to represent the invalid path, $\mathcal{P}$ represent the set of paths, $|p|$ is the length of the path p

V. Path algebras

We are interested in protocols which track the path tat routes are generated along and remove any looping path. Given the routing algebra $\mathcal{A} = (\mathcal{S}, \oplus, \mathcal{F}, \overline{0}, \overline{\infty})$ we can add path to the algebra as follows:

$\mathbb{PA}(\mathcal{A}) = (((S - \{\infty\}) \times \mathcal{P}) \cup \{\infty^{'}\}, \oplus^{'}, F^{'}, (\overline{0}, []), \infty^{'})$

In $\mathbb{PA}(\mathcal{A})$ weights are in form (s, p) or $\infty^{'}$. where $s \in S - \{\infty\}$ and $p \in \mathcal{P}$. $\forall (s,p) : (s,p) \oplus^{'} \infty^{'} = (s,p)$, non-$\infty^{'}$ rutes are compared lexicographically:

$$(s_1, p_1) \oplus (s_2, p_2) \triangleq \begin{cases} (s_1, p_1) & \text{if } s_1 = (s1 \oplus s2) \neq s_2, \\ (s_2, p_2) & \text{if } s_1 \neq (s1 \oplus s2) = s_2, \\ (s_1, p_1 \hat{\oplus} p_2) & \text{if } s_1 = s_2, \end{cases} \tag{6}$$

where

$$p_1 \hat{\oplus} p_2 \triangleq \begin{cases} p_1 & \text{if } |p_1| < |p_2|, \\ p_2 & \text{if } |p_1| > |p_2|, \\ \text{dict}(p_1, p_2) & \text{otherwise}, \end{cases} \tag{7}$$

dict returns the smallest path in dictionary order.

I define the concatenation operator as in [1]. $\hat{::}$ takes an arc and a path, $(i,j)\hat{::}p$, the result is $\perp$ if the first node of p is not j.

The $F^{'}$ fr the path-algebra is the collections of policies functions $g_{f,u,v}$ defined as:

$$g_{f,u,v}(\infty^{'}) = \infty^{'}$$

$$g_{f,u,v}(s,p) \triangleq \begin{cases} \infty^{'} & \text{if } f(s) = \infty, \\ \infty^{'} & \text{if } (u,v)\hat{::}p = \perp, \\ (f(s), (u,v)\hat{::}p) & \text{otherwise}, \end{cases} \tag{8}$$

Definition 2. A path algebra is a tuple $(\mathcal{S}, \oplus, \mathcal{F}, \overline{0}, \overline{\infty}, \textit{path})$ where:

- $(\mathcal{S}, \oplus, \mathcal{F}, \overline{0}, \overline{\infty})$ is a routing algebra;
- *path*: $S \to \mathcal{P}$ is a function that returns the path the weight was generated along

such that:

P1) $\text{path}(x) = \perp \iff x = \overline{\infty}$

P2) $\text{path}(x) = [] \Leftarrow x = \overline{0}$

P3) $\text{path}(\mathcal{X}_{ij}(x)) = (i,j) :: \textit{path}(x) \iff i \notin \textit{path}(x) \wedge j = \textit{src}(\textit{path}(x))$

P3) implies that if a path algebra is increasing it is automatically strictly increasing. $\mathbb{PA}(\mathcal{A})$ is a path algebra for any routing algebra, the path function is defined as $\textit{path}(\infty^{'}) = \perp$ and $\textit{path}((s,p)) = p$

VI. Buffer algebra

Thinking about MRAI could be easy to think about operation on buffers of paths rather than the comparison of only paths.

First of all I have to define what is a buffer. A *buffer* $b = [p_1, p_2, ..., p_k]$ is a (possibily empty) sequence of paths such that p_x is unique in b for all $0 \leq x \leq k$. For notational convenience I will write b as $[\ ]$ when $k = 0$. Buffers, for our pourpose, could be only simple. this restriction could be removed in future. I will refer to the set of buffers with $\mathcal{B}$ and the length of a buffer b as $|b|$. I will refer to the sub-buffer $b\{x,y\}$ as the subset $[p_x, ..., p_y]$ of b with $0 \leq x < y < |b|$ an invalid subbuffer will return the invalid operator $\perp$.

For this reason I thought about the following algebra. Given a routing algebra $\mathcal{A} = (\mathcal{S}, \oplus, \mathcal{F}, \overline{0}, \overline{\infty})$ one way to add buffers to $\mathcal{A}$ is as follows:

$\mathbb{BA}(\mathcal{A}) = (((S - \{\infty\}) \times \mathcal{B}) \cup \{\infty^{'}\}, \oplus^{'}, F^{'}, (\overline{0}, []), \infty^{'})$

In the augmented algebra, $\mathbb{BA}(\mathcal{A})$, weights are of the form $\infty^{'}$ or (s,b) where $s \in S - \{\infty\}$ and b is a buffer. For all (s,b) we have $(s,b) \oplus^{'} \infty^{'} = \infty^{'} \oplus^{'} (s,b) = (s,p)$. The result of a comparison is the the resulting best path in the buffer, with $p \in \mathcal{P}$. Non-$\infty^{'}$ routes are compared lexicographically:

$$(s_1, b_1) \oplus (s_2, b_2) \triangleq \begin{cases} (s_1, \hat{\oplus} b_1) & \text{if } s_1 = (s1 \oplus s2) \neq s_2, \\ (s_2, \hat{\oplus} b_2) & \text{if } s_1 \neq (s1 \oplus s2) = s_2, \\ (s_1, \hat{\oplus} b_1 \hat{\hat{\oplus}} \hat{\oplus} b_2) & \text{if } s_1 = s_2, \end{cases} \tag{9}$$

where

$$\hat{\oplus} b \triangleq \begin{cases} \infty^{'} & \text{if } b = [], \\ \textit{best}(b) & \textit{otherwise} \end{cases} \tag{10}$$

where, given p_0 as the first element of $b = [p_0, p_1, ..., p_{|b-1|}]$

$$\textit{best}(b) = p_0 \hat{\hat{\oplus}} \textit{best}(b\{1, |b-1|\}) \tag{11}$$

This function is used to recursively chose the best path in the buffer.

whith:

$$p_1 \hat{\hat{\oplus}} p_2 \triangleq \begin{cases} p_1 & \text{if } |p_1| < |p_2|, \\ p_2 & \text{if } |p_1| > |p_2|, \\ \text{dict}(p_1, p_2) & \text{otherwise}, \end{cases} \tag{12}$$

where $\text{dict}(p_1, p_2)$ returns the smallest path in dictionary order.

I define the concatention operator $\hat{::}$ that takes a path and a buffer. The result of $p\hat{::}b$ is b if p is already in b, otherwaise it returns $b :: p$. For example:

$$[(3,4),(4,5)]\hat{::}[[(3,4),(4,5)]] = [[(3,4),(4,5)]]$$

$$[(1,9)]\hat{::}[[(3,4),(4,5)]] = [[(3,4),(4,5)],[(1,9)]]$$

If in the buffer is already present a path to the same destination with the same first edge, the new path should substitute the old path in the buffer. In this way we have "Implicit Withdrawing". If a neighbour change idea on how to reach a destination the node can't have in it's knowledge both paths, but just the last one.

In a more formal way:

- Given (i, j) as the edge with the neighbour chosen.
- Given p as the best path to reach a destination given by the neighbour j.
- Given b as the local buffer.
- The first concatenation is applied to the path $(i,j)\hat{::}p$, if the concatenation is correct we obtain a path $p^{'} = [(i,j),(j,j_1),...,(j_k,k)]$ The destination of this path is the node k.

To sum up, the resulting buffer of the concatenation will be:

$$b\hat{::}p^{'} = \begin{cases} b & \text{if } p^{'} \in b, \\ \{b - p_p\} \cup p^{'} & \text{if } \exists p_p \in b : p_p[0] = p^{'}[0] \\ & \qquad \wedge\, dst(p_p) = dst(p^{'}) \\ b \cup p^{'} & \text{otherwise,} \end{cases} \tag{13}$$

The $F^{'}$ for the buffer algebra is the collection of policy functions $g_{f,p}$ which are defined as:

$$g_{f,p}(\infty^{'}) = \infty^{'}$$

$$g_{f,p}(s,b) \triangleq \begin{cases} \infty^{'} & \text{if } f(s) = \infty, \\ (f(s), p\hat{::}b) & \text{otherwise,} \end{cases} \tag{14}$$

Definition 3. A buffer algebra is a tuple $(\mathcal{S}, \oplus, \mathcal{F}, \overline{0}, \overline{\infty}, \textit{buffer})$ where:

- $(\mathcal{S}, \oplus, \mathcal{F}, \overline{0}, \overline{\infty})$ is a routing algebra;
- *buffer*: $S \to \mathcal{B}$ is a function that returns the buffer the weight was generated along

such that:

B1) $\text{buffer}(x) = \perp \iff x = \overline{\infty}$

B2) $\text{buffer}(x) = [] \Leftarrow x = \overline{0}$

B3)

$$\textit{buffer}(\mathcal{X}_{ij}(x)) = \begin{cases} \infty^{'} & \text{if } (i,j)\hat{::}path(x) = \perp, \\ b & \text{if } (i,j)\hat{::}path(x) \in b, \\ (i,j)\hat{::}path(x) \hat{::} b & \text{otherwise,} \end{cases} \tag{15}$$

$\mathbb{BA}(\mathcal{A})$ is a path algebra for any $\mathcal{A}$. Its buffer function is defined as $\text{buffer}(\infty^{'}) = \perp$ and $\text{buffer}((s,p)) = p\hat{::}b$.

VII. Solving problems

Having $\mathcal{G}(\mathcal{V}, \mathcal{E})$ as a directed graph. Let $\mathcal{A}$ be the adjacency matrix of $\mathcal{G}$ where $\mathcal{A}_{ij} \in F$ is the weight of the edge from i to j. Missing edges are represented by $f_{\overline{\infty}}$.

Let $\mathbb{M}_n(S)$ be the set of $n \times n$ matrices over S. Each matrix $X \in \mathbb{M}_n(S)$ represent one possible global state of the routing protocol. The row X_i represent all the current buffers of node i and X_{ij} is node i's current input buffer for reaching node j.

Every node has input buffers from which it will retrives information, an input buffer for a node is the actual knowledge of all its input neighbours.

Given Equation 4 of [1], I report an equation that will be useful later. define the application of A to X as:

$$A(X)_{ij} \triangleq \oplus_{0 \le k < n} A_{ik}(X_{kj})$$

If we apply also the idendity value in I_{ij} we then obtain that: node i's new route to j is the best choice out of the extensions of the best routes offered to it by each knowledge buffer of its neighbours k

VIII. Iterative solution

I therefore define the function $\sigma(X)_{ij}$ as:

$$\sigma(X)_{ij} \triangleq A(X)_{ij} \oplus I_{ij}$$

Given I as the identity matrix.

The result of $\sigma(X)_{ij}$ is that i will now extend the buffer in X_{ij} with the best path (extended with i) contained in all the output buffers of its neighbourhood.

Refers to [2] for the asynchronous convergence. I will explain it to be sure of my vision of it.

IX. Asynchronous convergence

Now that we have all the buffers that we need we can exploit an asynchronous convergence system. I use the model from Üresing & Dubois [insert citation] which assumes a discrete and linear notion of time $\mathbb{T}$ denoting the times of events of interest in the network

Definition 4. A Schedule consists of a pair of functions:

- $\alpha : \mathbb{T} \to 2^V$ is the activation function, where $\alpha(t)$ is the set of nodes which update their routing table at time t
- $\beta : \mathbb{T} \times V \times V \to \mathbb{T}$ is the data flow function where $\beta(t,i,j)$ is the time at which the information used by node i at time t was sent by node j.

where V is the set of nodes in the network, such that:

S1) Every node continues to activate indefinetely

$$\forall it. \exists k. i \in \alpha(t+k)$$

S2) information only travels forward in time

$$\forall ijt. \beta(t,i,j) < t$$

S3) no link has a loss rate of 100%

$$\forall ijt.\exists t'.\forall k.\beta(t'+k,i,j) \neq t$$

Nothing in S2 or S3 forbids the data flow function β from delaying, losing reordering or dublicating messages.

For a given schedule (α, β) and starting state X we define δ, the asynchonous version of σ as follows:

$$\delta^0(X)_{ij} \triangleq X_{ij}$$
$$\delta^t(X)_{ij} = \begin{cases} \oplus_k A_{ik}(\delta^{\beta(t,i,k)}(X)_{kj}) \oplus I_{ij} & \text{if } i \in \alpha(t), \\ \delta^{t-1}(X)_{ij} & \text{otherwise,} \end{cases} \quad (16)$$

Giving what is demonstrated in [2] if my implementation is strictly incrasing then absolute convergence is guaranted.

REFERENCES

[1] M. L. Daggitt and T. G. Griffin, "Rate of convergence of increasing path-vector routing protocols," in *2018 IEEE 26th International Conference on Network Protocols (ICNP)*. IEEE, 2018, pp. 335–345.

[2] M. L. Daggitt, A. J. Gurney, and T. G. Griffin, "Asynchronous convergence of policy-rich distributed bellman-ford routing protocols," in *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, 2018, pp. 103–116.