Local Pruning for Information Dissemination in Dynamic Networks for Solving the Idempotent Semiring Algebraic Path Problem

Kiran K. Somasundaram and John S. Baras

Abstract— **We present a method, inspired from routing in dynamic data networks, to solve the Semiring Algebraic Path Problem (SAPP) for dynamic graphs. The method can be used in dynamic networks such as Mobile Ad Hoc Networks, where the network link states are highly dynamic. The algorithm makes use of broadcasting as primary mechanism to recompute the SAPP solution. The solution suffers from broadcast storm problems, and we propose a selective broadcasting mechanism that reduces the broadcast storm. We call this method local pruning and prove its correctness.**

I. Introduction

In distributed computing, several different computations can be viewed from a common viewpoint as instances of the *Semiring Algebraic Path Problems* (SAPPs) on graphs [19]. These include well known graph problems, such as shortest path computations, and problems that seemingly have nothing to do with graphs, such as solutions to system of linear equations and regular expressions describing the language accepted by a finite automaton.

The functions that can be expressed as a SAPP on graphs have a linear algebraic interpretation over semirings [11]. Consequently, several well-know solution methods in linear algebra, such as Gauss elimination, Jacobi and Gauss-Seidel iterations, can be applied to the *weighted adjacency matrix* associated to this algebraic path problem. These solution methods can be classified as direct solution (elimination) methods and iterative solution procedures. While the direct solutions are easily implemented in a centralized processor, the iterative solutions are more amenable to be implemented in a distributed setting. For these iterative procedures, which are primarily based on matrix powers [19], the convergence depends on the structure of the weighted adjacency matrix [19], [4].

The advent of pervasive mobile devices has created the need for new algorithm design mechanisms for a mobile computing platform. The limited capabilities of these mobile devices has created several interesting problems that did not exist in traditional networks. These limitations have made communication an expensive commodity for distributed algorithms. In particular, network-wide broadcast, as a communication primitive, has an unreasonable cost in these networks. In communication networks, broadcasting is used for information dissemination in several algorithms, e.g., ARPANET modified algorithm, Optimized Link State Routing (OLSR), Topology Broadcast based on Reverse-Path Forwarding (TBRPF), Scalable Broadcast Algorithm (SBA), Dominant Pruning, Ad Hoc Broadcast Protocol (AHBP), Light and Efficient Network-Wide Broadcast (LENWB), trust certificate flooding algorithms. A summary of these algorithms is presented in [2], [25], [10]. In mobile networks, where the information state is highly dynamic, information dissemination by broadcasting causes significant overhead. This problem is called the *broadcast storm problem* [2]. Consequently, several local pruning approaches have been proposed to reduce the broadcast storm problem [25]. These pruning algorithms are aware of their local neighborhood information and selectively broadcast only "significant" information of their local neighborhood for the distributed computation. This selective broadcast reduces the broadcast storm problem.

In this paper, we consider distributed computation of the SAPP over dynamic graphs. Many problems where the dynamics arise from the mobility of the processors or from the changes in the element values of the weighted adjacency matrix fall under dynamic SAPPs. Even for a few changes in the weighted adjacency matrix, re-computation of the SAPP using iterative procedures, in general, can exhibit long convergence times. The communication cost associated with the message passing for these iterations might be prohibitively expensive. In this paper, we develop a new scheme for solving the dynamic SAPP by selective broadcasting that has a relatively inexpensive communication requirements for the re-computations. This algorithm is inspired from ideas from routing in dynamic networks (e.g., ARPANET, OLSR). We consider a special class of semirings called *idempotent semirings*. Several functions including routing objectives, overlay constructions, max-marginals, hypothesis testings and trust certification calculations can be abstracted using idempotent semirings.

The main contribution of this paper is identifying a selective broadcasting method that ensures that the SAPP can be computed on dynamic graphs. We call these methods local pruning methods and provide proofs of correctness. We provide sufficient conditions, called *strict monotonicity* conditions, when the pruning algorithms preserve the SAPP solutions in a distributed setting.

The paper is organized as follows. In Section II, we introduce the SAPP for idempotent semirings. In section III, we present the pruning algorithms and their correctness proofs for the information broadcast.

Kiran K. Somasundaram and John S. Baras are with the Institute for Systems Research and the Department of Electrical and Computer Engineering, University of Maryland, {kirans, baras}@umd.edu

This material is based upon work supported by the Communications and Networks Consortium sponsored by the U.S. Army Research Laboratory under the Collaborative Technology Alliance Program, Cooperative Agreement DAAD19-01-2-0011 and the MURI Award Agreement W911-NF-0710287 from the Army Research Office.

II. Semiring Systems

We define a semiring system as any system, monolithic or networked, that has the function of computing a special algebraic problem called the Semiring Algebraic Path Problem (SAPP) [11]. To illustrate this special path problem, we will introduce some notations and definitions from graph theory and some algebraic structures.

A. Notations and Definitions from Basic Graph Theory

We give definitions from graph theory that are necessary to describe the semiring algebraic path problem. For a detailed reference of graph theoretic notations and definitions, see [6]. Let $G(V, E)$ denote a labeled directed graph. The vertex set V is the set of processors in the system, which we call *nodes*, and the arc set $E \subseteq V \times V$ denotes the adjacencies between these nodes. We consider only arc labels: for any arc $(u, v) \in E$, there is an associated label $a(u, v)$.

A subgraph of G, denoted by $G' \subseteq G$, is a graph $G'(V', E')$ such that $V' \subseteq V$ and $E' \subseteq E$ (restricted to $V' \times V'$). For any vertex $i \in V'$, the set of arcs incident to i in any subgraph G' is denoted by $\Omega_i^{G'}$. The set of paths in any subgraph G' between a pair of vertices $i, j \in G'$ is denoted by $P_{ij}^{G'}$. A path $p \in P_{ij}^{G'}$ is a sequence of vertices $p = (i = u_1, u_2, \ldots, u_n = j)$, where $u_k \in V'$, $1 \leq k \leq n$. Also, for $i \in V'$, $P_{ii}^{G'}$ contains the empty path $p = (i)$. For any path $p = (i = u_1, u_2, \ldots, u_n = j) \in P_{ij}^{G'}$, the *successor vertex* for a vertex u_k, $1 \leq k < n$, in p is

$$\eta_p^{u_k} = u_{k+1}.$$

For any pair of vertices $i, j \in V$, let $2^{P_{ij}^{G'}}$ be the power set of $P_{ij}^{G'}$, which is the set of all subsets of $P_{ij}^{G'}$. For $P \in 2^{P_{ij}^{G'}}$ the successor vertex set of the first vertex, i, is given by

$$H_P = \{\eta_p^i : p \in P\}.$$

B. Classical Shortest Path Problem

The classical shortest path problem is a well studied graph optimization problem in computer science and operations research [11], [1]. Interestingly, a number of algorithms that solve the shortest path problem can be generalized to solve a particular algebraic path problem in *semirings* [11]. Before we introduce shortest path problem on a labeled graph $G(V, E)$.

Consider an example labeled graph $G(V, E)$ shown in Figure 1. The arc labels $a_{uv} \in \mathbb{Z}_+, (u, v) \in E$, and artificial ∞ weights (labels) are used for non-existent arcs. The weighted adjacency matrix corresponding to the labeled graph in Figure 1 is

$$\mathbf{A} = \begin{bmatrix} 1 & 4 & 7 & \infty \\ \infty & 3 & 2 & \infty \\ \infty & 1 & \infty & 3 \\ 5 & \infty & 6 & 2 \end{bmatrix}.$$

For the shortest path problem, the weight of a path p is given by a composition rule $w(p) = \sum_{(u,v)\in p} a(u, v)$, and

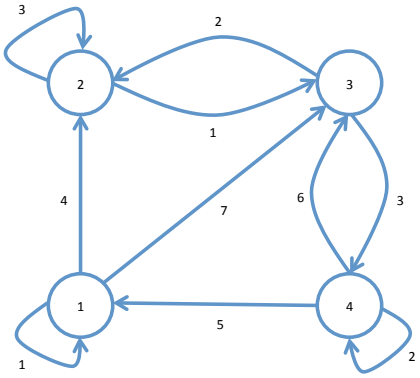

Fig. 1. Example network for shortest path computation

the weight of shortest path between a pair of vertices $i, j \in V$ is given by another composition rule

$$x_{ij}^G = \min_{p \in P_{ij}^G} w(p). \tag{1}$$

Since the number of paths in P_{ij}^G is, in general, numerous, it becomes computationally intractable to compute the shortest path metric using Equation (1). However, the rules of composition for the shortest path problem have an underlying structure that enables efficient computation. We will illustrate this using the instance in Figure 1. Consider the computation of the shortest path length from 1 to 4:

$$x_{14}^G = \min_{p \in P_{14}^G} w(p).$$

The shortest path from 1 to 4 is $(1, 2, 3, 4)$, which has weight 8. For this path, the sub-path from $(2, 3, 4)$ must be one of the shortest paths from 2 to 4; otherwise, we can construct an alternative better path from 1 to 4. This observation is referred to as Bellman's optimality principle [24]. This can be generalized as follows.

Consider the shortest path from i to j. If $i \neq j$, then this path is of the form $(i = u_1, u_2 \ldots, u_n = j)$. For this shortest path, the sub-path $p' = (u_2, u_3, \ldots, u_n = j)$ must be the shortest path from u_2 to u_n, and consequently, the shortest path metric is given by $x_{ij} = a_{ik} + x_{kj}$, for $k = u_2$. Thus, the shortest path metric computation for $i \neq j$ can be written as $x_{ij} = \min_{k \in V}(a_{ik} + x_{kj})$. For $i = j$, we need also consider the empty path from j to j, i.e, (j). For the shortest path computation, the weight of an empty path is 0 (there is no cost in staying at j). Thus, the shortest path computation from j to j is $x_{jj} = \min\{\min_{k \in V}(a_{jk} + x_{kj}), 0\}$. For all pairs of vertices, we can express these computations as a system of equations:

$$\begin{aligned} x_{ij}^G &= \min_{k \in V}(a_{ik} + x_{kj}^G), \text{ for } i \neq j, \text{ and} \\ x_{jj}^G &= \min\{(\min_{k \in V}(a_{jk} + x_{kj}^G)),\ 0\}. \end{aligned} \tag{2}$$

Note, the above systems of equations is a fixed point equation in $x_{ij}^G,\ i,j \in V$ that is referred to as the *Bellman* equation [5]. In general, the Bellman equation does not necessarily have a unique solution. However, there is always a minimal solution among the set of equations [4], [11]. These minimal solutions correspond to the weights of the shortest path between $i,\ j \in V$. The system of equations in (2) yields a computationally efficient method to compute the weight of the shortest paths for all pair of nodes. For the example in Figure 1, the unique solution of the system is given by the solution matrix

$$X = \begin{bmatrix} 0 & 4 & 6 & 8 \\ 9 & 0 & 1 & 4 \\ 8 & 2 & 0 & 3 \\ 5 & 8 & 6 & 0 \end{bmatrix}.$$

Suppose $x_{ij}^G,\ i,j \in V$ is a minimal solution to Equation (2), the corresponding set of *solution paths* is given by

$$P_{ij}^{G^*} = \{p_{sol} \in P_{ij}^G :\ w(p_{sol}) = x_{ij}^G\},\ i,j \in V. \quad (3)$$

(A *solution path*, p_{sol}, is a shortest path with respect to the weight w.)

In the forthcoming sections, we will generalize the structure of the shortest path problem to an algebraic structure called *semirings* and the system of equations in (2) to an algebraic path problem called *semiring algebraic path problem*.

C. Semiring Algebra

A semiring is an algebraic structure $(S, \oplus, \otimes)$ that satisfies the following axioms:

(A1) (S,⊕) is a commutative monoid with a neutral element $\textcircled{0}$:

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

(A2) (S,⊗) is a monoid with a neutral element $\textcircled{1}$*, and an absorbing element* $\textcircled{0}$:

$$\begin{aligned} a \otimes (b \otimes c) &= (a \otimes b) \otimes c \\ a \otimes \textcircled{1} = \textcircled{1} \otimes a &= a \\ a \otimes \textcircled{0} = \textcircled{0} \otimes a &= \textcircled{0} \end{aligned}$$

(A3) $\otimes$ *distributes over* $\oplus$:

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

The set S is called the carrier set of the semiring. For a detailed reference on semirings, see [11]. The axiom *A3*, which we call *semiring distribution*, plays a vital role in several computations. Computations that obey the semiring axioms can be solved distributively and in many cases efficiently. To illustrate this point, consider the classical shortest path problem (Subsection II-B). This problem can be expressed by the $(\hat{\mathbb{Z}}_+, \min, +)$ semiring (the structure satisfies all the semiring axioms), where $\hat{\mathbb{Z}}_+ = \mathbb{Z}_+ \cup \{\infty\}$, and $\textcircled{0}=\infty$ and $\textcircled{1}=0$.

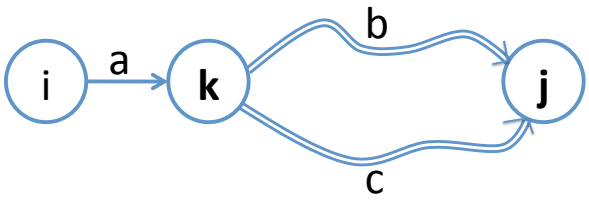

Fig. 2. Spoon network to illustrate semiring distribution

Consider the shortest path computation on the spoon-like labeled graph shown in Figure 2. The figure shows a labeled directed graph G, where the arc (i,k) has a label $a \in \hat{\mathbb{Z}}_+$. There are two paths from k to j with weights $b, c \in \hat{\mathbb{Z}}_+$. When computing the weight of the shortest path from i to j in Equation (1), there are only two paths to consider: $x_{ij}^G = \min\{a+b, a+c\}$. The $(\hat{\mathbb{Z}}_+, \min, +)$ structure, by virtue of its semiring distribution, reduces the computation to

$$\begin{aligned} x_{ij}^G &= a + \min\{b, c\} \quad (\because \text{Axiom A3}), \\ &= a + x_{kj}^G. \end{aligned}$$

The semiring distribution, in essence, "factors-out" the common computations, i.e., in the above example x_{kj}^G, and consequently, reduces the computational effort. In fact, for the general shortest path problem discussed in Subsection II-B, the semiring distribution of $(\hat{\mathbb{Z}}_+, \min, +)$ reduces the set of equations in Equation (1) to that in Equation (2)! In the next section, we will generalize the shortest path computation to an algebraic path problem computation, and will illustrate how the semiring distribution yields, again, a reduced representation.

D. Semiring Algebraic Path Problems

Consider a directed labeled graph G with the arc labels $a_{uv} \in S,\ (u,v) \in E$, where S is the carrier set of some semiring $(S, \oplus, \otimes)$. For any path $p = (i = u_1, u_2, \ldots, u_n = j) \in P_{ij}^G$, the weight of p is defined by an *arc composition rule*:

$$w(p) = a_{u_1 u_2} \otimes a_{u_2 u_3} \otimes \cdots \otimes a_{u_k u_{k+1}} \otimes \cdots \otimes a_{u_{n-1} a_n}. \quad (4)$$

The weight of an empty path is defined to be $\textcircled{1}$. Given the weights of all paths in P_{ij}^G, the aggregate weights from node i to node j is defined by a *path composition rule*:

$$x_{ij}^G = \oplus_{p \in P_{ij}^G}\ w(p). \quad (5)$$

The above compositions for $i,j \in V$, which forms a set of equations, is called the *Semiring Algebraic Path Problem* (SAPP).

These rules of composition can been seen as generalized version of the rules of computation for the shortest path problem (Subsection II-B), i.e., $\otimes$ and $\oplus$ are generalizations of $+$ and $\min$ operators, respectively. Factoring-out common terms (Subsection II-C) in the set of equations, by applying

the semiring distribution property, the SAPP can be reduced to

$$\begin{aligned} x_{ij}^G &= \oplus_{k \in V} \ (a_{ik} \otimes x_{kj}^G), \text{ for } i \neq j, \text{ and} \\ x_{jj}^G &= (\oplus_{k \in V} \ (a_{jk} \otimes x_{kj}^G)) \oplus 0. \end{aligned} \tag{6}$$

In general, the fixed points of the SAPP need not necessarily correspond to solutions of the path composition (5). However, under certain conditions of minimality, illustrated in Subsection II-D, some of the fixed points have a correspondence to paths. For a detailed reference on the SAPP, see [19], [11].

A number of composition rules used in networked systems can be expressed as a SAPP over some semiring algebras. For instance, SAPPs appear in control theory - dynamic programming [24], in information and coding theory - factor graphs [16], [14], in network security - PGP trust evaluations [23] and in algebraic routing - BGP [12], [20]. For other examples of SAPPs in networked systems, see [4].

E. SAPPs for Idempotent Semirings

Among the classes of semiring structures, a particularly useful and prevalent structure is the idempotent semiring algebra [11]. For these semirings, the idempotent law holds for $\oplus$, i.e., for an idempotent semiring $(S, \oplus, \otimes)$,

$$a \oplus a = a, \ a \in S.$$

This idempotent law induces a partial order in S. We define the partial order by

$$a, b \in S, \ a = a \oplus b \iff a \geq b \tag{7}$$

Note, the partial order can also be defined as a dual order [3] to the definition in Equation (7). To remain consistent with the notation used in [11], we follow the definition in Equation (7). This partial order makes the constituent monoid $(S, \oplus)$ of the semiring canonically ordered [11], and such semirings are also referred to as *dioids*, in the literature [11].

These dioids have an interesting property that there exists a minimal solution to the SAPP (Equation (6)) [11], similar to that described in Subsection II-B for the shortest path problem, that can be associated with a path-set for its solution. From henceforth, all solutions considered, in this paper, for the SAPP will be minimal solutions. Unlike the shortest path problem, for dioids, the solution does not necessarily correspond to single path (p_{sol} in Equation (3)). Instead, for a general dioid, the solutions correspond to a set of paths. We call this a *solution path-set*. Given a dioid $(S, \oplus, \otimes)$, and a solution $x_{ij}^G, \ i, j \in V$ to the SAPP (Equation (6)), we can define a textitsolution path-set $P_{sol} \in 2^{P_{ij}^G}$ as a "minimal" path set such that

$$\oplus_{p \in P_{sol}} = x_{ij}^G.$$

Clearly, there would be many such P_{sol} path-sets, and these path-sets are minimal in the sense that no subset of P_{sol} is another solution path-set. We denote the set of all solution path-sets by $\mathcal{P}_{ij}^{G^*}$.

III. Solving the Semiring Algebraic Path Problem in Dynamic Networks

The shortest path problem with time-varying weights is a well-studied problem in the context of data networks [5]. For instance, the initial ARPANET routing protocol was a highly ambitious adaptive routing protocol, which adapted its routes according to the link congestions (weights). However, the algorithm suffered from stability issues. The algorithm was later modified and fixed using a link state version [5] that broadcast periodically the link state information (congestion weights) across the network to compute the same routing objective. It was shown that this mechanism of computing the shortest path for dynamic networks was efficient and more stable [5].

Recently, a similar paradigm was adopted in proactive routing protocols in MANETs. However, in MANETs the link state information is very dynamic and broadcasting the entire link state is expensive and results in broadcast storm problems. Instead, methods of selective broadcasting were proposed. We will illustrate one such method used: Optimized Link State Routing (OLSR).

A. Selective Broadcasting in OLSR

In Optimized Link State Routing (OLSR) protocol [9], [8], every host in the network discovers its local neighborhood by heartbeat periodic HELLO messages [7]. Since every host broadcasts to its neighbors the set of neighbors that it can hear, every host discovers its one-hop and two-hop neighbors. Figure 3 shows the neighborhood that host h discovers from the HELLO messages. The neighbor discovery protocol [7] is designed to discover only symmetric neighbors (that can hear each other), and consequently all the links discovered are undirected.

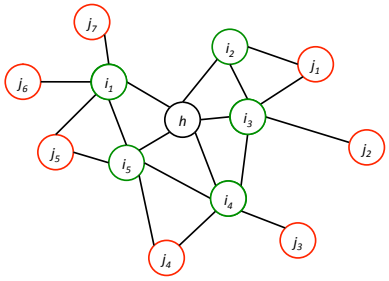

Fig. 3. Local View of Host h

The original version of OLSR [9], treats the topology pruning problem over a static graph. Every host solves a set-cover problem locally to find the minimum set of one-hop neighbors that cover all the two-hop neighbors. For instance, in the example graph shown in Figure 3, the host h selects from $\{i_1, i_2, \ldots, i_5\}$ a minimal subset of neighbors that cover all two-hop neighbors $\{j_1, j_2, \ldots, j_7\}$. This special set of one-hop neighbors is called Multi-Point Relays (MPRs) in OLSR. The pruning problem to compute the MPRs is shown to be NP-hard and a greedy heuristic was proposed in [15]. For the example of Figure 3, the host selects $\{i_1, i_3, i_4\}$ that

covers all the two-hop neighbors. Then the host broadcasts links $\{(h, i_1), (h, i_3), (h, i_4)\}$ across the entire network. Every host in the network performs similar broadcast. It was proved in [13] that this pruning mechanism preserves the shortest path, in *hop-count*, from every source to target host in the network.

However, the OLSR pruning mechanism is capable of handling, only, binary link state information (ON or OFF). It does not offer guarantees on the quality of the routing paths preserved, as a result of its pruning. In [22], we extended the pruning methods to guarantee that the shortest paths in the $(\hat{\mathbb{Z}}_+,)$ semiring algebra are preserved in the pruned graph.

In the rest of the paper, we develop pruning methods that will preserve the SAPP solutions for any general idempotent semiring. Then the pruned graph can be broadcast for computing the SAPP solutions.

B. Mathematical Notations and Definitions for the Pruning Problem

We introduce the notion of hop-count based neighborhoods. The hop-count of a path $p \in P_{ij}^{G'}$, denoted by $hc(p)$, is the number of arcs in p. Then the minimum hop-count distance between a pair of vertices $i, j \in V$ is

$$d_{ij}^{G'} = \min_{p \in P_{ij}^{G'}} hc(p).$$

Neighborhoods in G are defined relative to any vertex $h \in V$, which we call the *host* of the neighborhood. The k-hop neighborhood of a host h in G is the vertex set

$$N_h^k = \{j \in V : \ d_{hj}^G \leq k\}.$$

Here, k is called the *size of the neighborhood*. The boundary set for N_h^k is

$$\partial N_h^k = N_h^k \backslash N_h^{k-1},$$

where $N_h^0 = \{h\}$, and $N_h^k = \emptyset, \ k < 0$. Let N_h^{k-} denote the exclusive neighborhood, which is the neighborhood excluding h, i.e., $N_h^{k-} = N_h^k \backslash \{h\}$.

Consider a special labeled subgraph $G_h^{local} \subseteq G$ that contains only the vertices in N_h^k and all arcs between them except those between any two vertices of the boundary set, i.e., the vertex set is N_h^k and the arc set is $\{(u, v) \in E : u, v \in N_h^k \text{ and } \{u, v\} \not\subseteq \partial N_h^k\}$. We will, later, call this labeled subgraph the *local view* of h. We abuse notation a bit, to define the leaf set of the local view, L_h, the set of vertices in N_h^{k-} that have no children in G_h^{local}. Clearly, $N_h^k \subseteq L_h$. These quantities are illustrated in the example shown in Figure 4.

For the local pruning algorithms and their correctness proofs, we work with different local views of the graph G, so, for better readability, we denote the different graph quantities, defined above, relative to the local view, G_h^{local}. The set of paths from h to any vertex $j \in N_h^k$, $P_{hj}^{G_h^{local}}$, is denoted by $P_{hj}^{h-local}$, which we call the *h-local path-set*. We denote by $x_{hj}^{h-local}$, $j \in N_h^k$, the solution to the SAPP restricted to G_h^{local}, i.e., $x_{hj}^{h-local} = x_{hj}^{G_h^{local}}$. The corresponding set of solution path-sets is denoted by $\mathcal{P}_{hj}^{h-local*} (= \mathcal{P}_{hj}^{G_h^{local}*})$, which we call the set of *h-local solution path-set*.

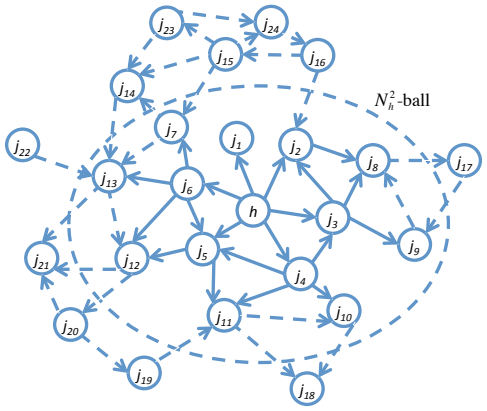

Fig. 4. Illustration of neighborhood terminology. The figure shows a directed graph G. The neighborhood (of size $k = 2$) corresponding to a host vertex h is indicated by N_h^2-ball. The neighborhood vertices, $N_h^2 = \{h, j_1, j_2, \ldots, j_6\}$, induce the *local-view* subgraph G_h^{local}. The arcs of G_h^{local} are indicated by solid lines. Note that the arcs between the boundary vertices, $\partial N_h^2 = \{j_7, j_8, j_9, j_{10}, j_{11}, j_{12}, j_{13}\}$, are not included in the arc set of G_h^{local}. Here, the leaf set $L_h = \{j_1\} \cup N_h^2$.

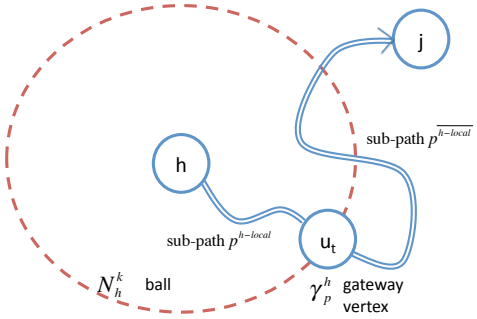

Fig. 5. Gateways for path p in local view

Finally, we introduce the notion of *gateways* of paths and path-sets relative to the local view G_h^{local}. For any path $p = (h = u_1, u_2, \ldots, u_n = j) \in P_{hj}^G$, the *gateway* of p in G_h^{local}, denoted by γ_p^h, is the first vertex of p that is in the leaf set L_h. i.e., $\gamma_p^h = u_t$ if and only if $u_t \in L_h$ and $u_s \notin L_h, 1 \leq s < t$. If the path p never intersects L_h, i.e., $u_s \notin L_h, \ 1 \leq s \leq n$, then γ_p^h is not defined. For a path-set $P \in 2^{P_{hj}^G}$, we denote the set of gateway vertices by $\Gamma_P^h = \{\gamma_p^h : \ p \in P\}$.

We denote by $p^{h-local}$ the sub-path $(h = u_1, u_2, \ldots, u_t = \gamma_p^h)$, which is completely contained in the local view, by $p^{h-local}$ and the remnant of the path, $(\gamma_p^h, u_t, u_{t+1}, \ldots, u_n = j)$, by $p^{\overline{h-local}}$. Similarly, for a path-set $P \in 2^{P_{hj}^G}$, we define $P^{h-local} = \{p^{h-local} : \ p \in P\}$ and $P^{\overline{h-local}} = \{p^{\overline{h-local}} : p \in P\}$.

C. Local Pruning

Local pruning [2] in mobile/dynamic networks is an interesting graph optimization problem. These pruning algorithms

make use of the local neighborhood information that is provided by neighbor discovery protocols [7]. From this local neighborhood information they select a subset of the topology information that is broadcast to the network. This subset is chosen so that the resulting pruned graph preserves some properties of the original graph. The non-triviality in these problems is establishing a relation between the local and pruned global graph. We will make these notions more rigorous. In [21], we extended the notion of local and global views introduced in [25] to encompass labeled dynamic graphs. We summarize these extensions.

We assume that every host has a neighbor discovery module [9], [8], [18]. It discovers its local neighborhood information using periodic *HELLO* messages. The *HELLO* message from each host contains both the communication adjacency and the arc labels for all of its $(k-1)$-hop neighbors ($k \geq 2$). Every host, $h \in V$, exchanges these *HELLO* messages with their neighbors. Consequently, h has access to the dynamic labeled graph G_h^{local} (since the *HELLO* messages from a neighbor j contain its neighborhood information, N_j^{k-1} and the arc labels between them). For instance, in OLSR $k = 2$ because every host exchanges its one-hop link state information with its neighbors. This notion is formally abstracted as the *local view*:

Definition At every host station $h \in V$, the *local view* is the labeled subgraph G_h^{local}, with a neighborhood size k, that is exposed by the neighbor discovery mechanism at h.

In local pruning algorithms, the host $h \in V$, which has discovered the labeled graph G_h^{local}, chooses a subset of its incident links in G_h^{local}, which we call the *pruned arc set*, and broadcasts this arc set to the entire network. For h, the subset of its incident links is is $\Omega_h^{G_h^{local}}$, which is also Ω_h^G because the local neighborhood is completely exposed (for $k \geq 2$). The set of *pruning policies* at host h is the set of functions

$$F_h^{prune} = \{f : G_h^{local} \rightarrow 2^{\Omega_h^G}\},$$

where $2^{\Omega_h^G}$ is the power-set of Ω_h^G (set of of all subsets of Ω_h^G).

For a given pruning policy $f_h \in F_h^{prune}$ at $h \in V$, the pruned arc set is denoted by $\delta_h = f_h(G_h^{local})$. From henceforth, f_h and δ_h will represent the pruning policy and pruned arc set at host h respectively. δ_h is, then, broadcast along with the corresponding arc labels network-wide. If the subset δ_h is small compared to Ω_h^G, then the broadcast information rate is significantly reduced. This controlled flooding (of pruned link states) reduces the broadcast storm. The arc set that is broadcast, by all the hosts, is given by $E^{broadcast} = \cup_{h \in V} \delta_h$, and this induces a labeled subgraph $G^{broadcast}$, which we call the *broadcast view*. The local view and the broadcast view together create, what we call, a *global view* of the graph at h.

Definition At every host station $h \in V$, the *global view* G_h^{global} is the labeled graph union $G_h^{local} \cup G^{broadcast}$, where G_h^{global} and $G^{broadcast}$ are exposed by some neighbor discovery and link state broadcast mechanisms respectively.

As mentioned above, the goal of any local pruning algorithm is to prune effectively (maximally) while preserving some desired properties in the pruned graph, i.e., $|\delta_h|$ must be minimal and G_h^{global} must preserve some desired property of the G.

For the pruning problem of interest – pruning for the SAPP solutions – we require the G_h^{global} to preserve the solutions to the SAPP defined on G (Equation (6)). More precisely, we require at h, G_h^{global} to preserve the solution from h to $j \in V$. We define the SAPP-solution preserving property at host h, π_h^{global}. A subgraph $G' \subseteq G$ is said to have the property π_h^{global} if and only if

$$x_{hj}^{G'} = x_{hj}^{G},\ j \in V,$$

where $x_{hj}^{G'}$ is the SAPP-solution restricted to the labeled graph G'. Note that the pruning problem is defined at every host, $h \in V$. Let Π_h^{global} denote the set of all subgraphs of G for which the property π_h^{global} holds, i.e.,

$$\Pi_h^{global} = \{G' \subseteq G :\ \pi_h^{global} \text{ holds for } G'\}.$$

The desired SAPP-solution preserving property for the pruned graph can be then expressed as a constraint

$$G_h^{global} \in \Pi_h^{global}. \tag{8}$$

Note that in this formulation, the constraint $G_h^{global} \in \Pi_h^{global}$ is a global constraint, i.e., the constraint is not limited to the local view; the constraint depends on the global properties of G.

The corresponding pruning problem is an interesting multi-agent optimization problem where the objective function (finding a minimal pruned arc set) for each agent (host) depends only on local neighborhood information (local view). However, the agents (hosts) together must satisfy a global constraint (the global view must preserve the SAPP-solutions). This is non-trivial because the global constraint involves the global view, while the hosts have access to strictly their local view. In general, this global constraint cannot be expressed in terms of the local view. However, we will show that under certain conditions, the global constraint, on the global view, can be reduced to a local constraint, on the local view.

Clearly, there are many local constraints that guarantee the global constraint. For instance, naively preserving all paths locally, or with a little more sophistication, preserving all the solution path-sets to all vertices in the exclusive neighborhood is sufficient local constraint. However, since the objective function is to minimize the pruned arc set, we need a sufficient condition that also requires the minimal number of paths to be preserved. In the next section, we provide such a sufficient local condition that satisfies the global constraint.

D. Strict Inflatory Condition, Sufficiency and Loop-Freedom

Since the local pruning policies of interest at host $h \in V$ are given by the functions $f_h \in F_h^{prune}$, it is natural to establish the conditions that these functions must satisfy.

Consider a local property π_h^{local} at $h \in V$. The property is local in the sense, it is defined only for the local view at h. Given the local view G_h^{local}, the property π_h^{local} is said to hold for a pruning function $f_h \in F_h^{prune}$, if for all $j \in L_h$ there exists a solution path-set $P_{sol} \in \mathcal{P}_{hj}^{h-local^*}$ such that $\forall p \in P_{sol}$, $(h, \eta_p^h) \in \delta_h$. Let Π_h^{local} denote the subset of functions of F_h^{prune} for which π_h^{local} holds.

The local constraint $f_h \in \Pi_h^{local}$, in essence, requires that at least one solution path-set to every leaf vertex in the local view is covered by δ_h. We will illustrate, using an example, the intuition behind this condition. Consider a directed labeled linear graph shown in Figure 6. Let the size of the neighborhood be $k = 2$. We will illustrate that if the local pruning condition is not satisfied, then arc (h_3, h_4) is not chosen for broadcast. Clearly, for this graph, only host h_3 is responsible of selecting (h_3, h_4) (by the virtue of the local pruning policy definition in Subsection III-C). Let us consider the pruning policy at h_3. Here, $Ł_{h_3} = \{h_5\}$, and (h_3, h_4, h_5) is the only path, and therefore, a solution path from h_3 to h_5. If $(h_3, h_4) \notin \delta_{h_3}$, then $(h_3, h_4) \notin G^{broadcast}$. Since $(h_3, h_4) \notin G_{h_1}^{local}$, $(h_3, h_4) \notin G_{h_1}^{global}$. Consequently, $G_{h_1}^{global}$ does not preserve the solution path from h_1 to h_4 and h_5. The same argument can be applied to any leaf vertex of any local view. However, the condition $f_h \in \Pi_h^{local}$ is not a necessary condition. As the example suggests, for $G_{h_1}^{global} \in \Pi_{h_1}^{global}$, the only necessary condition at h_3 is to preserve the arc (h_3, h_4). The local condition π_h^{local} only implies this necessary condition. Although there may be other means to achieve this necessary condition, we choose to work with π_h^{local} because of the ease of implementation, which will be illustrated in the forthcoming sections.

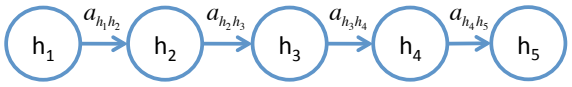

Fig. 6. Example line graph illustrating the sufficient condition

The condition $f_h \in \Pi_h^{local}$ is not sufficient in a distributed setting to ensure $G_h^{global} \in \Pi_h^{global}$, since it does not guarantee *loop-freedom*. This is a well-known problem for distributed routing protocols [17]: Loops typically occur in distributed graph algorithms when tie-breaking mechanisms are not employed. Using an example, we illustrate that a similar problem is likely to occur in distributed local pruning without tie-breaking. Figure 7 illustrates a scenario where the distributed pruning leads to loops. The figure shows a labeled directed graph. Let the size of the neighborhood be $k = 2$. The leaf sets are $L_{h_1} = \{h_3\}$, $L_{h_2} = \{h_4\}$, $L_{h_3} = \{h_4\}$, $L_{h_4} = \emptyset$ and $L_{h_5} = \{h_4\}$. Let $a_{h_1h_2} = a_{h_2h_1} = ①$, and $a_{h_1h_5} \otimes a_{h_5h_3} = a_{h_2h_3} > a_{h_2h_3}$. Then the set of *h-local solution path-sets* are $\mathcal{P}_{h_1h_3}^{h-local^*} = \{\{(h_1, h_2, h_3)\}, \{(h_1, h_5, h_3)\}\}$, $\mathcal{P}_{h_2h_4}^{h-local^*} = \{\{(h_2, h_3, h_4)\}, \{(h_2, h_1, h_5, h_3, h_4)\}\}$, $\mathcal{P}_{h_3h_4}^{h-local^*} = \{\{(h_3, h_4)\}\}$ and $\mathcal{P}_{h_5h_4}^{h-local^*} = \{\{(h_5, h_3, h_4)\}\}$. Note that all solution path-sets in the example are singletons, i.e., the solutions correspond to a single path, similar to the shortest path example discussed in Subsection II-B. The pruned arc sets $\delta_{h_1} = \{(h_1, h_2)\}$, $\delta_{h_2} = \{(h_2, h_1)\}$, $\delta_{h_3} = \{(h_3, h_4)\}$ and $\delta_{h_5} = \{(h_5, h_3)\}$ satisfy the local condition $f_h \in \pi_h^{local}$ at each host, but the resulting $G^{broadcast}$ is disconnected! Thus the local conditions do not guarantee loop-freedom and the solutions are not preserved.

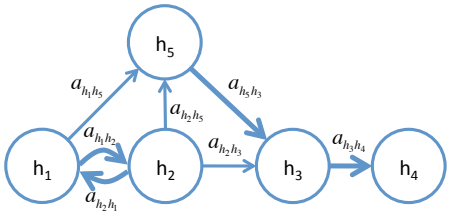

Fig. 7. Illustration of loops in pruning. Shows an example labeled directed graph. The arcs of $G^{broadcast}$ are shown with thick lines.

The example of Figure 7 suggests a sufficient condition, which is called strict monotonicity condition [20]:

$$a, b \in S,\ a \otimes b < a \text{ and } a \otimes b < b.$$

This property is also referred to as a deflatory by other authors [12]. We will show that under the strict monotonicity assumption, the local condition $f_h \in \Pi_h^{local}$ becomes sufficient.

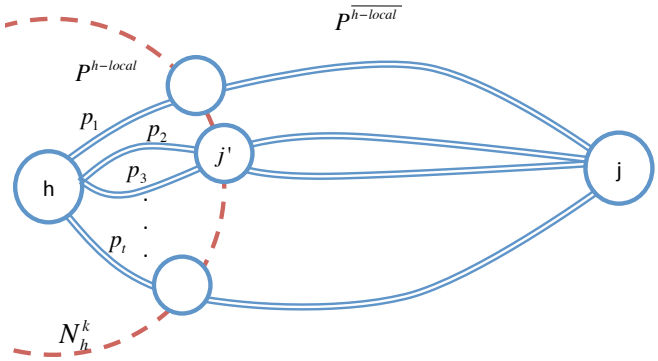

Fig. 8. Generalized Bellman's Optimality with path-sets

Consider the illustration shown in Figure 8. Let $P \in \mathcal{P}_{hj}^{G^*}$ be any solution path-set from h to any $j \notin N_h^k$. The path set clearly intersects with the boundary (leaf vertices). For each $j' \in \Gamma_P^h$, we define the constituent solution paths

$$P_{j'}^{h-local} = \{p \in P^{h-local} : \gamma_p^h = j'\}.$$

In general, $P_{j'}^{h-local}$ need not be singleton, as shown in the Figure 8 - paths p_2, p_3. Then the following lemma establishes Bellman's optimality principle for solution path-sets for a general idempotent SAPP.

Lemma 3.1: $\oplus_{p \in P_{j'}^{h-local}} w(p)$ is one solution to $x_{hj'}^{h-local}$.

Proof: We will assume otherwise and derive a contradiction. Let us assume that any solution $x_{hj'}^{h-local} >$

$\oplus_{p \in P^{h-local}_{j'}} w(p)$. Consider the solution path-set corresponding to such a solution P^{rep}. If we replace the $P^{h-local}_{j'}$ with P^{rep} we obtain a dominating solution and this contradicts that P is a solution to x^G_{hj}. ∎

Theorem 3.2: Under the strict monotocity assumption, if $h \in V$, $f_h \in \Pi^{local}_h$, then $G^{global}_h \in \Pi^{global}_h$.

Proof: Suppose a solution path-set to j is contained in G^{local}_h, then the proof is trivial. Consider the other case: the solution path-set to j is not contained in G^{local}_h. From Lemma 3.1, we know for any solution path-set, there is a local solution to every gateway vertex. Since the pruning policy $f_h \in \Pi^{local}_h$ ensures that such paths are preserved locally, they are also preserved globally. The strict monotonicity property ensures global loop-freedom. ∎

E. Optimal Pruning as a Local Set-Cover Problem

With the notation introduced in the previous sections, the local pruning problem for preserving the SAPP-solution can be expressed mathematically as follows.

$$\min_{f_h \in \Pi^{local}_h} |\Omega_h| \tag{9}$$

Attempting to list out all feasible pruning policies $f_h \in \Pi^{local}_h$, in general, is computationally intractable. We will show that this problem can be reduced to a set-cover problem. To formulate this set-cover problem, we introduce further notation. Let $\zeta_h : 2^{\partial N^1_h} \to 2^{L_h}$ denote the covering function: for $S \subseteq \partial N^1_h$ and

$$\begin{aligned}\zeta_h(S) = \quad & \{j \in L_h : \exists P \in \mathcal{P}^{h-local^*}_{hj} \text{ for each } x^{h-local}_{hj} \\ & \text{such that } H^h_P \subseteq S\}.\end{aligned}$$

This function ζ_h can be computed locally efficiently using generalized Gauss elimination [11], [19]. Then the set-cover problem is

$$\begin{aligned}\min_{\Delta \in 2^{\partial N^1_h}} \quad & |\Delta| \\ \text{subject to} \quad & \cup_{S \subset \Delta} \zeta_h(S) = L_h.\end{aligned} \tag{10}$$

Theorem 3.3: For any minimizer Δ^* of the problem in Equation (10), $\{(h,i) : i \in \Delta^*\}$ solves the minimal pruning problem of Equation (9).

Proof: Since $\cup_{S \in \Delta} \zeta_h(i) = \partial L_h$, $f_h(G^{local}_h) = \{(h,i) : i \in \Delta^*\} \in \Pi^{local}_h$. ∎

For a more detailed illustration of the proofs and algorithms to construct ζ for the $(\hat{\mathbb{Z}}_+, \min, +)$ semiring, see [22].

IV. Conclusion

We presented an alternative method to solve for SAPP solutions on a dynamic graphs. The algorithm makes use of broadcasting to recompute the SAPP solution when the network state changes. To reduce the associated broadcast storm, we propose a selective broadcasting solution and prove that the pruned graph preserves the solution to the original SAPP.

References

[1] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, 1993.

[2] Williams B. and Camp T. Comparison of broadcasting techniques for mobile ad-hoc networks. In *Proceedings of the ACM International Symposium on Mobile Ad-Hoc Networking and Computing (MOBIHOC)*, 2002.

[3] B.A.Davey and H.A. Priestley. *Introduction to Lattices and Order*. Cambridge University Press, 1990.

[4] J. S. Baras and G. Theodorakopoulos. *Path Problems in Networks*. Synthesis Lectures on Communication Networks. Morgan and Claypool, 2010.

[5] D. Berksekas and R. Gallager. *Data Networks*. Prentice Hall, 1992.

[6] Bela Bollobas. *Modern Graph Theory*. Springer, 1998.

[7] T. Clausen, C. Dearlove, and J. Dean. Mobile ad hoc network (manet) neighborhood discovery protocol (nhdp). Draft-IETF, October 2009.

[8] T. Clausen, C. Dearlove, and P. Jacquet. The optimized link state routing protocol version 2. Draft-IETF, September 2009.

[9] T. Clausen and P. Jacquet. Optimized link state routing protocol (olsr). RFC, Oct 2003.

[10] L. Eschenauer, V. Gligor, and J. S. Baras. On trust establishment in mobile ad-hoc networks. In *Security Protocols Workshop*, 2004.

[11] M. Gondran and M. Minoux. *Graphs, Dioids and Semirings - New Models and Algorithms*. Springer, 2008.

[12] T. G. Griffin. The stratified shortest-paths problem. In *COMNETS*, 2010.

[13] P. Jacquet, A. Laouiti, P. Minet, and L. Viennot. Performance analysis of olsr multipoint relay flooding in two ad hoc wireless network models. Technical report, INRIA, September 2001.

[14] F.R. Kschischang, B.J. Frey, and H. Loeliger. Factor graphs and sum-product algorithm. *IEEE Transactions on Information Theory*, 46:489–519, 2001.

[15] A. Laouiti, A. Qayyum, and L. Viennot. Multipoint relaying: An efficient technique for flooding in mobile wireless networks. In *35th Annual Hawaii International Conference on System Sciences (HICSS'2001)*. IEEE Computer Society, 2001.

[16] R.J. McEliece and S.M. Aji. The generalized distributive law. *IEEE Transactions on Information Theory*, 46(2):325–343, 2000.

[17] C. E. Perkins. *Ad hoc networking*. Addison Wesley, 2001.

[18] H. Rogge, E. Baccelli, and A. Kaplan. Packet sequence number based etx metric for mobile ad hoc networks. IETF Draft, December 2009.

[19] G. Rote. Path problems in graphs. In *Computing Supplementum*, volume 7, pages 155–189, 1990.

[20] J. L. Sobrinho. Algebra and algorithms for qos path computation and hop-by-hop routing in the internet. *IEEE/ACM Transactions on Networking*, 2002.

[21] K. K. Somasundaram and J. S. Baras. Semiring pruning for information dissemination in mobile ad hoc networks. In *Workshop on Applications of Graph Theory in Wireless Ad hoc Networks and Sensor Networks*, 2009.

[22] K. K. Somasundaram, J. S. Baras, K. Jain, and V. Tabatabaee. Distributed topology control for stable path routing in multi-hop wireless networks. Technical report, Institute for Systems Research, 2010.

[23] G. Theodorakopoulos and J.S. Baras. On trust models and trust evaluation metrics for ad hoc networks. *IEEE Journal on Selected Areas in Communication*, 24(2):318–328, 2006.

[24] S. Verdu and V. Poor. Abstract dynamic programming models under commutativiy conditions. *SIAM Journal on Control and Optimization*, 25(4):990–1006, 1987.

[25] J. Wu and F. Dai. A generic distirbuted broadcast scheme in ad hoc wireless networks. *IEEE Transactions of Computers*, 53(10):1343–1354, 2004.