International Journal of Distributed Sensor Networks, 4: 223–246, 2008

ISSN: 1550-1329 print / 1550-1477 online
DOI: 10.1080/15501320701260063

Taylor & Francis
Taylor & Francis Group

A Flexible Stochastic Automaton-Based Algorithm for Network Self-Partitioning

YAN WAN[1], SANDIP ROY[1], ALI SABERI[1], and BERNARD LESIEUTRE[2]

[1]The first three authors are with the Washington State University, Pullman, WA
[2]The fourth author is with the Lawrence Berkeley National Laboratory, Berkeley, CA

This article proposes a flexible and distributed stochastic automaton-based network partitioning algorithm that is capable of finding the optimal k-way partition with respect to a broad range of cost functions, and given various constraints, in directed and weighted graphs. Specifically, we motivate the distributed partitioning (self-partitioning) problem, introduce the stochastic automaton-based partitioning algorithm, and show that the algorithm finds the optimal partition with probability 1 for a large class of partitioning tasks. Also, a discussion of why the algorithm can be expected to find good partitions quickly is included, and its performance is further illustrated through examples. Finally, applications to mobile/sensor classification in ad hoc networks, fault-isolation in electric power systems, and control of autonomous vehicle teams are pursued in detail.

Keywords Partitioning; Distributed Partitioning; Islanding; Stochastic Automata

1. Introduction

Networks of communicating agents, including sensor networks and autonomous-vehicle teams, require distributed algorithms for a variety of tasks, including data communication/routing, estimation/agreement, and pattern-formation control, among others (see [1] and [2] for interesting overviews). In this article, we put forth the perspective that algorithms for network *self-partitioning* or *self-classification*, i.e. algorithms using which of the network's nodes can form groups so as to minimize cost while communicating in a distributed manner, are needed. We further contend that partitioning algorithms for these communicating-agent networks, whether distributed or centralized, must be flexible, in the sense that the algorithms should permit the minimization of complex and varied cost measures. With these motivations in mind, we develop a flexible algorithm for network partitioning and self-partitioning using a stochastic automaton known as the influence model [3].

Distributed algorithms for self-partitioning may be valuable for various sensor networking and autonomous vehicle control applications. Consider the following:

- A group of autonomous vehicles in the field may need to self-assemble into multiple teams, in order to simultaneously complete multiple control tasks, e.g. search-and-destroy tasks (see e.g. [4,5] for formation-control algorithms for autonomous vehicles). The vehicles should be grouped (partitioned) in such a manner that the

Address correspondence to Sandip Roy, School of EECS, Washington State University, P.O. Box 642752, Pullman, WA, 99164–2752. E-mail: sroy@eecs.wsu.edu

self-assembly takes little time, and the robots in each group can easily communicate with each other.

- Sensors in an ad hoc network must choose one of several base stations for communication, so as to minimize the power required for multicasting as well as the latency of transmission from the sensors back to the base (see [10] for an overview of multicasting in ad hoc networks). Further, the sensors may need to classify themselves in such a manner that all the sensors associated with a particular base station can communicate among themselves, and further the network can tolerate any single failure in a communication link.
- Weakly-connected subnetworks within a computer network may need to be identified, so as to isolate a spreading computer virus.

In each of these tasks, the nodes in a network must be partitioned so as to minimize the cost. Further, for a variety of reasons (including security concerns, need for low-power and hence localized communication, and possibly for topological changes that are not known by a central authority), we may require a distributed algorithm for these partitioning tasks.

While there is a wide literature on graph partitioning (which derives primarily from parallel-processing applications, see [6] for an overview), the partitioning tasks for the communicating-agent networks described above are novel in several respects:

1) As motivated above, the algorithms used often must be distributed, e.g. because of the high power cost of communicating with a central agent or the need for security. For the same reasons, sparsity of communication in use of the algorithm is also often a must. Further, algorithms that are scalable, i.e. ones in which the computational cost for each agent grows in a reasonable manner with the network size, are needed; distributed algorithms can permit scalability.
2) The cost to be minimized is often a complex or multivariate one (e.g., for sensor network applications, delay, power dissipation, and reliability may each play a role in the cost), and varies from one application to another. Thus, we require algorithms that are flexible with respect to the cost minimized. This contrasts with the bulk of the literature on partitioning [7, 8, 9], in which algorithms are designed for a particular cost, typically a min-cut cost or a min-cut cost with partition-size constraints[1].
3) Communicating-agent networks are commonly subject to topological changes, for instance due to the addition of an agent or the failure of a particular communication link. Thus, partitions of the network may need to be adapted rapidly and frequently, ideally with minimal communication.

These novel features have motivated us to develop a distributed and flexible algorithm for network partitioning/classification.

Specifically, we introduce an algorithm for network self-partitioning (i.e., distributed partitioning) that is based on a stochastic automaton known as the influence model. The influence model can be viewed as a network of discrete-time, finite-state Markov chains, which interact in the sense that the current status (state) of each site (chain) probabilistically influences the future statuses of its neighboring sites [3]. The basic premise for using the influence model (specifically the copying influence model for partitioning graphs is that groups of sites in the model that are separated by weak influences tend to have a different statuses, while sites interconnected by strong influences tend to form a

[1]Bisection, in which the minimum cut that breaks the network into multiple equal-sized partitions is found, is of particular interest in the partitioning community [6].

cluster with a common status. Therefore, by associating influences with edge weights in a graph, allowing the influence model to run for some time, and then examining the statuses, we can identify a good partition quickly with respect to many typical cost functions. At the same time, the algorithm randomly searches through many potential partitions, and hence holds promise for minimizing multi-objective and complex cost functions. The technique is distributed in that each site only needs to communicate with graphical neighbors to determine its own partition.

This algorithm for network partitioning builds on our earlier work on a control-theoretic approach to distributed decision-making or agreement [11] (see also [12, 13] for other control-theoretic approaches to agreement and [14] for a study of sensor fusion that addresses/motivates distributed detection/decision-making). In the context of decision-making, we used the influence model to reach consensus among nodes in a manner that reflected their initial divergent opinions about a topic of interest; here, the influence model does not generate one opinion, but instead finds low cost cuts as boundaries between multiple opinions or statuses. Also of interest to us, stochastic automata have been used as tools for routing in sensor networks (e.g. [15]), and have been used as *gossip protocols* for information dissemination in ad hoc networks [16]. There is also a much broader literature on the analysis of stochastic automata, and their application to modeling and computational tasks. This literature is outside the scope of this paper; see [17, 18] for general introductions.

While our primary motivation is self-partitioning, our studies suggest that the influence model-based algorithm is also valuable for centralized partitioning problems in which multiple complicated costs must be minimized, or in which costs are implicitly found through a simulation. For instance, motivated by fault-tolerance and fault-isolation applications (e.g. [19]), we have applied the algorithm to partition an electric power system so as to minimize both a line-weight and a power-imbalance cost in isolating two generators from each other. We shall briefly explore this centralized application in the article.

The remainder of this article is organized as follows. Section 2 poses the graph partitioning problem in quite a general way, in the process overviewing commonly-studied partitioning problems and standard algorithms for solving them. Section 3 briefly reviews the influence model, on which our partitioning algorithm is based. Section 4 describes the influence model-based distributed partitioning algorithm, in particular describing the mapping from the graph to the influence model, the distributed recursion used for partitioning, and centralized/distributed means for stopping the algorithm. In Section 5, we prove that the algorithm finds the optimal partition with certainty given certain weak graph-structural assumptions, and also discuss the performance of the algorithm. In Section 6, we pursue applications and give several illustrative examples, to better motivate our approach to partitioning and to further evaluate its performance.

2. Problem Statement

Since one of the features of our influence model-based partitioning algorithm is its flexibility, we begin by describing the partitioning (classification) problem in quite a general manner, but taking care to highlight sub-problems of particular interest. In the process, we give a brief review of the research on partitioning that is relevant to our development. We refer the reader to [6, 23] for thorough reviews of the partitioning literature.

Broadly, a k-way partitioning algorithm is concerned with classifying the vertices of a graph into k disjoint subsets. Specifically, let us consider a graph with (finite) vertex-set V that has cardinality n. We associate a positive *mass* m_υ with each vertex (node) $\upsilon \in V$. In addition to the vertices, our graph also comprises a set of positively-weighted, directed edges. That is, for each ordered pair of distinct vertices v_i, v_j, we associate a weight $w_{ij} \geq 0$, where $w_{ij} = 0$ indicates that a directed edge is not present while $w_{ij} > 0$ indicates a weighted edge.

The partitioning problem that we consider is to classify the n vertices into k disjoint, non-empty subsets so as to minimize a cost function, while possibly enforcing one or more constraints. The cost function and constraints are phrased in terms of the total masses of the subsets and the edge weights on cuts.

Formally, we define a k-way partition of a graph as a subdivision of the nodes of the graph into k disjoint, non-empty subsets (components) $S_1, \ldots, S_k$. We are interested in identifying a partition that minimizes a cost function

$$f(M(S_1), \ldots, M(S_k), W(S_1, S_2), \ldots, W(S_k, S_{k-1})),$$

where $M(S_i) \triangleq \sum_{i \in S_i} m_i$ is the mass of subset i, and $W(S_1, S_m) = \sum_{i \in S_i} \sum_{j \in S_j} w_{ij}$ is the size of the cut between subsets i and j. We seek to minimize the cost function over the class of partitions that, in general, satisfy a number of constraints of the following types:

- **Algebraic constraints**. These are of the form $g(M(S_1), \ldots, M(S_k), W(S_1, S_2), \ldots, W(S_k, S_{k-1})) = 0$.
- **Set inclusion constraints**. These have the form $v_i \in S_j$, i.e. particular vertices are constrained to lie in particular subsets. We often refer to a vertex that is constrained to lie in S_j as a reference vertex for subset j.

We use the notation $S_i^*, \ldots, S_k^*$ for a partition that minimizes the cost subject to the constraints, and refer to this partition as an optimal solution of the partitioning problem[2]. Our aim is to solve the partitioning problem in a distributed manner, i.e. so that only communications along the edges of the graph are needed in finding the optimal partition.

A variety of partitioning problems considered in the literature are examples of the problem described above. It is worth our while to briefly discuss these problems and associated literature, focusing in particular on *Partitioning with Reference Nodes* (Item 4 below) because of its relevance to our applications. Commonly-considered partitioning problems include the following:

1) The *Min-cut Problem* is a k-way partitioning problem in which the subsets are chosen to minimize the total strength of the cuts between the components, with no algebraic or set inclusion constraints enforced. That is, the components are chosen to minimize the unconstrained cost function $f = \sum_{i=1}^{k} \sum_{\substack{j=1 \\ j \neq i}}^{k} W(S_i, S_j)$. The min-cut problem is well-known to admit a polynomial-time solution, and several search algorithms have been developed (see e.g. [24]). Spectral methods ([27, 28, 26]) and stochastic algorithms

[2]We can in fact allow a far more general cost function, e.g. one that depends on dynamics defined on the graph. We adopt this form here for clarity in our explanation of why our algorithm is expected to work well.

based on coalescing strongly-connected nodes [25] have also long been used to find min-cuts.

2) The *Bisection Problem* is a 2-way partitioning problem, in which the subsets are chosen to minimize the strength of the cut between them, subject to the constraint that the masses of each subset are equal. That is, the cost function $f = W(S_1, S_2) + W(S_2, S_1)$ is minimized, subject to the algebraic constraint $g(M(S_1), M(S_2)) = M(S_2) - M(S_1) = 0$. Very often, the masses of the vertices are assumed to be all unity, so that the constraint reduces to enforcing that subsets have equal cardinality. The bisection problem finds its major application in parallel computing [6], where equally distributed workloads are desired. Bisection is a difficult (NP–hard) problem and has a wide literature. Specifically, classical bisection algorithms fall into four categories:
    1) geometric partitioning algorithms based on coordinate information [29, 30]
    2) greedy search algorithms like the Kernighan-Lin algorithm [8]
    3) spectral methods (methods based on eigenvalue/eigenvector structure of matrices associated with the network graph) [7, 31], and
    4) stochastic algorithms including genetic algorithms [32, 33] and simulated annealing [9, 34].
3) In some applications, the exact mass constraint of bisection is not needed, yet it is useful to have subsets of roughtly equal or mass size. For such applications, a *mass-weighted min-cut problem* is often solved. In particular, a cost function $f = \frac{\sum_i \sum_{j \neq i} W(S_i,S_j)}{M(S_1)...M(S_k)}$ is minimized, assuming no algebraic constraints. The form of this cost function has been studied in [36] and is deeply connected to the convergence rate of the linear dynamics defined on the graph. We note that the term *ratio cut* has sometimes been used in the literature for these weighted problems.
4) Sometimes, an application dictates that one of the above problems (or another k-way partitioning problem with a different cost) must be solved, subject to set-inclusion constraints, i.e. subject to constraints that certain reference nodes are contained in each component. We refer to such problems as *k-way partitioning problems with reference nodes*. Partitioning with reference nodes is of interest to us for several reasons:
    1) problems in several distributed applications—for instance, the problem of grouping sensor nodes with base stations for multicasting—have this form
    2) these problems are known to be NP-hard for $k \geq 3$ and hence still require development of good algorithms [22], and
    3) our algorithm is naturally designed to address this problem and hence gives fast solutions to the problem.

There is a wide literature on algorithms for solving these partitioning problems (see, e.g., the review articles [6, 23, 29]). A thorough review of this literature is far beyond the scope of this article, but lets us attempt to briefly summarize this work with the aim of delineating our approach from that in the literature. Most of the current partitioning algorithms are aimed at solving a particular problem (perhaps most commonly the bisection problem) in a centralized manner. For example, spectral methods have been used for min-cut and bisection problems, while SA, GA, and K-L are designed specifically for bisection [32, 9, 8]. In constrast to these methods, our applications motivate us to seek an algorithm that can find the optimal partition for a range of cost functions, even if perhaps at slightly higher computational complexity.

The algorithms in the literature that are stochastic (e.g., GA and SA) are of interest to us [6], since our algorithm is also stochastic. Very broadly, our algorithm is similar to these in that it searches randomly through plausible partitions, using the uncertain generation to seek more optimal partitions. However, our algorithm is significantly different from those in the literature, in that it does not react to the cost of the current partition: instead, its update is based solely on the graph topology. This topological approach has the advantage of permitting flexibility in the optimized cost, and (as we shall show) of allowing identification of the minimum-cost solution with probability 1.

Perhaps most significantly, we contribute to this broad literature by developing an algorithm for distributed or self-partitioning, i.e. an algorithm in which agents associated with graph vertices can decide their optimal partitions based solely on communication with neighboring agents. To the best of our knowledge, there have been no other algorithms developed that achieve partitioning without any global perspective at all in the graph.

3. The Copying Influence Model: A Brief Review

Our algorithm for partitioning is based on evolving a stochastic automaton model. Specifically, we map a graph to a dynamic *stochastic network model*—a model in which values or statuses associated with network nodes are updated based on interactions with neighboring nodes. The statuses associated with the nodes form patterns as they evolve with time; these patterns turn out to identify good partitions of the graph. Since the automaton is updated only through interactions of nodes with graphical neighbors, it permits partitioning in a decentralized manner. The automaton that we use for partitioning is an instance of the *Influence model* [3], a stochastic network automaton with a special quasi-linear structure. In this section, we very briefly review the influence model. We refer the reader to [3] for a much more detailed development.

An influence model is a network of n nodes or vertices or *sites*, each of which takes one of a finite number of possible *statuses* at each discrete time-step. We use the notation $s_i[k]$ for the status of site i at time k. We refer to a snapshot of the statuses all the sites at time k as the *state* of the model at time k. The model is updated at each time-step according to the following two stages:

1) Each site i picks a site j as its *determining site* with probability d_{ij}.
2) Site i's next-status is then determined probabilistically based on the current status of the determining site j. That is, the next status is generated according to a probability vector, which is parameterized by the current status of the determining site.

We shall only be concerned with a special case of the influence model called the *copying influence model*, in which each site takes on the same number k of statuses (labeled $1,\ldots, k$ w.l.o.g.), and furthermore each site simply copy's the status of its determining site at each time step. To reiterate, at each time-step in the copying influence model, each site i picks a neighbor j with probability d_{ij} and copy's the current status of that neighbor.

The influence model and copying influence model are compelling as modeling and algorithmic tools because they have a special quasi-linear structure. In general, for stochastic network models such as the influence model, we note that the statuses of all sites together are updated in a Markovian fashion, and hence the joint status of all sites are governed by a very large "master" Markov chain with k^n states. However, for the influence model, the status probabilities of individual sites and small groups of sites can in fact be found using low-order recursions. For instance, the probability of site i taking

status m at time $k + 1$ in the copying influence model can be tracked using the following low-order recursion:

$$P(s_i[k+1] = m) = \sum_j P\big(s_j[k] = m\big) d_{ij} \tag{1}$$

Furthermore, the special structure of the influence model permits us to identify qualitative features of the master Markov chain based on the low-order recursions. These special tractabilities of the influence model make it possible to characterize the performance of algorithms built using the model, such as the algorithm developed here.

4. Algorithm Description

We can use the copying influence model as a tool for solving the partitioning problem described in Section 2 under rather broad conditions. Furthermore, since the influence model update only requires interaction among graphical neighbors (in a sense that will be made precise shortly), the algorithm is essentially decentralized (though a bit further effort is needed to *stop* the algorithm in a decentralized manner). The combination of flexibility and decentralization makes the influence model-based algorithm applicable to a range of partitioning tasks, including those discussed in the introduction. In this section, we describe the influence model-based partitioning algorithm. In the next section, we prove that the algorithm works (finds the optimal solution with certainty) under broad conditions. Here, we first outline the algorithm, and then fill in the details.

1) **Mapping.** We map the graph to a copying influence model, by associating large influences with strong interconnections in the graph, and weak influences with weak interconnections. We note that we can permit asymmetric interconnection strengths.
2) **Initialization and Recursion.** We choose the initial state for the copying influence model. Here, the status of each site identifies the subset of the corresponding node in the graph. The statuses of the sites are updated recursively according to the developed copying influence model, and hence a sequence of possible partitions of the graph are generated. We note that this is a distributed computation, in that each site updates its status using only local information (i.e. information from graphical neighbors). Thus, in cases where a group of nodes in a real distributed system must self-partition, the influence model recursion can be implemented using localized communications between agents in the network. In presenting and analyzing the recursion, we find it convenient to first consider the case of partitioning with reference nodes[3], and then address the partitioning problems without reference nodes.
3) **Stopping.** The recursion is terminated based on cost evaluations for a centralized algorithm and by decreasing influence model probabilities in the decentralized case. The statuses of the influence model at the stopping time specify the chosen partition.

4.1. Mapping

We map the graph to a copying influence model with k possible statuses, with the motivation that we can identify a sequence of partitions of the graph by updating the influence model.

[3]For notational convenience, we focus on the case where there is one reference node per component, but our development can straightforwardly be generalized to cases where the number of partitions is different from the number of references.

That is, our algorithm classifies (partitions) the vertices in the graph according to the statuses of the corresponding influence model sites at each time-step of the recursion. The first step toward building this partitioning algorithm is to map the graph to a copying influence model, in such a manner that the copying probabilities in the influence model reflect the branch weights. In particular, we associate an influence model site with each vertex in the graph. We then choose the copying probabilities (influences) as

$$d_{ij} = \begin{cases} \frac{\Delta w_{ji}}{mi}, & i \neq j; \\ 1 - \Delta \sum_{l} \frac{w_{li}}{mi}, & i = j, \end{cases} \tag{2}$$

where Δ is chosen such that $\Delta \leq \frac{1}{\max_i \sum_j \frac{w_{ji}}{m_i}}$. Thus, large weights are associated with large influences, and small weights are associated with small influences; moreover, a large mass (inertia) m_i incurs small influence from other sites on site i (and large influence from itself), and a small mass m_i incurs large influence from other sites on site i.

In addition to the above direct interpretation, we can also give a linear systems-based interpretation for the mapping. In particular, we can show that the status-probability recursion (Equation 1) of the developed influence model is a discretized version of a certain linear differential equation defined on the graph. This linear system viewpoint is valuable because it indicates the close connection of our algorithm with some typical network dynamics, and because it can potentially permit an analytical connection of our algorithm with spectral partitioning algorithms. From the linear system viewpoint, the parameter Δ can be interpreted as the discretization step. More generally, Δ should be chosen large enough to achieve a fast convergence rate. We have specified the upper bound to guarantee that all the influence model parameters are valid.

In many decentralized and centralized applications, we envision this mapping stage as being done *a priori* by a centralized authority, even when the partitioning itself must be done in a decentralized manner. For instance, when new sensors are added to an existing network, the network designer can perhaps pre-program information about the communication topology and strengths of interactions between the sensors. However, it is worth noting that the mapping to the mapping to the influence model is in fact inherently decentralized (i.e., an agent associated with vertex i in the graph can compute the weights d_{ij} from the vertex's mass and the weights of edges to neighbors) except in one sense: the scaling parameter Δ is a global one. Noticing that the maximum allowed value for Δ depends on the total weights of edges out of nodes and node masses, we note that Δ can often be selected *a priori* based on some generic knowledge of the graph topology (for instance, knowledge of the maximum connectivity of any single node), when decentralized mapping is also required.

4.2. Initialization and Recursion

Let us first develop an algorithm for k-way partitioning with reference nodes (specifically, with one reference node per partition). For the problem of k-way partitioning with reference nodes, we fix the k *reference sites* (the sites in the influence model corresponding to the reference nodes) with distinct statuses from 0 to $k - 1$, and choose the initial statuses of other sites arbitrarily. Here, in order to fix the statuses of the reference sites, we need to make a slight modification to the influence model developed in Equation 2 such that reference site i always chooses itself as the determining site:

$$d_{ij} = \begin{cases} 0, & i \neq j; \\ 1, & i = j, \end{cases} \tag{3}$$

(In a distributed context, notice that we only require that the reference nodes know their own identities to implement this initialization).

To generate a good partition, we then update the copying influence model. The state at each time-step of the recursion identifies a partition of the graph: that is, we classify the nodes whose associated sites are in status i in subset S_i. We note that the partition identified at each time-step automatically satisfies the set inclusion constraints for k-way partitioning with reference nodes. We shall show that this recursion, which generates a random sequence of partitions, eventually finds (passes through) the optimal solution with probability 1 under broad assumptions, after sufficient time has passed. We note that the recursion is completely distributed, in the sense that each node can decide its own subset at each time-step solely from its graphical neighbors.

In practice, we must develop a methodology for stopping the algorithm. Below, we discuss distributed and centralized approaches for stopping. The stopping methodologies seek to select low-cost partitions, while checking possible algebraic constraints. We shall show that appropriate stopping criteria permit identification of the optimal solution under broad assumptions, with probability 1.

Conceptually, one might expect this partitioning algorithm to rapidly identify low-cost partitions, because strongly-connected sites in the influence model (sites that strongly influence each other) tend to adopt the same status through the influence model recursion[4], while weakly-connected sites do not influence each other and hence maintain different statuses. Recalling that the influence strengths reflect edge weights and node masses, we thus see that the partitions identified by the model typically have strongly-connected subsets with weak cuts between them. For many typical cost functions, the optimal partition comprises strongly-connected subsets with weak links, and hence we might expect the algorithm to find good cuts quickly.

For k-way partitioning (without reference nodes), we can find the optimum by solving the partitioning problem with reference nodes for all sets of distinct reference node selections, and optimizing over these. (Notice that we can actually keep one reference fixed, and search through possible placements of the other references.) This search is impractical when a large number of partitions are desired; we shall briefly consider alternatives in discussing future work. Most applications of interest to us have natural reference vertices, so we do not focus on the case without references.

A few further notes about the recursion are worthwhile:

- For simplicity of presentation, we have considered a discrete-time update, and hence a distributed implementation of the recursion in a network nominally requires a common clock for the agents in the network. However, we can equivalently use an update in which each site updates its status at random times (specifically, according to a Poisson arrival process); the recursion in this case is amenable to the same analyses as the recursion described here, and hence can be shown to achieve optimal partitioning.
- Regarding scalability in a distributed setting, we note that each agent in a network only needs to randomly select a neighbor and poll that neighbor at each time-step

[4]We refer the reader to our earlier work on *agreement* for further discussion about the dynamics of strongly-influencing sites[11].

to implement the recursion, so the processing/communication per time step does not increase with the size of the network. The total processing/communication cost thus scales with the duration of the recursion. In the next section, we give an argument that the scaling of the algorithm's duration with the size of the network is good compared to other partitioning algorithms in many cases.

- In some applications, we may already have one partition of a graph, and may wish to improve on this partition (with respect to a cost of interest) or to adapt the partition to changes in the graph. In such cases, we can speed up the recursion by initializing the influence model according to the original partition.

4.3. Stopping

Again, consider the k-way partitioning problem with reference nodes. (The adaptation to the general k-way problem is trivial.) For centralized problems, the global partition is known to a central agency at each recursion stage (time-step) and hence the cost can be evaluated and constraints can be checked. The minimum cost partition found by the algorithm can be stored. In this case, we propose to stop the updating after a waiting time, i.e. when the minimum-cost partition has not changed for a certain number of algorithm stages. This waiting time depends on the network structure and should be pre-calculated before the updating process. Generally speaking, the larger the size of the network, and the smaller the influences, the bigger the waiting time should be. We will show that a sufficiently long waiting time guarantees that the optimal solution is identified.

For distributed problems, it is unrealistic that a single agency can evaluate the global cost of a partition as in the centralized case, since each node only has available local information. A simple strategy in the distributed case is the blind one: the algorithm can be stopped after a finite number of time-steps, where this number is based on the convergence properties of influence models. A more complex strategy is to distributedly compute the cost at each stage using an agreement protocol (see, e.g. [12]).

Another clever strategy for distributed stopping is to use an influence model with state-dependent parameters. In particular, we progressively isolate (reduce the influence) between sites with different statuses after each update (and increase the self-influence correspondingly), until the influence model is disconnected (partitioned). More specifically, for each update, the (time-varying) influence $d_{ij}[k]$ is modified as follows:

- If $s_i[k] \neq s_j[k]$ and $d_{ij}[k] \geq \delta$, then $d_{ij}[k+1] = d_{ij}[k] - \delta\,(i \neq j)$ and $d_{ii}[\mathrm{k}+1] = d_{ii}[k] + \delta$;
- If $s_i[k] \neq s_j[k]$ and $d_{ij}[k] < \delta$, then $d_{ii}[k+1] = d_{ii}[k] + d_{ij}[k]$ and $d_{ij}[k+1] = 0\ (i \neq j)$;
- If $s_i[k] = s_j[k]$, then $d_{ij}[k+1]$ remains the same.

When this time-varying algorithm is used, we note that the statuses of sites converge asymptotically (see Fig. 1). This is because the influence model becomes disconnected, so that each partitioned component has only one injecting site and is guaranteed to reach consensus. Thus, a partition is found asymptotically. Furthermore, it is reasonable that this algorithm finds a good partition, since weak edges in the original graph tend to have different statuses at their ends in the influence model, and hence these edges are removed by the algorithm. We refer to this strategy as partitioning with adaptive stopping.

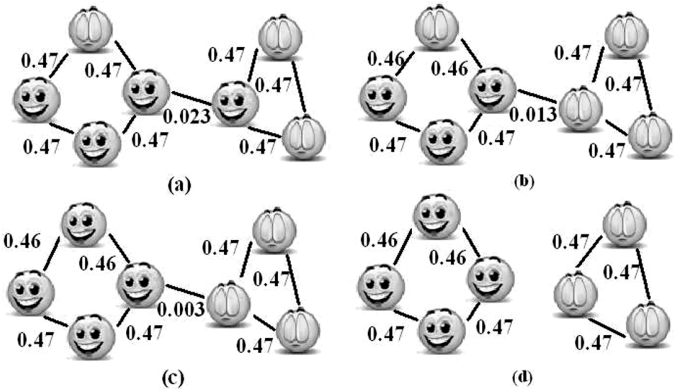


Figure 1. This diagram illustrates how a network partitions itself (based on the update of the time-varying copying influence model) in a totally distributed manner.

5. Algorithm Analysis

In this section, we prove that the influence model-based partitioning algorithm finds the optimal solution when either centralized or decentralized stopping is used. Specifically, we show that the influence model algorithm identifies the optimal solution with probability 1, given that the optimal solution satisfies certain broad connectivity conditions (which, as we show, is automatic for several common partitioning problems). The (quite-weak) connectivity conditions required of the optimal solution are based on the requirement that the influence model must be able to distribute a single status to all sites corresponding to a particular subset, from a particular *source* site (which in the case of partitioning with reference nodes is the reference).

Before presenting results on the algorithm's ability to find optimal partitions, let us begin by formally defining source vertices, so that we can formalize the connectivity conditions required of the optimal:

Definition 5.1. Consider a particular partition of a graph, and a vertex v within a particular subset. For this partition, the vertex v is a source vertex, *if we can find a path from v to each other vertex in the subset that remains within the subset (i.e., never enters a vertex in another subset).*

We are now ready to present the main result on the algorithm's ability to obtain the optimal solution. We assume throughout this development that the partitioning problem of interest to us has at least one feasible solution. We first give conditions under which the algorithm can reach the optimal solution for a partitioning problem with reference nodes.

Theorem 5.1. Consider the general k-way partitioning problem with reference nodes, as described in Section 2. An optimal solution is identified by the influence model algorithm with probability 1 *(i.e., the algorithm passes through an optimal solution), if there is an optimal solution such that each reference vertex is a source vertex.*

Proof. In order to show that the optimal solution is identified with probability 1, let us consider the master Markov chain for the influence model. We only need to show that the optimal state (the influence model state associated with the optimal solution) can be reached with positive probability from any other state (i.e. there is a sequence of

influence model updates that leads from an arbitrary state to the optimal state) [38]. The optimal state has the property that all the sites in each partition have the same status, while sites in different partitions have different statuses.

In showing that the optimal state can be reached, let us limit ourselves to updates in which sites determine their statuses from other sites in the same partition in the optimal solution—only such updates are needed. Now consider a single subset in the optimal solution. Let us call the reference vertex in the subset v_s. Since v_s is a source vertex, there is a path from v_s to every other vertex in the subset that remains in the subset. Let us suppose that the longest path from v_s to another vertex in the subset is m. Then we note that there is a positive probability that all influence model sites corresponding to that subset take a status of the reference site (the site corresponding to the reference vertex) after m time-steps. This can be proved simply by recursion: assume there is a positive probability that all sites within a distance of i from the reference take on the initial status of the reference site at each time step i; since there is a positive probability that each site within a distance of $i + 1$ is influenced by a site within a distance of i from the reference, there is also a positive probability that all sites within a distance of $i + 1$ from the reference site take on the reference status at time $i + 1$. Using this argument, we also find that there is non-zero probability that all sites take on the reference status, at any time $k \geq m$. Thus considering the influence model as a whole, we see that there is a positive probability that all partitions are found after a finite number of time-steps, and so the theorem is proved. □

We note that this proof is closely related with the proof characterizing the asymptotics of a *binary influence model* in [3].

We have thus shown that the algorithm can solve the partitioning problem for a wide variety of cost functions, specifically ones in which the optimal solution has the described connectivity condition. We stress that the connectivity condition—namely, the existence of paths from each reference vertex to the other vertices in its subset—is quite weak: connectedness of the subsets in the (directed) graph is sufficient but not necessary for the connectivity condition to hold. In fact, for a range of distributed applications (for instance, for multicasting in mobile networks or tracking using autonomous-vehicle teams), such connectivity may automatically be required or desired since we need agents/nodes in each identified subset to subsequently communicate among themselves.

Let us next formalize that the algorithm can be used to find optimal solutions for the k-way partitioning problem without reference nodes.

Theorem 5.2. Consider the general k-way partitioning problem. An optimal solution is identified by the influence model algorithm with probability 1, *if each subset of some optimal solution has a source vertex.*

Proof. Since we solve partitioning problems without reference nodes by searching through distinct reference node placements, this result follows directly from Theorem 5.1.

We have noted that Theorems 5.1 and 5.2 require the optimal solution to have a particular weakly-connected structure to guarantee its identification. Of course, the optimal partition is not known *a priori*, so it is helpful to identify classes of partitioning problems for which this connectivity condition is necessarily true. The following corollary identifies two such classes.

Corollary 5.3. Consider the min-cut problem and mass-weighted min-cut problem with/without reference nodes. An optimal solution is identified by the influence model with probability 1, *if the graph has the following structure: all the edges are bi-directional.*

Proof. It is easy to check that optimal partitions for these problems constitute connected subgraphs. Thus, together with the bi-directionality assumption, we see that Theorems 5.1 and 5.2 can be applied.

Theorems 5.1, 5.2, and Corollary 5.3 show that an optimal solution is identified with probability 1 (i.e. the influence model passes through an optimal solution), given that this solutions satisfies the appropriate connectivity conditions. However, we have not yet shown that the algorithm will stop at the optimal solution with certainty. The following two theorems show that our partitioning scheme is successful when the centralized and distributed stopping criteria are used, respectively.

Theorem 5.4. Consider the general k-way partitioning problem with (without) reference vertices, and assume that each reference vertex is a source vertex (respectively, each subset has a source vertex) in the optimal solution. Then the probability that the influence model algorithm with centralized stopping chooses the optimal solution approaches 1, *in the limit of long waiting times.*

Proof. This result follows directly from the standard analysis of Markov chains (see e.g. [38]). Specifically, as the waiting time is increased, the probability that a better solution, if one exists, is not found while waiting can be seen to decrease to 0.

Partitioning with distributed stopping (in particular, partitioning with adaptive stopping) is quite a bit more complicated to analyze than the centralized algorithms, because the parameters of the influence model are changing in reaction to the site statuses. Here, we formalize that the partitioning-with-adaptive-stopping algorithm is able to solve the min-cut k-way partitioning problem with reference nodes, in the case where the edges between subsets are weak (of order ϵ in weight) compared to edges in the partition. Although our formal result is in such a limiting case, the proof in fact makes clear that the minimum cut is found whenever the influence model associated with the original graph is more likely to have status differences over the minimum cut than over any other cut. The influence model has this property for a large (albeit somewhat hard to delineate) class of graphs, not only ones with weak minimum cuts; this is sensible, since after all the influence model update is structured to find minimum cuts (not only order- ϵ cuts) more commonly than other cuts. Our examples bear out that the distributed-stopping algorithm is practical for typical distributed applications.

Here is the formal result, with proof:

Theorem 5.5. Consider the min-cut partitioning problem with reference nodes. Assume that the graph has bi-directional edges, and further that the optimal cut is small (of order ϵ*) compared to any other cut. Then the probability that the influence model algorithm with distributed stopping chooses the optimal cut approaches* 1, *in the limit of small* δ.

Proof. First notice that if we can show that all the weights of edges in the optimal cutset (the cutset associated with the optimal solution) go to 0 before any other one does in the average sense, we are done since as δ approaches 0, the probability for a particular run to be deviated from the average run approaches 0. Let $\epsilon_1[0]$, $\epsilon_2[0]$,. . .$\epsilon_n[0]$ denote the weights of edges in the optimal cutset of an influence network I at time-step 0. Without loss of generality, we arbitrarily pick an edge with weight l[0] other than the edges in the optimal cutset and show that all the n $\epsilon_i[k]$'s approach 0 before l[k] approaches $\lambda' = \lambda[0] - \sum_{i=1}^{n} \in_i [0]$ in the average sense at some time-step k. With the assumption that the edges in the optimal cutset are sufficiently weak, we have $\lambda' > 0$, then we are done.

To do so, we construct a new influence network I', whose only difference with I resides in that the weight $\lambda[k]$ is replaced by l', and each $\epsilon_{\mathrm{i}}[k]$ is replaced by $\epsilon_{\mathrm{i}}[0]$. The reason to come up with I' is that the original I is a very complex network with varying weights. By proving for I' whose weights never change, that the conclusion holds first, and reducing the problem for I to the one for I', we can simplify the proof.

Considering I' with fixed weights, it is easy to check that in the average sense, $\epsilon_{\mathrm{i}}[k]$ reaches 0, before $\mathrm{l}[k]$ reaches $\lambda[0] - \epsilon_{\mathrm{i}}[0]$; consequently, both $\epsilon_{\mathrm{i}}[k]$ and $\epsilon_j[k]$ reach 0 before $\lambda[k]$ reaches $\lambda[0] - \epsilon_{\mathrm{i}}[0] - \epsilon_{\mathrm{j}}[0]$; and finally, all $\epsilon_{\mathrm{i}}[k]$ reach 0 before $\lambda[k]$ reaches λ', where $\lambda[k]$ and $\epsilon_{\mathrm{i}}[k]$ are weakened with time. This is because with the existence of a sufficiently small optimal cut, the probability for each site in an optimal partition to take the reference site's status is very high, and thus the joint probability for a pair of sites in an optimal partition to take different statuses are very small. In contrast, the probability for a pair of sites across the optimal cut to have different statues are very high. Therefore, the edges in the optimal cutset are weakened faster than weight $\lambda[k]$ does in the average sense.

Now that we know for I' with fixed weights, all $\epsilon_{\mathrm{i}}[k]$ reach 0 before $\lambda[k]$ reaches l' in average, we need to show that it implies for I with varying weights, the same conclusion also holds. With the assumption that $\lambda[k]$ is greater than λ', $\lambda[k]$ in I approaches λ' slower than $\lambda[k]$ in I' does, and every $\epsilon_{\mathrm{i}}[k]$ in I approaches 0 no slower than than $\epsilon_i[k]$ in I' does, since $\epsilon_{\mathrm{i}}[k]$ may be weakened in I. The above assumption is true since before $\lambda[k]$ decreases to l', all the edges in the optimal cutset are already broken. Hence we prove that all the edges in the optimal cutset approach 0 before other edge does in the average sense. The proof is complete.

We have thus shown that our algorithm can find the optimal partition in both a centralized and a distributed manner, under broad conditions. Next, it is natural to characterize or test the performance of the algorithm: of course, any algorithm that searches through all possible partitions can find the optimal one, so an algorithm such as ours is useful only if it can find the optimal solution quickly compared to a combinatorial search. Although we leave a full analytical treatment of the algorithm's performance for future work, we give here a conceptual discussion of why the algorithm is fast, and also evaluate the performance of the algorithm in examples in the next section.

The *No Free Lunch* theorems [39] provide an interesting conceptual framework for the performance evaluation of our algorithm. These negative results state that, over the class of all possible cost functions, there are no algorithms that always perform well; in fact, all algorithms are equally costly (i.e., take equally long) on average. Thus, an algorithm must be tailored for the particular cost function of interest. From this perspective, our algorithm works well because typical optimal costs correspond to weak cuts in the graph and strongly-connected partitions, and hence good algorithms should search through these weak-cut partitions first. Our algorithm is tailored to quickly find these weak-cut solutions (since the strongly-connected sites in the copying influence model tend to adopt the same status while weakly connected ones differ), while also searching through other solutions less frequently.

We have recently obtained an analytical justification for the performance of the algorithm. Specifically, we can show that, on average, the algorithm solves the k-way min-cut problem with reference nodes in polynomial time, given that the minimum cut is sufficiently weak compared to other cuts in the graph. Since the k-way partitioning problem with reference nodes is NP-hard, a polynomial-time algorithm for a class of graphs is a worthwhile result, and gives some indication of the performance of the algorithm. This performance analysis also has the benefit of explicitly connecting the performance with spectral properties of the linear recursion for influence model site statuses, and hence potentially permitting comparison of the algorithm with spectral partitioning methods (e.g. [26, 27, 28]). This analysis of performance unfortunately

requires rather extensive review of the influence-model's analysis, so we omit the details of the result from this expository article.

6. Applications and Examples

In this section, we briefly introduce several potential applications of our algorithm, and also present canonical examples that illustrate or enrich aspects of our analytical development. The applications and examples together are meant to further motivate the described algorithm.

6.1. Application 1: Classification for Multicasting in Ad Hoc Networks

Distributed partitioning holds promise as a tool for classification in distributed sensor networks and mobile ad hoc networks, e.g. for the purpose of multicasting or of transmitting information from the sensors/mobiles back to "leader nodes" or base stations or central authorities.

There is a wide literature on *routing* in ad hoc networks when the absolute positions of the sensors/mobiles are known (see [40] for a survey of methods). Recently, distributed algorithms (specifically local-averaging methods) have been used to infer location information in the case where absolute positions are unknown except at peripheral locations (see e.g. [41]), and hence permit the development of routing algorithms for these networks. Beyond routing, the classification of sensors/mobiles with base stations is an important task, for the purpose of multicasting (transmitting information to many destinations from multiple sources) or so that subsequently data can be routed to and from appropriate base stations to the sensors/mobiles.

Several recent articles have addressed multi-hop multicasting in ad hoc networks (see e.g. [10]). In multicasting applications as well as other settings where data may be transmitted to/from several sources or base stations, classification of mobiles/sensors with the base stations is important. We contend that the influence model-based partitioning tool can advance the state-of-the-art on classification in ad hoc networks, for several reasons:

- As made clear by the comparison of location-known and location-unknown algorithms for routing, decentralized algorithms for classification may be needed in cases where there is no central authority with full knowledge of the network. Even if the classification is done in a centralized manner, only partial information may be known about the network. For instance, the distances between sensors/mobiles or at least the connection topology may be known, but the exact locations of each sensor/mobile may not. Conversely, the exact locations may be known, but the connection topology may be unknown. The mapping from the graph to the influence model, and the influence model update itself, are based on local information and hence our partitioning method is suited for this setting.
- We may need to optimize the classification with respect to several (possibly complex) cost criteria (including for example minimum (or average) hops to each base station, average delay cost, and various reliability criteria). In fact, the costs may depend on the specifics of the decentralized algorithm used for routing/multicasting. The influence model-based algorithm permits us to consider multiple and complex cost criteria.
- Often, the topologies of sensor networks and mobile ad hoc networks change with time, and hence it is beneficial to use an algorithm that can update the optimum with little effort (in either a distributed or centralized case). The influence model-based algorithm has this advantage.

For illustration of this application, we have used the influence model-based algorithm for sensor classification in a small example (one with 3 base stations and 27 sensor nodes). The example was generated by placing the 30 sensors in a uniform i.i.d. manner within the unit square, allowing communication between sensors within 0.3 units of each other, and choosing three sensors (Sensors 3, 14, and 20) to also serve as base stations. We associate a weighted undirected graph with the sensor network in which the 30 vertices correspond to the 30 sensors, and the branches indicate communication between sensors. Each branch weight is chosen to be inversely proportional to the distance between the pair of sensors, with the motivation that longer communication links are more apt to failure and delay and hence are more weakly connected. We consider 3-way partitioning of this graph with reference vertices 3, 14, and 20 using the influence model algorithm.

We consider partitioning with centralized stopping, with respect to two cost functions:

- First, we partition the graph so as to *maximize* the minimum of the positive eigenvalues of the *Laplacian matrices* associated with the three subsets (partitions)[5]. The minimum non-zero eigenvalue of the Laplacian associated with each subset is well-known to indicate the connectivity of that subset, and can be used to bound several relevant graph-theoretic properties such as the graph diameter (see [42] for a full development). By maximizing the minimum among the non-zero eigenvalues, we thus find a partition with strongly-connected subsets and weak links between them. The optimal partition with respect to this *minimum-subgraph-eigenvalue* cost measure is shown in Fig. 2.
- Our second cost measure is motivated by consideration of low-cost and low-overhead distributed routing for ad hoc and sensor networks. A simple greedy algorithm for routing when location information is available is to send the message to the node (sensor) closest to the destination during each transmission (see e.g. [40]). Assuming such a greedy routing algorithm is used, our aim is to classify the sensors with base stations so that the maximum number of hops to a sensor from its base station is minimized. (The average number of hops could be used instead.) Thus, we partition the graph using this maximum number of hops when greedy routing is used. The optimal partition when this *greedy-routing cost measure* is shown in Fig. 2. We note that, as expected, the optimal partition has subsets which are more balanced in size but contain weaker links, as compared to optimum for the minimum-subgraph-eigenvalue measure. This example highlights an interesting advantage of the influence model: the greedy-routing cost function does not admit an analytical form but can be computed for a given partition, but nevertheless the optimal partition can be found.

We have also considered min-cut partitioning with distributed (adaptive) stopping for this example. The result is shown in Fig. 2. We note that such a distributed algorithm could be implemented in the sensor network itself, and would only require individual sensors to have local parameters (in particular, distances to neighbors). Such a distributed algorithm might be especially useful in cases where the topology is subject to change, so that the sensors must re-classify themselves periodically.

As further illustration, we also show a 4-way partition with reference nodes of a 100-sensor network in Fig. 2.

[5]We notice a maximization problem can routinely be converted to a minimization by choosing a cost that is the negative of the original cost.

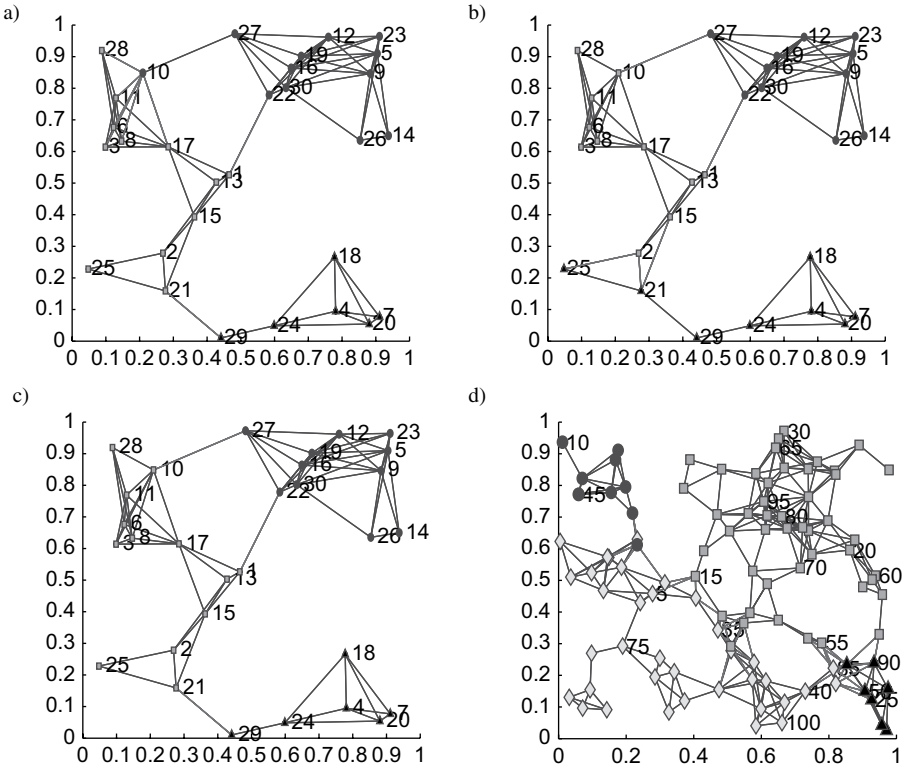
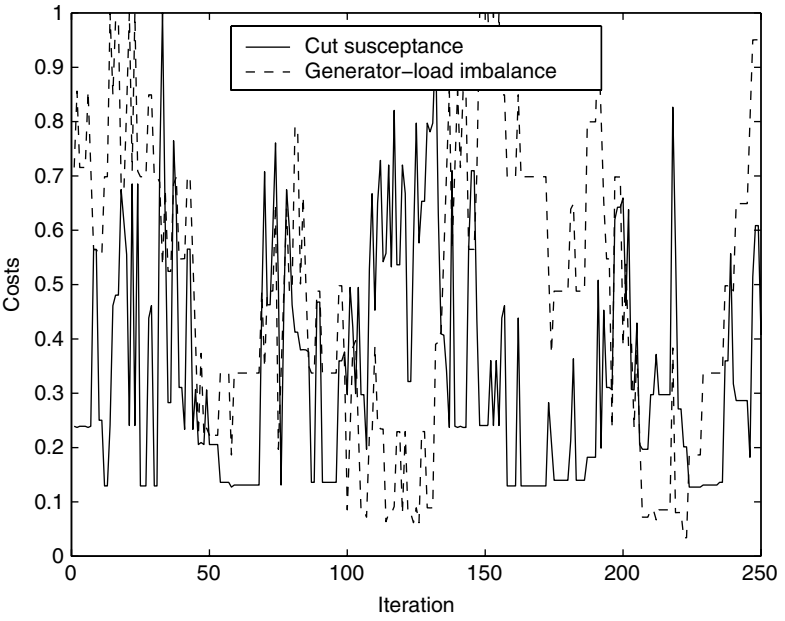


Figure 2. Partitioning a 30-sensor network with reference nodes a) based on a minimum-subgraph-eigenvalue cost, b) based on a greedy-routing cost, and c) with distributed stopping. We also partition a 100-sensor network based on a minimum-subgraph-eigenvalue cost (d).

6.2. Application 2: Flexible Partitioning for Electric Power Systems

We believe that our algorithm potentially has significant application in power system analysis, because of its flexibility. One potential application is for *islanding*, i.e. the isolation of a portion of the power network to prevent disturbance propagation. We note that a good algorithm for islanding may need to take into account several requirements, including small generation-load imbalance within the isolated component, feasibility of making the desired cut, and disturbance-propagation dynamics. Our algorithm is a natural tool for such partitioning problems, in that we can minimize and keep track of multiple costs while identifying plausible islands (partitions). We have applied the influence model algorithm to identify plausible islands in a very small (14−bus) example (see Fig. 3). Specifically, we have used the influence model algorithm to track and minimize two cost metrics—the cut susceptance and the total absolute generator-load imbalance over the partitions. Our optimization shows the use of a partitioning algorithm that can track multiple costs: two different optimal solutions are identified quickly based on the two cost metrics, and both costs are computed for a family of plausible partitions (Fig. 3). Interestingly, the optimal partition with respect to the minimum susceptance cost is typically found more quickly than the optimal partition with respect to the generator-load imbalance cost. The better performance in finding the minimum susceptance cost is not surprising, in that the susceptances are directly mapped to influence probabilities.



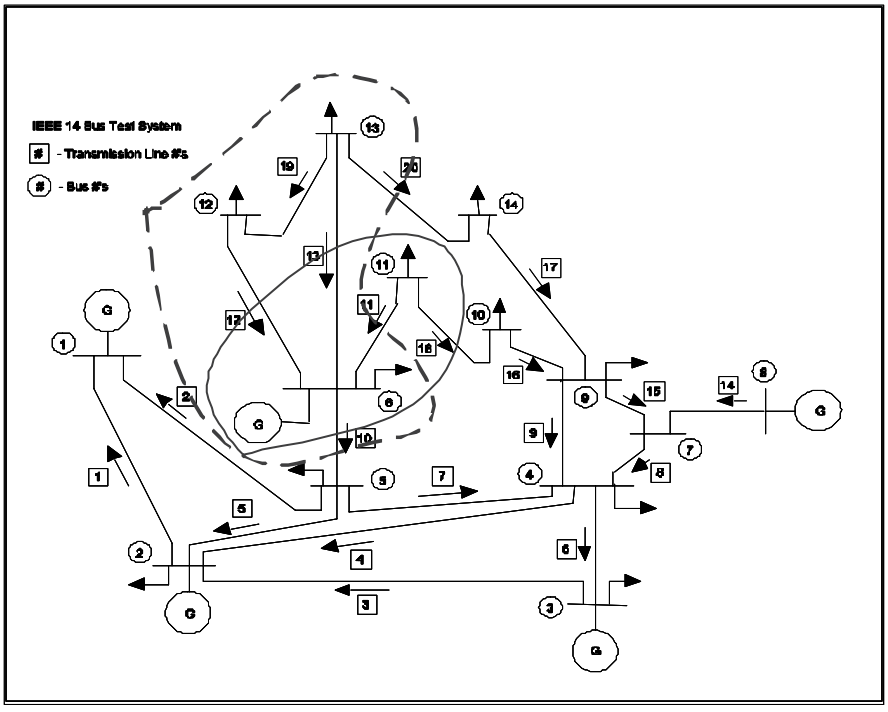


Figure 3. We consider partitioning the Standard 14-Bus IEEE Test System, for the purpose of isolating a disturbance at Bus 6 from the slack bus 1. The upper figure shows the cut susceptance and generator-load imbalance for a sequence of 250 partitions generated by our influence model algorithm. The lower figure highlights that the minimum-cost partitions according to these two criteria are different. In the lower figure, the solid line indicates the minimum susceptance cut while the the dashed line indicates minimum generator-load imbalance.

Because our algorithm can track multiple costs, we believe that it can potentially enhance the recent approach to islanding suggested in [19], which is slow-coherency (equivalently, spectrally) based. More generally, we believe that the flexibility offered by our partitioning algorithm may be valuable for various model-reduction tasks in power system analysis, as well as other network-centric tasks such as the identification of line trips of minimal cardinality that make infeasible the power flow solution.

6.3. Application 3: Self-Classification for Autonomous Vehicles

Our distributed partitioning algorithm provides a means for networked autonomous vehicles to classify themselves. For instance, say that a network of communicating autonomous vehicles seeks to form two teams, each of which must congregate at a convenient location in space. Our self-partitioning algorithm can be used to identify groups of autonomous vehicles that are close to each other, and so to form the teams, in a completely distributed manner. We note that using partitioning in this context permits task dynamics in autonomous-vehicle applications, for which the role played by each agent actually depends on its initial position (state). In the interest of space, we do not pursue this application in detail here.

6.4. Canonical Example 1: Performance Simulations

We explore the performance of our algorithm by pursuing min-cut partitioning with reference nodes on a seven-node circuit. Figure 4 illustrates the mapping between the circuit's conductance graph and an influence model. It is worth noting that the mapping has a meaningful dynamic interpretation in this case: the expected dynamics of the influence model is a discretization of the dynamics of a circuit with the specified conductances and unit capacitors from each node to electrical ground.

Our algorithm is guaranteed to find the optimal cut. Table 1 shows the performance of our algorithm, in terms of the average number of time-steps needed to reach this cut, for several values of the cut strength and discretization time-step. We note that the expected number of time-steps needed to reach the optimal cut is dependent on the strength of the optimal cut: the weaker the cut, the faster the algorithm.

We have compared our algorithm with spectral methods for this circuit example. When the weak cut is between nodes 2 and 3 and nodes 4 and 5, both the spectral method and our algorithm find the min-cut partition for all $0 \leq \epsilon < 1$. We notice that our algorithm requires 5 random number generations and 5 copying actions (communications in a decentralized setting) per time-step, and requires between 5 and 10 time-steps for a good choice of Δ and $0 \leq \epsilon \leq 1$. In comparison, a simple implementation of the spectral method requires on the order of 7^3 additions and multiplications. We notice that the expected computational cost of our algorithm depends on the strength of the cut, in contrast to the spectral algorithm. Interestingly, if the size-ϵ cuts are placed between node 1 and nodes 2 and 3 instead, the spectral method only finds the weak cut for $\epsilon \leq 0.26$, while our algorithm obtains the optimal solution for all ϵ (albeit at higher computational cost).

6.5. Canonical Example 2: Convergence of Distributed Stopping

Recall that distributed stopping of our algorithm is achieved through reduction of influences. In this case, the algorithm's ability to obtain the minimum cut is predicated on choosing a sufficiently small influence reduction size. In this example (see Fig. 5), we have characterized the percent of time that the correct partition is found, for several cut

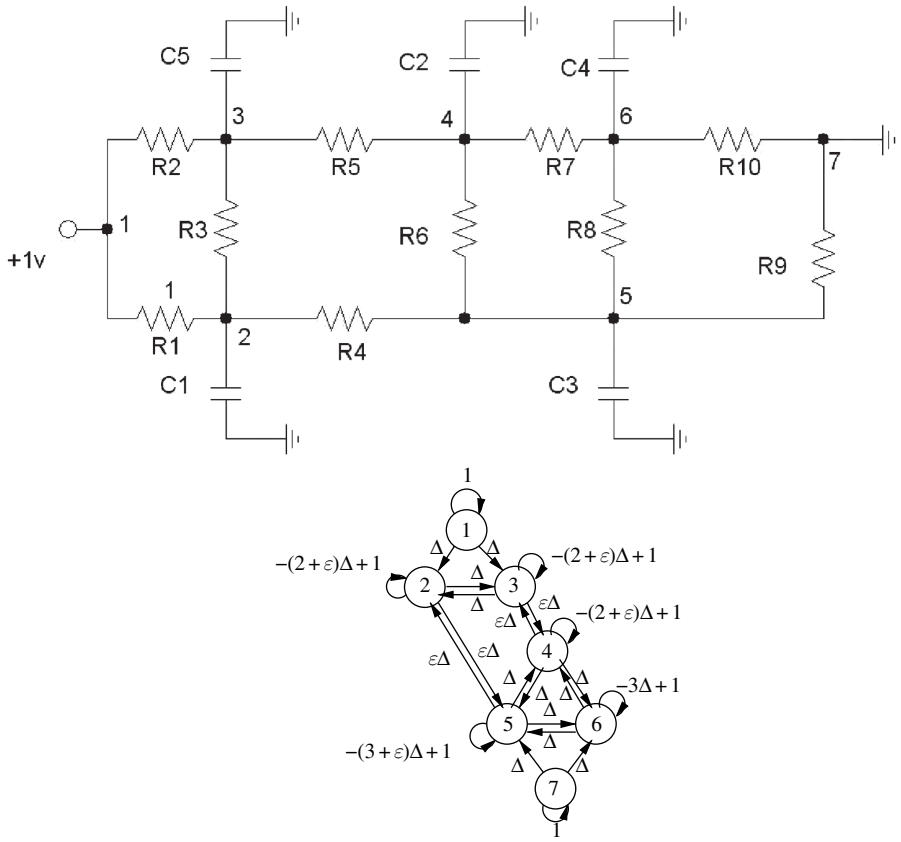


Figure 4. Seven-node circuit and its influence model: Conductance values of R_4 and R_5 are ϵ, and all the other conductances are 1; all the capacitances are 1; Δ stands for the discretization step.

Table 1
The Average Steps to First Reach the Partition State with Respect to Δ and ε Based on 1000 Sample Runs

	ε			
Δ	0.1	0.2	0.5	1
0.05	26.2	27.6	31.8	37.8
0.1	13.2	13.8	16.3	19.6
0.2	6.9	7.0	8.6	10.8
0.25	5.5	5.7	7.1	9.6
0.3	4.7	4.9	NA	NA

strengths and influence reduction sizes. In each case, we have also determined the expected number of iterations until the algorithm stops (see Table 2 for the results). The simulation results indeed show that the optimal cut is found with certainty, when the influence reduction size is chosen sufficiently small. We note that the time required to find the optimal partition increases as the influence reduction size is decreased.

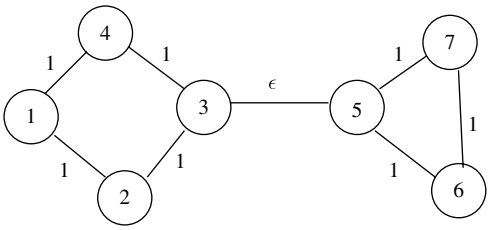


Figure 5. Example for distributed partitioning.

Table 2
Simulation Result for Example 2 Based on 1000 Sample Runs: *Steps* Represents the Average Steps the Algorithm Takes to Distributedly Stop, and *Percent* Represents the Percentage of Correct Partitions the Algorithm Finds.

	ϵ	δ	Δ	Steps	Percent
1	0.1	0.02	0.47	6.2	99.6
2	0.1	0.01	0.47	8.3	100
3	0.5	0.01	0.39	30.759	99.7
4	0.5	0.005	0.39	56.672	100
5	1	0.001	0.33	122.49	86.8

7. Future Work

Several directions of future work are worth discussing:

- **Complexity Analysis and Comparison**. A careful analysis of the complexity of our algorithm is required. Roughly, the computation required at each time-step scales with the number of edges in the graph. However, we have not determined the scaling of the number of time-steps required to find the optimal solution with the size of the graph, expect in the case where the optimal cut is sufficiently weak compared to the others. We reiterate that, in contrast to e.g. spectral algorithms, the time taken by our algorithm depends on the structure of the graph and the cost being minimized, and hence much remains to be done in connecting the number of time-steps with graph properties. We believe that our earlier study of the settling rates of influence model-based agreement protocols (see [11]) may permit analysis of the settling rate of our partitioning algorithm. A complexity analysis of our model will also permit further comparison of our algorithm's performance with those of other algorithms.
- **Anti-Copying Algorithm for Bisection**. Bisection and recursive bisection are of significant interest in the parallel processing community, and remain hard for some graph structures [35]. In its current form, our algorithm is not well-suited for bisection problems: the constraint of equal-mass components is rarely satisfied by the partition generated at each time-step, so the algorithm is likely to take many time-steps to identify the optimal solution. Further, the bisection solution need not have connected components (even in the symmetric case), and hence the optimal solution may not be found at all. However, we believe that a simple modification

of the influence model algorithm can permit bisection. To conceptualize this algorithm, we note that the bisection problem can be rephrased as an unweighted max-cut problem for a certain dual graph [37], and hence that we can seek an algorithm for identifying max-cuts to solve the bisection problem. An interesting approach for finding maximum cuts is to use a two-status influence model in which each site copies the opposite of its determining neighbor's status. This model favors configurations in which nearby neighbors' statuses are different from each other, and so has a tendency to find large cuts. This may turn out to be a computationally effective technique for finding maximum cuts and hence solving bisection problems.

- **Reference-Free Partitioning**. The sequential selection of reference vertex sets required in our algorithm is undesirable when a large number of partitions are needed. We are exploring variants of the influence model-based algorithm that do not require selection of reference generators. One interesting strategy is to initialize the sites in the influence model with different statuses, and then update the model until the desired number of partitions is identified. This solution can possibly be selected as the partition, or it can be used as a pre-partition based on which reference generators are selected influence-model algorithm is applied.

Acknowledgements

The second author thanks Professor George C. Verghese of the Massachusetts Institute of Technology for several illuminating conversations on influence model-based partitioning. The second and third authors were partially supported by the National Science Foundation under Grant ECS-0528882 (Sensors), and the third author was also partially supported by the Office of Naval Research under Grant N000140310848.

About the Authors

Yan Wan received her B.S. Degree in Electrical Engineering from Nanjing University of Aeronautics and Astronautics, Nanjing, China in 2001, and her M.S. Degree from University of Alabama, Tuscaloosa, Alabama in 2004. She is currently working toward her Ph.D. Degree in the School of Electrical Engineering and Computer Science at Washington State University, Pullman, Washington.

Sandip Roy received a B.S. in Electrical Engineering from the University of Illinois at Urbana-Champaign in 1998, and a M.S. and Ph.D. in Electrical Engineering from the Massachusetts Institute of Technology in 2000 and 2003, respectively. Since 2003, he has worked as an assistant professor in the School of Electrical Engineering and Computer Science at Washington State University. He has also held summer research appointments at the University of Wisconsin at Madison, the National Aeronautics and Space Administration, and the Lawrence Berkeley National Laboratories during this time. Dr. Roy's current research is focused on the control and design of dynamical networks, and has application in such diverse fields as sensor networking, electric power network analysis, and systems biology.

Ali Saberi lives in Pullman, Washington.

Bernard Lesieutre (S'86-M'93) received the B.S., M.S., and Ph.D. degrees in electrical engineering from the University of Illinois at Urbana-Champaign in 1986, 1988, and 1993, respectively. He is an Associate Professor at University of Wisconsin-Madison.

Prior to this appointment, he was a Staff Scientist at the Ernest Orlando Lawrence Berkeley National Laboratory, Berkeley, CA. He served on the faculty of the Massachusetts Institute of Technology, Cambridge, as Assistant Professor and then Associate Professor of electrical engineering from 1993 to 2001. He has also held Visiting Associate Professor appointments at Caltech and Cornell University. His research interests include the modeling, monitoring, and analysis of electric power systems and electric energy markets.

References

1. H. Qi, S. S. Iyengar, and K. Chakrabarty, ''Distributed sensor networks – a review of recent research'', *Journal of the Franklin Institute*, vol. 338, pp. 655–668, 2001.
2. R. M. Murray, K. J. Astrom, S. P. Boyd, R. W. Brockett, and G. Stein, "Future directions in control in an information-rich world," *IEEE Control Systems Magazine*, vol. 23, no. 2, Apr. 2003.
3. C. Asavathiratham, S. Roy, B. C. Lesieutre and G. C. Verghese. ''The influence model,'' *IEEE Control Systems Magazine*, Dec. 2001.
4. J. A. Fax and R. M. Murray, ''Information flow and cooperative control of vehicle formations'', submitted to *IEEE Transactions on Automatic Control*, April 2003.
5. S. Roy, A. Saberi, and K. Herlugson, ''Formation and alignment of distributed sensing agents with double-integrator dynamics'', *IEEE Press Monograph on Sensor Network Operations*, 2004 (in press).
6. B. Chamberlain, "Graph partitioning algorithms for distributed workloads of parallel computations," *Technical Report UW-CSE-98-10-03*, University of Washington, Oct. 1998.
7. R. B. Boppana, "Eigenvalues and graph bisection: and average case analysis," *IEEE FOCS*, pp. 280–285, 1987.
8. B. Kernighan and S. Lin, "An efficient heuristic procedure for partitioning graphs," *Bell Systems Technical J*., vol. 49, pp 291–307, Feb. 1970.
9. D. Johnson, C. R. Aragon, L. A. McGeoch, and C. Schevon, "Optimization by simulated annealing: an experimental evaluation; part I, graph partitioning," *Operations Research*, Nov/Dec 1989; 37, 6.
10. C. Cordiero, H. Gossain, and D. P. Agrawal, "Multicast over wireless mobile ad hoc networks: present and future directions," *IEEE Networks Magazine*, Jan./Feb. 2003.
11. S. Roy, K. Herlugson, and A. Saberi, "A control-theoretic approach to distributed discrete-valued decision-making in networks of sensing agents," to appear in the *IEEE Transactions on Mobile Computing*.
12. S. Roy, A. Saberi, and K. Herlugson, "A control-theoretic perspective on the design of distributed agreement protocols" to appear in the *International Journal of Robust and Nonlinear Control, Special Issue on Communicating-Agent Systems*. Short version in the *Proceedings of the American Control Conference*, Jun. 2005.
13. R. O. Saber and R. M. Murray, "Consensus problems in networks of agents with switching topology and time-delays," *IEEE Transactions in Automatic Control*, vol. 49, pp. 1520–1533, Sep. 2004.
14. B. Krisnamachari and S. S. Iyengar, "Distributed Bayesian algorithms for fault-tolerant event region detection in wireless sensor networks," *IEEE Transactions on Computers*, vol. 53, no. 3, Mar. 1, 2004.
15. S. D. Servetto and G. Barrenechia, "Constrained random walks on random graphs," in *Proceedings of the* 1 *st ACM International Workshop on Wireless Sensor Networks and Applications*, Atlanta, GA, Sep. 2002.
16. D. Kempe, J. Kleinberg, and A. Demers, "Spatial gossip and resource location protocols," in *Proceedings of the* 33*rd ACM Symposium on Theory of Computing*, 2001.
17. T. Liggett, *Interacting Particle Systems*, Springer-Verlag (Mathematical Reviews Series), New York, 1985.

18. G. Grimmett, *Percolation*, 2*nd ed*., New York, 1999.
19. H. You, V. Vittal, and X. Wang, "Slow Coherency-based Islanding," IEEE Transactions on Power Systems, vol. 19, no. 1, Feb. 2004.
20. A. Pinar and B. Hendrickson, "Partitioning for complex objectives," in *Proceedings of the International Parallel and Distributed Processing Symposium (IPDPS)*, 2001.
21. M. Garey and D. S. Johnson, *Computers and intractability: a guide to the theory of NP-completeness*, San Francisco: Freeman, 1979.
22. "A compendium of NP optimization problems," http://www.nada.kth.se/~viggo/wwwcompendium/wwwcompendium.html.
23. M. S. Khan, *Sequential and distributed algorithms for fast graph partitioning*, Master's thesis, University of Victoria, Victoria, B.C., Canada, Aug. 1994.
24. M. Stoer and F. Wagner, "A simple min-cut algorithm," *Lecture Notes in Computer Science*, vol. 855, pp. 141–147, 1994.
25. D. Karger and C. Stein, "A new approach to the minimum cut problem," *Journal of the ACM*, vol. 43, no. 4, pp. 601–640, 1996.
26. P. K. Chan, M. D. Schlag, and J. Y. Zien, "Spectral k-way ratio-cut partitioning and clustering," *30th ACM/IEEE Design Automation Conference*, Dallas Texas, June 1993.
27. C.L. DeMarco and J. Wassner, "A generalized eigenvalue perturbation approach to coherency," *Proc. IEEE Conference on Control Applications*, pp. 611–617, Sept. 1995.
28. S. Roy and B. Lesieutre. "Studies in network partitioning based on topological structure," *32nd Annual North American Power Symposium*, Waterloo, Canada, Oct. 2000.
29. R. D. Williams, "Performance of dynamic load balancing algorithms for unstructured mesh calculations," *Concurrency: Practice and Experience*, vol. 3, pp. 457–481, 1991.
30. F. Cao, J. R. Gilbert, and S. Teng, "Partitioning meshes with lines and planes," *Technical Report CSL-96-01*, Xerox Palo Alto Research Center, Jan. 1996.
31. M. Fiedler, "A property of eigenvectors of nonnegative symmetric matrices and its application to graph theory," *Czechoslovak Mathematics Journal*, vol. 25, no. 100, pp. 619–633, 1975.
32. T. Nguyen Bui and B. R. Moon, "Genetic algorithm and graph partitioning," *IEEE Transactions on Computers*, vol. 45, no. 7, July 1996.
33. D. E. Goldberg, "Genetic algorithms in search," *Optimization and Machine Learning*, Addison-Wesley, New York, 1989.
34. B. Hajek, "Cooling schedules for optimal annealing," *Mathematics of Operations Research*, vol. 13, pp. 311–329, May 1998.
35. H. D. Simon, S-H. Teng, "How good is recursive bisection," *SIAM Journal on Scientific Computing*, vol. 18, no. 5, pp. 1436–1445, 1997.
36. C. K. Cheng and Y. C. Wei, "An improved two way partitioning algorithm with stable performance," *IEEE Trans. On Computer-Aided Design*, 10(12):1502-1511, Dec. 1991.
37. C. H. Lee and C. I. Park, "An efficient *k*-way graph partitioning algorithm for task allocation in parallel computing systems," *Proceedings of the* 1*st International Conference in System Integration*, pp. 748–751, 1990.
38. R. G. Gallager, *Discrete stochastic processes*, Kluwer Academic Publishers, 1996.
39. D. H. Wolpert and W. G. Macready, "No free lunch theorems for search," Technical Report, Santa Fe Institute, no. 95-02-010, 1995.
40. M. Mauve, J. Widmer, and H. Hartenstein, "A survey of position-based routing in mobile ad hoc networks," *IEEE Networks Magazine*, vol. 6, pp. 30–39, Dec. 2001.
41. A. Jadbabaie, "On geographic routing with location information," submitted to *Proceedings of the IEEE Conference on Decision and Control*, The Bahamas, 2004.
42. F. R. K. Chung, *Spectral Graph Theory*, American Mathematical Society Press: Providence, RI, 1997.