

Formal Foundations

Summary: This chapter sets out a precise model of decentralized spatial information systems, like geosensor networks. The model has three main levels, each of which builds on structures in the previous level. First, a minimal, neighborhood-based model of geosensor networks provides the most fundamental structures for sensing and communicating information about the environment. Next, an extended spatial model of static geosensor networks provides additional quantitative and qualitative structures to model spatial location. Third, a spatiotemporal model of dynamic geosensor networks provides the capability to precisely model spatial and environmental change over time. In addition, two auxiliary structures are also discussed: a model of the partial knowledge about the decentralized system available to each individual node; and the most common communication network structures, which constrain the movement of information in any decentralized spatial information system.

SETTING the scene for decentralized spatial computing, the previous chapter aimed to motivate our interest in the topic, frame the fundamental problems and concepts, and sketch the vision for the application of decentralized spatial computing to solving real-world problems. In this chapter, we come back down to earth (with a bump!) and examine the basic components of decentralized spatial computation by defining in a precise and abstract way the key computational elements of a geosensor network.

2.1 Introduction

A formal model of a geosensor network must provide assistance in representing and reasoning about the features of geosensor networks important in decentralized spatial computing. Equally important, a formal model also needs to suppress unnecessary detail about the geosensor network. For example, in domains such as network routing, hardware design, and operating systems,

many technical details of geosensor networks are important, including the specific details of:

- the characteristics and types of sensors available on the node;
- the processing speed, architecture, and capabilities of the microcontroller;
- the storage capacity and characteristics of the node's memory;
- the communications protocol used to exchange information between nodes; and
- the power requirements of specific node operations, such as the relative energy budgets required for computation, communication, and sensing.

However, in designing decentralized spatial algorithms it is helpful to abstract away from these technical details, especially as they are expected to develop rapidly, changing with future technological advances. Instead, the features of a geosensor network that are important to a decentralized spatial algorithm include:

- the set of nodes and the structure of the communication network connecting the nodes;
- the geographic locations of the nodes in their environment (either relative or absolute);
- the environmental parameters in each node's vicinity (i.e., what each node can sense); and
- the changes in all the above: nodes and the communication network, locations in geographic space, and environmental parameters.

This chapter builds up to a detailed model of a geosensor network in three stages. First we construct a basic model of a geosensor network which contains only minimal spatial information about *neighborhoods* in the communication network (§2.2). Then we add more sophisticated models of spatial location (§2.3). Finally, we add time to the mix, allowing modeling of not only where things are, but also of how they change (§2.4).

2.2 Neighborhood-Based Model

The most basic components of a geosensor network are:

- the nodes themselves, with their names or identities;
- the wireless communication network that connects and constrains the movement of information between nodes (i.e., the network neighborhood); and
- the values of environmental parameters that can be detected by sensors at each node.

Thus in the basic neighborhood-based model we ignore temporal issues, such as changes to the environment over time, and almost all spatial issues, such as the coordinate locations of the nodes.

At this point, we shall use some structures and syntax borrowed from discrete mathematics in order to precisely define the different components of a geosensor network. As a result, some familiarity with basic discrete mathematics structures (e.g., sets, relations, and functions) will be an advantage. Appendix A provides a brief discrete mathematics primer to these key concepts. However, for more information the reader is referred to any introductory discrete mathematics textbook (e.g., [53, 74]).

We can represent these basic components of a geosensor network formally as:

1. a *graph* $G = (V, E)$, which models the network and its connections;
2. a function $s : V \rightarrow C$, which models what a node can sense; and
3. a function $id : V \rightarrow \mathbb{N}$, which models the identities or names of each node in the network.

The set of vertices V of the graph represents nodes of the sensor network. The set of edges E in the graph represents direct, one-hop communication links between nodes. In this book we use the terms “node” and “link” to refer to sensor nodes and communication links, while “vertex” and “edge” are used to specifically refer to the formal, graph-based structures that are used to model sensor nodes and communication links. The graph itself is termed the *communication graph*. An undirected communication graph is used to model the situation where all communication links are bidirectional. A directed communication graph allows for the possibility that some communication links are unidirectional (i.e., where direct communication from v_1 to v_2 does not imply direct communication from v_2 to v_1). As we shall see in later chapters, for simplicity an undirected communication graph is often assumed; however, in reality communication between nodes may not be bidirectional.

The only spatial information available in the neighborhood-based model is that which is embedded in the communication graph (i.e., which nodes are adjacent to which other nodes). This information may be represented using the function $nbr : V \rightarrow 2^V$, where $nbr(v) \mapsto \{v' \in V \mid (v, v') \in E\}$. Note, however, that the neighborhood function nbr contains no information that is not already available from the communication graph G — nbr is merely a notational convenience.

The function s , termed the *sensor function*, represents the data sensed by nodes in the geosensor network. The codomain C for the function s will depend on the sensors used. In many of our later discussions, the codomain C is assumed to be simply the real numbers \mathbb{R} . However, more sophisticated sensors and sensor arrays can also easily be represented in the neighborhood-based model. For example, imagine a sensor network where nodes are equipped with sensors for temperature (measured in degrees centigrade), humidity (measured from 0% to 100% relative humidity), and light intensity (normally measured in lux, varying for example, from 0 lux for total darkness to 100,000 lux or more for bright sunshine). This sensed

data would be represented as the domain $C = \mathbb{R} \times [0, 100] \times \mathbb{R}^+$. For a particular node $v \in V$, writing $s(v) = (-4.3, 67, 500)$ would indicate that node v 's sensors detected a temperature of -4.3°C (cold), relative humidity of 67% (comfortable), and light intensity of 500 lux (typical indoor conditions).

Finally, the function $id : V \rightarrow \mathbb{N}$, termed the *identifier function*, represents the identities or names of the nodes in the geosensor network. Without loss of generality, we model the set of names or identifiers as the natural numbers \mathbb{N} (i.e., $\{1, 2, 3, \dots\}$). For example, for some node $v \in V$, $id(v) = 5$ specifies that v 's identifier is 5. In many (but not all) cases, we may wish model the situation where each node has a *unique* identifier. This uniqueness constraint can be represented by specifying that the id function is an *injection* (i.e., where each element in \mathbb{N} is mapped to by at most one $v \in V$). In this book, we assume the id function is injective unless otherwise stated. Although it is important to distinguish between a node's identity and the node itself in many decentralized spatial algorithms, this book will occasionally refer to the node v (instead of the identity of the node $id(v)$) in contexts where no ambiguity exists.

Table 2.1 summarizes informally the different symbols used for a basic geosensor network.

2.2.1 Example Neighborhood-Based Model

To illustrate, consider the graph $G = (V, E)$ in Fig. 2.1. The communication graph G has 25 vertices, $V = \{a, b, \dots, x, y\}$, and 42 edges, $E = \{\{a, b\}, \{a, f\}, \dots, \{w, x\}, \{x, y\}\}$. We imagine that the nodes can sense a Boolean value, either *black* or *white* (indicated by the color of the nodes in Fig. 2.1). Thus, the sensor function may be represented as $s : V \rightarrow \{\text{black}, \text{white}\}$, where $s(x) \mapsto \text{white}$ if $x \in \{h, i, l, m, n, q\}$ and $s(x) \mapsto \text{black}$ otherwise. Finally, an (injective) identifier function could be constructed to map nodes to (unique) node identifiers $1, \dots, 25$ as $id : V \rightarrow \mathbb{N}$, where $id(a) = 1, id(b) = 2, \dots, id(y) = 25$.

The only spatial information available about the network concerns the neighborhoods of nodes, a direct consequence of the communication graph. For example, $nbr(g) = \{b, f, h, l\}$ and $nbr(o) = \{j, n, t\}$.

2.3 Extended Spatial Model

Extending the neighborhood-based model of a geosensor network requires a mechanism to represent more sophisticated aspects of the node's spatial location, beyond basic neighborhoods in the communication graph. Unfortunately, this representation is complicated by the fact that there are many different types of location information an individual node may possess. The process of determining the locations of sensor nodes in a geosensor network is termed *localization*. "Location" in this context does not necessarily

Formal definition	Summary
$G = (V, E)$	Communication graph.
V	Set of nodes in a geosensor network. Lowercase letters v, v' , v_1 , etc. are used to refer to individual nodes in V .
E	Set of communication links between pairs of nodes in a geosensor network. A particular communication link l from v_1 to v_2 is written $l = (v_1, v_2)$ (unidirectional) or $l = \{v_1, v_2\}$ (bidirectional).
$nbr : V \rightarrow 2^V$	The neighborhood of a node, where $nbr(v)$ refers to the set of nodes that is in the <i>neighborhood</i> of $v \in V$ (i.e., those nodes within direct one-hop communication distance from v).
$s : V \rightarrow \mathbb{R}$	Sensor function sensing a particular environmental parameter. For example, $s(v) = 10.3$ indicates that the on-board sensor of node $v \in V$ senses environmental value of 10.3.
$id : V \rightarrow \mathbb{N}$	Identifier function for nodes. For example, $id(v) = 5$ indicates that a particular node $v \in V$ has identifier (or “name”) 5.

Table 2.1. Summary of basic geosensor network model structures

imply the *coordinate location*. *Positioning* is the term for the special case of localization that generates *coordinate* location (i.e., position). However, localization may also involve less detailed quantitative information about the relative distances or directions (bearings) between nodes, or even qualitative information about a node’s proximity to other nodes or known locations. Localization is a highly active area for current research. In general, localization techniques can be classified into *passive* or *active* techniques [119]. Active techniques rely on the active transmission of signals (such as radio frequency or ultrasound signals) from other nodes or beacons; passive techniques do not require active transmission and instead detect “naturally” occurring signals (cf. active and passive sensors in Table 1.1). Examples of active techniques include *lateration* (computing location based on distances to known locations) and *angulation* (computing location based on angles to known locations), as well as *proximity* systems (which determine the closest neighbors). GPS, for example, is a lateration-based, active positioning technique. Examples of passive techniques include *scene analysis* (determining location from analysis of digital camera images) and *dead reckoning* (determining displacement of

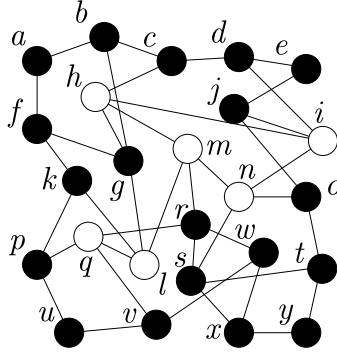


Fig. 2.1. Example neighborhood-based model, with graph $G = (V, E)$, where $V = \{a, b, \dots, x, y\}$ and $E = \{\{a, b\}, \{a, f\}, \dots, \{w, x\}, \{x, y\}\}$, sensor function $s : V \rightarrow \{\text{black}, \text{white}\}$, and identifier $id : V \rightarrow \mathbb{N}$, where $id(a) = 1, id(b) = 2, \dots$

mobile nodes, for example, through inertial tracking). For more information about localization, see [54, 60, 119].

From the perspective of the spatial model of geosensor networks, there are two top-level classes of spatial information generated by localization:

- *absolute location* is spatial information about the location of a sensor node that is referenced to some external system. This most common external reference system is a geodetic framework, where coordinate location can be determined, termed a *positioning system*. However, other external reference systems are possible, such as referencing to known locations within a transportation network (such as known intersections or kilometer posts, a process known as *stationing*).
- *relative location* is spatial information about the location of a sensor node that is referenced to the locations of other nearby sensor nodes.

Table 2.2 summarizes five of the most common types of location information available to sensor nodes in a geosensor network, giving their formal definitions alongside an informal summary. Absolute location information generally concerns either coordinate location (for example, in two or even three dimensions), or close proximity to some known locations, termed “anchor locations” (such as intersections in a transportation network). GPS, for example, is potentially able to provide absolute coordinate location for nodes. Alternatively, RFID (radio frequency identification) tags, attached to mobile nodes, can provide absolute location in terms of proximity to RFID readers at known locations.

Relative location information might include information about the distances from a node to its neighbors; the quantitative bearings from a node to its neighbors; or qualitative bearings of a node’s neighbors, in terms of

the (counter)clockwise sequence of neighbors around a node (termed “cyclic ordering”). Figure 2.2 summarizes diagrammatically these different types of location information, in addition to relative neighborhood location, the most basic type of spatial information assumed to be available to all geosensor networks (see §2.2).

Type	Method	Definition	Summary
Relative	Neighborhood distance	$dist : E \rightarrow \mathbb{R}$	$dist(v, v')$ refers to distance between a node v and its neighbor v' , where $(v, v') \in E$ (Fig. 2.2d).
	Neighborhood bearing	$bear : E \rightarrow \mathbb{R}$	$bear(v, v')$ refers to the bearing of node v' from node v , where $(v, v') \in E$ (Fig. 2.2e).
	Cyclic ordering	$cyc : E \rightarrow V$	$cyc(v, v')$ refers to the next neighbor of v in an anticlockwise direction from v' (Fig. 2.2f).
Absolute	Coordinate location (position)	$p : V \rightarrow \mathbb{R}^d$	$p(v)$ refers to the coordinate location of node $v \in V$ (Fig. 2.2c). In most cases in this book, we assume planar coordinates (i.e., $d = 2$).
	Anchor location	$anch : V \rightarrow A$	$anch(v)$ refers to the <i>anchor</i> location of a node (i.e., the nearest anchor). A is a set of anchors at known locations (which may or may not include nodes in V , Fig. 2.2b).

Table 2.2. Examples of common types of location information in geosensor networks (cf. Fig. 2.2)

As might be expected with spatial information, the different types of location information listed in Table 2.2 and Fig. 2.2 are interconnected. Information about the absolute (coordinate) locations of nodes and the neighborhoods of a node can be combined to compute (using standard geometry) information about the neighborhood distances and bearings of nodes. Further, information about the neighborhood bearings can be used to compute the cyclic ordering of nodes (the cyclic ordering is a less precise, qualitative version of the quantitative neighborhood bearing).

2.3.1 Example Spatial Model

Building on the neighborhood-based model example in §2.2.1, consider now the graph $G = (V, E)$ in Fig. 2.3. The communication graph, sensor function, and identifier function in Fig. 2.3 are identical to those in Fig. 2.1. However, the

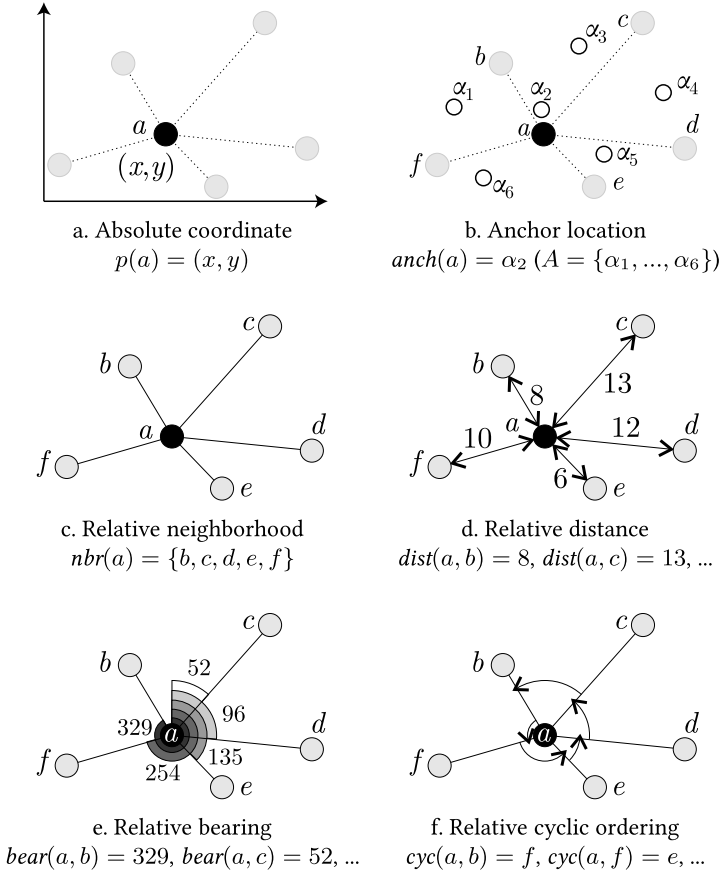


Fig. 2.2. Summary of common types of location information available to a node

nodes are no longer arbitrarily placed, but are arranged in a grid to emphasize their spatial locations.

In a spatial model, we might have access to any or all the levels of spatial information given in Table 2.2 (and potentially more). For example, assume each edge has length 1 unit. It then follows that:

- the neighborhood distance $dist : E \rightarrow \mathbb{R}$ has assignment mapping $dist(v_1, v_2) \mapsto 1$;
- the neighborhood bearing $bear : E \rightarrow \mathbb{R}$ has assignment mapping:

$$bear(v_1, v_2) \mapsto \begin{cases} 0 & \text{if } id(v_2) < id(v_1) - 1 \\ 90 & \text{if } id(v_2) = id(v_1) + 1 \\ 180 & \text{if } id(v_2) = id(v_1) - 1 \\ 270 & \text{if } id(v_2) > id(v_1) + 1 \end{cases}$$

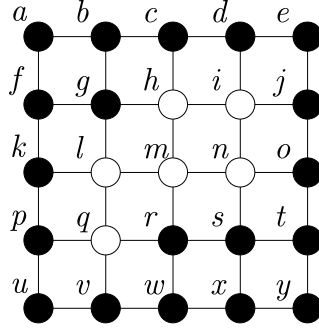


Fig. 2.3. Example spatial model, assuming an identical communication graph to that in Fig. 2.1, but with planar coordinate embedding such that all edges have length 1 and node u is located at the origin $(0, 0)$

- the (anticlockwise) cyclic ordering $\text{cyc} : E \rightarrow V$ has assignment mapping $\text{cyc}(a, b) = f$, $\text{cyc}(a, f) = b$, $\text{cyc}(b, a) = g$, $\text{cyc}(b, g) = c$, $\text{cyc}(b, c) = a$, and so forth.

If we additionally add the assumption of a planar coordinate system, with node u at the origin, we may further specify the position of each node as:

- the planar positioning function $p : V \rightarrow \mathbb{R}^2$ has assignment mapping $p(a) = (0, 5)$, $p(b) = (1, 5)$, $p(c) = (2, 5)$, ..., $p(y) = (5, 0)$.

Note that as long as we have the highest level of information, the planar positioning function p , all the other levels of relative location information may be deduced from p .

2.4 Spatiotemporal Model

The two models discussed so far have been *atemporal*, in the sense that they are static and make no reference to change over time. As we shall see in later chapters, in some situations it is reasonable to use such simplified models. However, in general, geosensor networks monitor changing phenomena and may themselves be subject to change. It is possible to identify three main types of change that are important to decentralized spatial computing:

- Environmental dynamism*: Geosensor networks are usually tasked with monitoring geographic environments that change; static environments are both rare in the world (try to think of a place that does not change) and by definition not especially interesting to monitor with geosensor networks (a single survey or a photograph would be enough to record a static environment).

- *Node mobility*: Nodes in a geosensor network may not be static, and may instead change their position over time. Node mobility may occur through movement of a host to which a node is attached (e.g., nodes attached to people, vehicles, animals, etc.) or through more purposeful mobility (e.g., robotic nodes capable of motorized movement or opportunistic mobility using the movement of the wind or waves).
- *Node volatility*: Nodes in a geosensor network may be *volatile* in the sense that they may activate, deactivate, or reactivate over time. Changes in activation may be deliberately controlled by the nodes or the network (e.g., duty cycling) or may be by-products of other processes (e.g., nodes deactivate due to technical failures or depletion of energy resources, or nodes are activated by new nodes being introduced into the network). Mobility can be seen as a special case of volatility, where volatile nodes may deactivate but immediately reactivate in a new location.

As a result, in most cases, it is important additionally to be able to model changes in the environment and the geosensor network. Informally, this is achieved by extending each of the structures described in the previous two sections, covering the neighborhood-based and extended spatial models, with time-varying capabilities. Table 2.3 summarizes the extended time-varying structures. Environmental dynamism can be modeled using a time-varying sensor function ($s : V \times T \rightarrow \mathbb{R}$) which represents the different data generated by each node's sensors over time. The only structure we assume cannot change is the identifier function; node identities are assumed to persist through time.

2.4.1 Example Spatiotemporal Model

Figure 2.4 shows an example of a spatiotemporal model of a geosensor network, building on the structures in §2.2.1 and §2.3.1. In this simple example we model only environmental dynamism: the communication graph and nodes are assumed to be static. Thus, the sensed value for node c , for example, changes from time t_2 to t_3 : $s(c, t_1) = \text{black}$, $s(c, t_2) = \text{black}$, $s(c, t_3) = \text{white}$.

More sophisticated spatiotemporal models must additionally deal with change over time in the absolute or relative locations of nodes (mobility) and/or in the active set of nodes (volatility). For example, Fig. 2.5 shows a spatiotemporal model with two mobile nodes, with absolute locations referenced to intersections in a traffic network (anchors). Thus, in contrast to earlier figures, the network in Fig. 2.5 is the *transportation* network, not the communication network. The changing location of the mobile node v_1 , for example, is given by: $\text{anch}(v_1, t_1) = n_9$, $\text{anch}(v_1, t_2) = n_{10}$, $\text{anch}(v_1, t_3) = n_{15}$.

Atemporal structure	Temporal extension	Informal summary
$s : V \rightarrow \mathbb{R}$	$s : V \times T \rightarrow \mathbb{R}$	$s(v, t)$ refers to the value sensed by node v at time t .
$p : V \rightarrow \mathbb{R}^n$	$p : V \times T \rightarrow \mathbb{R}^n$	$p(v, t)$ refers to the coordinate location of node v at time t .
$anch : V \rightarrow V_{anch}$	$anch : V \times T \rightarrow V_{anch}$	$anch(v, t)$ refers to anchor location of node v at time t .
$nbr : V \rightarrow V$	$nbr : V \times T \rightarrow V$	$nbr(v, t)$ refers to the set of neighbors of node v at time t .
$dist : E \rightarrow \mathbb{R}$	$dist : E \times T \rightarrow \mathbb{R}$	$dist(v, v', t)$ refers to the distance between v and its neighbor v' at time t .
$bear : E \rightarrow \mathbb{R}$	$bear : E \times T \rightarrow \mathbb{R}$	$bear(v, v', t)$ refers to the bearing of neighbor v' from v at time t .
$cyc : E \rightarrow nbr(v)$	$cyc : E \times T \rightarrow V$	$cyc(v, v', t)$ refers to the next neighbor of v in an anticlockwise direction from v' at time t .
$G = (V, E)$	$G(t) = (V, E(t))$	For mobile (but not volatile) geosensor networks, $G(t)$ refers to the communication graph at time t , $E(t)$ refers to the set of links in G at time t .
$G = (V, E)$	$G(t) = (V(t), E(t))$	For volatile (including volatile and mobile) geosensor networks, $G(t)$ refers to the communication graph at time t , $E(t)$ refers to the set of links in G at time t , and $V(t)$ refers to set of nodes at time t .

Table 2.3. Extending static definitions to allow for change over a totally ordered set T of discrete times (including informal descriptions for particular time $t \in T$)

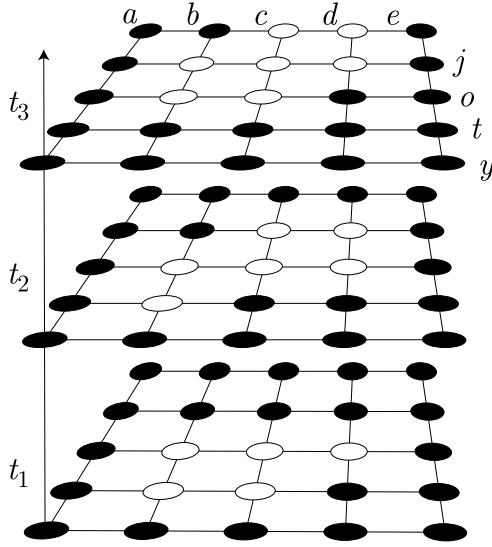


Fig. 2.4. Example spatiotemporal model for modeling environmental dynamism ($s : V \times T \rightarrow \{\text{black}, \text{white}\}$) using a static communication graph, $G = (V, E)$, for times $T = \{t_1, t_2, t_3\}$

2.5 Partial Knowledge

The formal structures introduced above provide an overarching framework for representing geosensor networks with a range of different characteristics. However, let us now turn away from overarching frameworks and instead consider the individual nodes. The information available to a particular node will typically differ from that in the structures already introduced in two very important respects:

1. *Partial knowledge*: By default, a node in a geosensor network is expected to have information only about its *own* state. Any information about the state of other nodes in the system must be explicitly communicated to that node. Thus, we must be able to represent the situation where nodes have only *partial* knowledge of the state of the entire network.
2. *Uncertain knowledge*: Uncertainty is an endemic feature of geospatial information. Thus, we must also be able to represent the situation where a node's knowledge of its geographic environment may in some ways diverge from the "ideal."

The issue of uncertain knowledge is vital, but will be held in abeyance until Part III, Chapter 8, when we shall address this issue directly. Until then we shall make the simplifying assumption that nodes have ideal location and environmental sensors which can make perfect observations of geographic space.

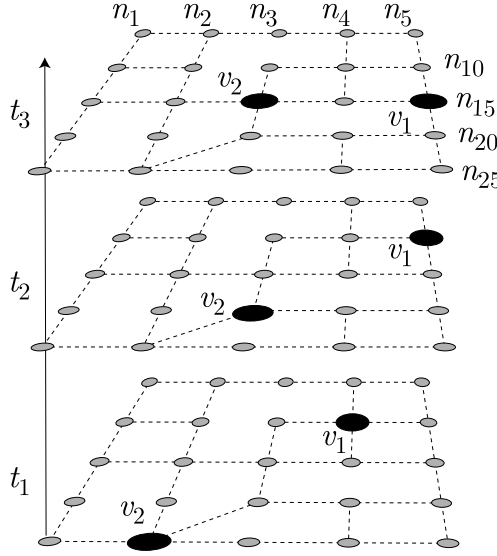


Fig. 2.5. Example spatiotemporal model of mobile nodes ($anch : V \times T \rightarrow A$), using the static nodes $A = \{n_1, \dots, n_{25}\}$ of a transportation network as anchors, and times $T = \{t_1, t_2, t_3\}$

The issue of partial knowledge, however, does need to be addressed directly at this point. Some of the most frequent mistakes in decentralized spatial algorithm design arise from mistakes in distinguishing between information that is *local* to an individual node and that which is *global* to the entire system. Only information local to a node (such as that node's sensor data or its location) can be *directly* accessed by that node. All other data must be explicitly communicated to a node. Thus, one of the keys to successful decentralized algorithm design is carefully and rigorously to identify a node's *local* information—its partial knowledge of the entire system.

The formal model set out above makes no distinction between local and global information. Functions are defined at the global level: for example, the sensor function $s : V \rightarrow C$ is defined globally, across all nodes in the network. An individual node $v \in V$ can only access a small part of that function directly, the piece of information that relates to its own sensor function, $s(v)$.

To distinguish between a global function and a node's local knowledge about that function requires some additional notation. For some global function $f : V \rightarrow C$ we will use the overdot notation, \dot{f} (read “my” f or “local” f), to refer to an arbitrary (unspecified) node's local knowledge of that function. Thus, for any function f with domain V , \dot{f} is interpreted to mean $f(\circ)$ for some node $\circ \in V$ that is clear from the context.

This approach can be directly extended to those functions with a domain that is a product set (such as spatiotemporal algorithms). Any function with a product set as its domain, such as $g : V \times T \rightarrow C$, can be rewritten as an equivalent chain of single-argument functions, such as $g : V \rightarrow (T \rightarrow C)$ (a transformation known as “Currying”). Thus, we may write $\hat{g}(t)$ (“my g of t ”) in place of $g(\circ, t)$ for some arbitrary $\circ \in V$ and time $t \in T^1$.

For example, where we might write globally $s(\circ) = 10$, inside a decentralized algorithm we instead write $\hat{s} = 10$ (“my s is 10”). Similarly, in a spatiotemporal algorithm, instead of writing the global statement “ $s(\circ, t) = 10$ ” to indicate that node $\circ \in V$ senses the value 10 at time t , we will use the corresponding local function inside a protocol, $\hat{s}(t) = 10$. Finally, in a spatial algorithm $\hat{dist}(v) = 12$ may be used in place of $dist(\circ, v) = 12$.

In this way, direct references to global functions can be avoided in the specification of algorithms, which helps to ensure that only local information is used in the algorithm. Information that is not part of the local knowledge of a node must be explicitly communicated to that node before it can be used.

2.6 Neighborhood Structure

One final important component of our formal foundations remains only partially defined: the precise neighborhood structure of the communication graph. In “computing somewhere” the neighborhood structure is a key constraint on the movement of information. The communication graph $G = (V, E)$ specifies for each node $v \in V$ those nodes that are capable of direct communication with v (given by $nbr(v)$, defined as the set $\{v' \mid \{v, v'\} \in E\}$; see §2.2). This section examines more closely some of the most common neighborhood structures that can constrain communication in a geosensor network, or indeed in any decentralized spatial information system.

2.6.1 Unit Disk Graph (UDG)

In a geosensor network the constraints on the movement of information are imposed by the communication network structure, which in turn is constrained by the relative locations of the nodes. Our model has so far considered the spatial locations of the nodes and the structure of the network separately. However, because of the limited communication range of nodes (see §1.2.2), the structure of the communication network connecting

¹ More specifically, given the function $g : V \rightarrow (T \rightarrow C)$, the result of applying the function g to \circ , $g(\circ)$, yields a function $h : T \rightarrow C$. In turn, this function h can be applied to some time t , $h(t)$. Informally, Currying ensures that any functions with multiple arguments can be decomposed into a chain of single argument functions, each applied in sequence.

geosensors is in practice inherently dependent on the spatial location of the nodes.

The most basic network structure used to represent this dependency is the *unit disk graph* (UDG). The UDG begins by assuming nodes have a uniform communication range, c (the “unit distance”). The UDG is then the communication graph that is formed by creating edges to connect all nodes that are within unit distance c (i.e., that are within direct communication range). Figure 2.6 illustrates an example UDG for a set of nodes and an arbitrary unit distance c .

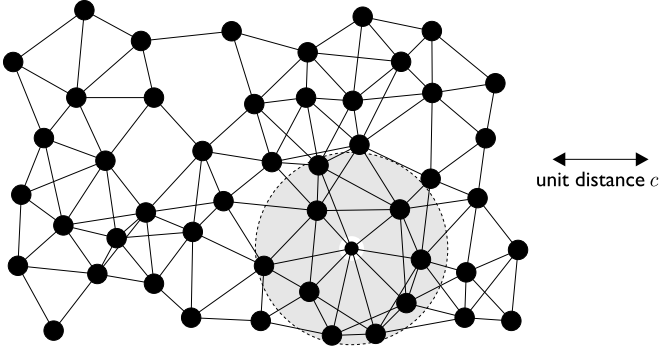


Fig. 2.6. Example unit disk graph (UDG), highlighting the unit distance communication range of one particular node (gray circle)

For a given set of nodes V , node positions $p : V \rightarrow \mathbb{R}^2$, and unit distance c , the UDG is formally defined as:

$$\text{UDG} = (V, \{\{u, v\} \mid (u, v) \in V \text{ and } 0 < \delta(p(u), p(v)) \leq c\})$$

where $\delta(a, b)$ is the Euclidean distance between two points a and b .

It is important to note at this point that the UDG is a model of the communication graph that results from the *physical* constraints of limited power radio frequency communication. Consequently, the UDG is often assumed as the default structure for a network, even in cases where nodes have no capability for localization or other knowledge of their coordinate location (i.e., the *locator* function $p : V \rightarrow \mathbb{R}$ is not known to the nodes). In other words, the UDG is assumed to result from the nodes' actual position in geographic space, even in cases where they have no knowledge of that position.

Two implicit assumptions behind the UDG are:

- communication is bidirectional; and
- the communication range is uniform across the network.

In practice, neither of these assumptions may hold. Environmental conditions, as well as varying energy resources of nodes, will often result in different nodes in a network having different communication distances. In turn, this leads to unidirectional communication, where the fact that a node can transmit directly to another does not imply that it can receive messages communicated from that node. Of course, where the varying communication ranges of different nodes are known or can be estimated, a (directed) communication graph can still be constructed to represent that situation.

The UDG represents a common baseline assumption about communication in a geosensor network, and is used as the foundation of several other neighborhood structures, discussed further in the following sections. All of these other network structures are *spanning subgraphs* of the UDG. A spanning subgraph is a graph that contains all the vertices but a subset of edges of another graph. More formally, for a graph $G = (V, E)$ a spanning subgraph is a graph $G' = (V, E')$ such that $E' \subseteq E$. This property is important because a communication graph that is *not* a spanning subgraph of the UDG necessarily contains one or more edges that are not contained within the UDG (and so connects nodes that are too far apart to communicate directly).

Topology control

A natural question that follows the discussion of the UDG is “What happens if we adjust the unit distance?” In the extreme case where the unit distance c is very large, we obtain the *complete* graph of the vertices V (the graph where an edge exists between every pair of nodes). As already discussed, such situations are not especially interesting in the context of decentralized spatial computing, since they present no constraints on the movement of information (any node can communicate directly with any other node). Further, the fundamental resource constraints of geosensor networks means that they are also in practice not commonly encountered.

Moving to the other extreme, for any graph there exists a unique minimum unit distance c such that the communication graph remains connected (i.e., there exists a path connecting any pair of nodes, and as a result it would be possible to multi-hop information from any node to any other node). In this case, the unique minimum unit distance is termed the critical transmission range (CTR). The UDG resulting from using the CTR as the unit distance is called the minimum radius graph (MRG). The MRG for the node set in Fig. 2.6 is illustrated in Fig. 2.7. Any further reduction in the unit distance c will lead the graph to become disconnected.

As a result of the direct relationship between the distance of communication and the power required to communicate (see §1.2.2), topology control has clear practical implications for geosensor networks. At least in theory, the MRG provides a mechanism for setting globally a minimum energy

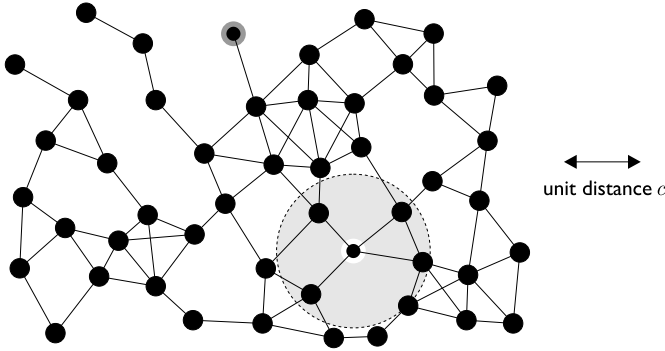


Fig. 2.7. Minimum radius graph (MRG) for the nodes from Fig. 2.6. The node highlighted in gray will become disconnected by any further reductions in the unit distance c

level for wireless transceivers that still ensures nodes can engage in (multi-hop) communication. In practice, the simplifying assumptions of the UDG (discussed above), and the desire for improved robustness in communication by allowing for some redundancy in communication paths, means that a network would normally aim to operate with a unit distance c that is some way above the CTR.

2.6.2 Plane and Planar Graphs

The structure of the UDG is determined spatially, based on the coordinate locations of the nodes and the unit distance. However, many of the algorithms we will look at in later chapters require further spatial structure. In particular, a common requirement concerns the *planarity* of the communication graph. A *planar* graph is a graph that can be embedded in the plane in such a way that no edges cross (i.e., edges only intersect at nodes). A *plane* graph is a planar graph plus a particular planar embedding that ensures that no edges cross. In short, a planar graph *could* be drawn on a sheet of paper without any crossing edges; a plane graph (or a *planar map*) is a specific drawing of a planar graph that has no crossing edges. The graphs in Figs. 2.6 and 2.7 are clearly not plane (they have several crossing edges), and in fact are non-planar (i.e., it is impossible to redraw the figure in a way that preserves all nodes and connections but removes crossing edges).

Four of the most important plane graph structures commonly used with geosensor networks are explored in the following subsections. As already highlighted, each of these structures is a spanning subgraph of the UDG.

Delaunay triangulations

A triangulation is a *maximal plane graph*: a plane graph where no edge can be added between existing nodes without making the graph non-planar.

There are many different possible maximal plane graphs for a set of vertices in the plane. One special maximal plane graph, the *Delaunay triangulation*, has the defining property that a circumcircle through the vertices of any triangle contains no other vertices. Figure 2.8 illustrates this property. More intuitively, this property ensures the Delaunay triangulation has the “fattest” triangles of all triangulations. The Delaunay triangulation is an important structure in GIScience, with a great many applications and articles in the literature.

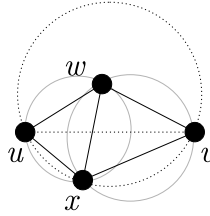


Fig. 2.8. The Delaunay triangulation has the defining property that a circumcircle through the vertices of any triangle contains no other vertices. The dashed circumcircle contains vertex w , and so the edge uv is not in the Delaunay triangulation

Unfortunately, the Delaunay triangulation (or indeed any maximal plane subgraph) is not guaranteed to be a subgraph of the UDG. Any maximal plane graph may have arbitrarily long edges. (Imagine, for example, a set of vertices in the shape of a crescent: any triangulation of these points will necessarily require an edge between the two extreme tips of the crescent). In the context of geosensor network, arbitrarily long edges translate into arbitrarily large amounts of power required to transmit directly between two nodes.

For a given set of vertices, the unit Delaunay triangulation (UDT) is the intersection of the UDG and the Delaunay triangulation (i.e., the graph formed by the edges that are contained in both the UDG and the Delaunay triangulation of the vertices). Figure 2.9 shows the UDT derived from the UDG in Fig. 2.6. Note that there are many possible edges that could be added to the graph in Fig. 2.9 without leading to a non-planar result: in fact the UDT is not a triangulation at all!

Gabriel graph

The Gabriel graph (GG) has the property that an edge exists between two nodes $u, v \in V$ only if there is no node $w \in V$ such that the angle formed by uwv is greater than 90° . Equivalently, this condition is more easily framed using Pythagoras' theorem rather than trigonometry, as follows (where E_{UDG} is the set of edges in the UDG for a given unit distance):

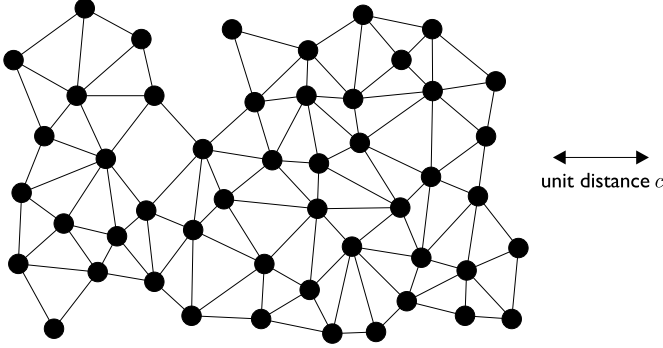


Fig. 2.9. Example unit Delaunay triangulation (UDT) (cf. Fig. 2.6)

$$\text{GG} = (V, \{(u, v) \in E_{\text{UDG}} \mid \text{for all } w \in V, \\ \delta(p(u), p(w))^2 + \delta(p(w), p(v))^2 \geq \delta(p(u), p(v))^2\}).$$

Figure 2.10 illustrates this definition graphically. Figure 2.11 shows the GG derived from the UDG in Fig. 2.6.

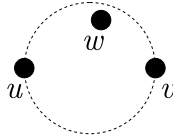


Fig. 2.10. The Gabriel graph has the property that a pair of nodes u and v may only be connected by an edge if there exists no other node w such that the angle formed by uvw is greater than 90° (i.e., there exists no w located in the dashed circle)

Relative neighborhood graph

The relative neighborhood graph (RNG) has the property that two nodes $u, v \in V$ will not be connected by an edge if there exists a node $w \in V$ where w is closer to u and to v than u and v are to each other. Formally,

$$\text{RNG} = (V, \{(u, v) \in E_{\text{UDG}} \mid \text{for all } w \in V, \\ \delta(p(u), p(w)) \geq \delta(p(u), p(v)) \text{ or } \delta(p(w), p(v)) \geq \delta(p(u), p(v))\})$$

where E_{UDG} is the set of edges in the UDG for a given unit distance. Figure 2.12 illustrates this definition graphically. Like the GG, the RNG is a

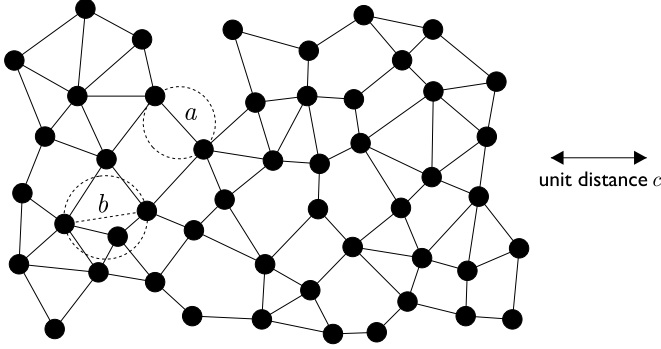


Fig. 2.11. Example Gabriel graph (GG), highlighting examples of (a) an edge maintained from the UDG; and (b) an edge removed from the UDG (cf. Fig. 2.6)

spanning subgraph of the UDG. As could be deduced by comparing Figs. 2.10 and 2.12, the RNG is also a spanning subgraph of the GG (since any edge excluded by the GG condition will also be excluded by the more restrictive RNG condition). Figure 2.13 shows the RNG derived from the UDG in Fig. 2.6.

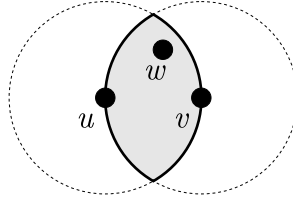


Fig. 2.12. The relative neighborhood graph has the property that any pair of nodes u and v will not be connected by an edge if there exists a node w that is closer to both u and v than u and v are to each other (i.e., there exists no w located in the highlighted lune)

2.6.3 Trees

A *tree* is a graph where any pair of vertices is connected by exactly one path (sequence of adjacent nodes). Figure 2.14 shows an example of a tree that is a spanning subgraph (subtree) of the UDG. For a graph such as the UDG in Fig. 2.6 there are many possible spanning subtrees (approximately 1×10^{30} —a “nonillion”—to be more precise; the exact number can be computed using a result in graph theory called *matrix tree theorem*). In fact, the tree shown in Fig. 2.14 is a special, uniquely defined tree, called the *minimum spanning tree* (MST). The minimum spanning tree has the property that it

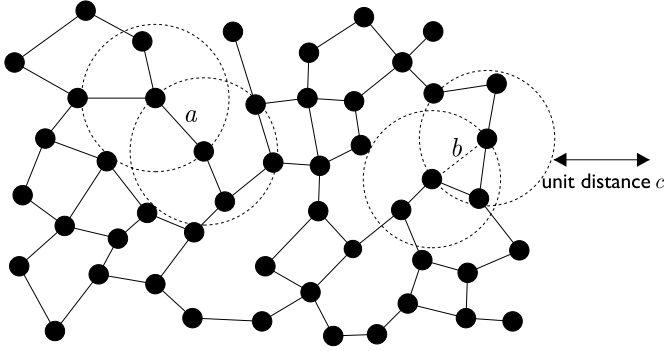


Fig. 2.13. Example relative neighborhood graph (RNG), highlighting examples of a , an edge maintained from the UDG; and b an edge removed from the GG (cf. Figs. 2.6 and 2.11)

is the spanning tree in the graph that has the minimum possible total edge length (i.e., adding up the lengths of all the edges in the graph in Fig. 2.14 results in a total length less than any other spanning tree of the UDG in Fig. 2.6). The minimum spanning tree also happens to be a spanning subtree of the RNG (and by implication the GG).

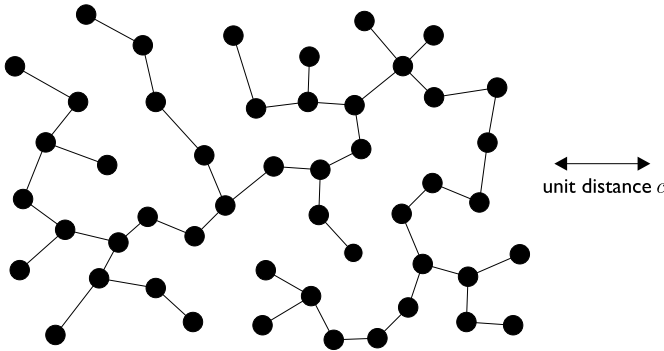


Fig. 2.14. Example minimum spanning tree (MST) (cf. Fig. 2.6)

Trees are commonly used because they provide a convenient neighborhood structure for wireless sensor networks. A *rooted tree* has one designated node as its *root*. Because a tree by definition contains no cycles (since every pair of nodes is connected by exactly one path), edges in a rooted tree have a natural orientation (towards or away from the root). As we shall see, this feature can be helpful in decentralized computation, as it provides a natural structure for aggregating and centralizing data.

2.7 Chapter in a Nutshell

The formal foundations set out in this chapter are founded on three interconnected models (Fig. 2.15). At the core, the formal foundations depend on the neighborhood-based model of nodes, their sensors, and their communication network. Building on this core, the extended spatial model depends on the neighborhood-based model, and introduces more sophisticated representations of the spatial locations of nodes. In turn, the spatiotemporal model depends on the neighborhood-based or the extended spatial model, further extending the core by representing change in the network and in the environment over time.

In addition, two auxiliary models are added. First, we define a model of the partial knowledge of a node, restricting the information available at each node to that which it senses directly or which is explicitly communicated to that node. Second, a neighborhood model provides more details about the different neighborhood structures that may be used to constrain communication between nodes. A third auxiliary model, allowing for uncertainty in the information sensed by a node, will be added later (Chapter 8).

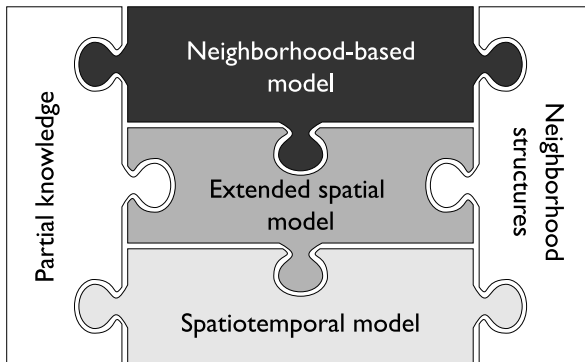


Fig. 2.15. Summary of formal model structure

These foundations deliberately suppress the technical or application details of sensor networks. The aim is to provide a basis for discussing decentralized spatial computing independently of changing node architectures and software, sensor technologies, batteries and energy harvesting, routing protocols, and application requirements.

Importantly, the formal foundations are the basis for all subsequent chapters in the book. Not all these structures are required at all times. Indeed, the models are designed to be deployed in stages, supporting increasingly sophisticated design and analysis of decentralized spatial algorithms. In some decentralized spatial information systems, only the extended spatial model

(and by implication the neighborhood-based model) are required. But, based on the extended spatial model, any combination of the remaining two core models can be “plugged in” depending on the specific application requirements.

It is worth highlighting that there are several features our basic model has not yet addressed, including:

- *Node heterogeneity*, where nodes possessing different spatial and environmental sensing capabilities are combined in the same network. Homogeneous networks are assumed in Parts I and II of this book, with an exploration of the important issues connected with node heterogeneity discussed in Chapter 9.
- *Three-dimensional networks*, such as might be deployed for monitoring marine environments, vegetation structure, or the atmosphere. Although it is straightforward to include 3D coordinate locations in the positioning of nodes, many of the later structures in this chapter (such as planar graphs) rely on a two-dimensional representation of geographic space. Extending algorithms to monitor environments in three spatial dimensions (for example, additionally incorporating information about the vertical profile of changes in sea temperature and salinity across a coral reef) remains an important challenge for future work.
- *Asynchronous networks*, where different sensors cannot be relied upon to sample at the same time or frequency. Extended models of spatiotemporal networks that can allow for asynchronous sensing are discussed in Chapter 9.

Armed with our formal foundations, we are almost ready to start exploring the design and analysis of decentralized spatial algorithms, capable of processing spatial information with no centralized control. However, before that, we must first turn our attention to the computational characteristics of decentralized spatial computing. These algorithmic foundations are the topic of the next chapter.

Review questions

- 2.1 Under a certain condition, it will always be the case that for a communication graph $G = (V, E)$, $v \in nbr(v')$ assuming $v' \in nbr(v)$. What is this condition? (Hint: see §2.2).
- 2.2 Table 2.2 provides five examples of common methods for specifying location in geosensor networks. What other examples of different methods for specifying relative and absolute location information can you think of?
- 2.3 Using a Web search, try to find different technologies for localization that are capable of generating the information required for the different methods for specifying location in Table 2.2 and in your answer to

question 2.2 above. (Hint: to start you off, GPS is one example of a technology for generating coordinate location—but there are others!)

- 2.4 In §2.3, the different methods for specifying location are categorized into two types: absolute and relative. However, one might also classify location information as quantitative (concerned with the *measurable* aspects of space) and qualitative (referring to discrete and meaningful *categories* or “qualities”). Reclassify the different methods for specifying location in Table 2.2 and in your answer to question 2.2 above into quantitative and qualitative location information.
- 2.5 Node mobility can be seen as a special case of node volatility. Explain in your own words why this is so. (Hint: see §2.4).
- 2.6 The Yao graph is another graph structure. In a Yao graph, the space around each node is partitioned into sectors of fixed opening angle (e.g., 60°). Each node is then connected to its the unique nearest neighbor (if any) in each of these sectors [120]. Is the Yao graph a spanning subgraph of the UDG? Check your answer by sketching a Yao graph for the nodes in Fig. 2.6.
- 2.7 We have stated that the relative neighborhood graph is a (spanning) subgraph of the Gabriel graph. Construct a proof (or find and understand an existing proof) that this must always be the case (i.e., that the RNG is necessarily a subgraph of the GG).



<http://www.springer.com/978-3-642-30852-9>

Decentralized Spatial Computing
Foundations of Geosensor Networks
Duckham, M.
2013, XXII, 322 p., Hardcover
ISBN: 978-3-642-30852-9