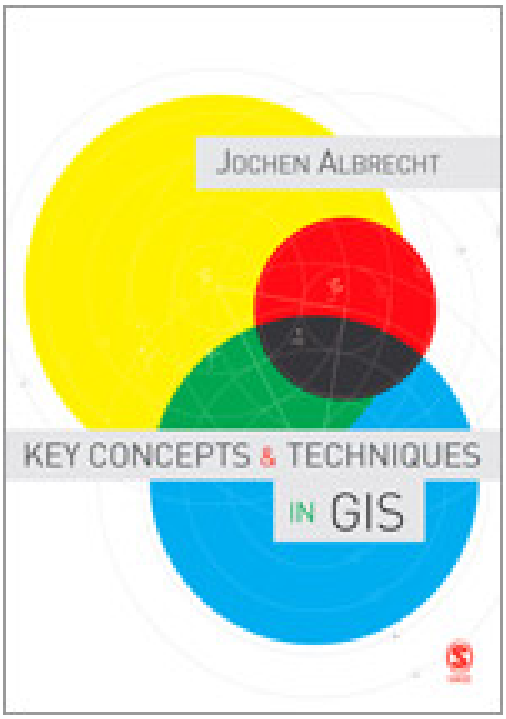


Sage Research Methods

Key Concepts and Techniques in GIS

For the most optimal reading experience we recommend using our website. A free-to-view version of this content is available by clicking on this link, which includes an easy-to-navigate-and-search-entry, and may also include videos, embedded datasets, downloadable datasets, interactive questions, audio content, and downloadable tables and resources.

Author: Jochen Albrecht

Pub. Date: 2011

Product: Sage Research Methods

DOI: https://doi.org/10.4135/9780857024442

Methods: Geographic information systems, Artificial neural networks, Databases

Keywords: overlay, layer
Disciplines: Anthropology, Geography
Access Date: July 21, 2025
Publisher: SAGE Publications Ltd
City: London
Online ISBN: 9780857024442

Combining Spatial Data

As mentioned in the previous chapter, many spatial relationships are difficult to derive or even describe. Rather than storing all possible relationships between all features of a database, we can use GIS operations to answer specific questions about the spatial relationships among our features of interest. This is not only far more efficient but also gives us more freedom, because we can determine on the fly what pieces of data we want to relate to each other and in the age of the Internet, these pieces may even be distributed across the world.

This chapter deals with two families of GIS operations that in practice make up some 7580% of all analytical GIS operations. ***Overlay*** is the quintessential GIS operation that seems to define a GIS. If a software package can perform overlay operations, then it is indeed a GIS and not a mere CAD or cartography program. The poor cousin is the *buffer* operation, which always seems to be mentioned second. Both are actually placeholders for a number of different operations, but we will discuss this in detail in the following.

6.1 Overlay

In the recoding section of the last chapter, we saw how tightly linked attributes and geometries are. By recombining attributes we automatically changed the graphical representation as well. However, when we look at it from the perspective of lines store in our database (see Figure 20), the recoding operation did not create any *new* geometries. This changes now with the group of overlay operations. Let's look at Figure 23 to see what happens to geometries and attributes in an overlay operation.

For pedagogical reasons, we use very simple geometries and only two layers of binary data. This is extremely unrealistic but helps us to get the principle across. We will look at more realistic examples later on.

The figure shows a number of important aspects of the overlay operation. We have two or more input feature classes and one (new) output feature class. The geometries of the two input layers are most likely to be different; that is, they do not have to come from the same provider and do not have to have anything in common other than the general extent (it does not make sense to overlay data in South America with other data from Africa). Also observe that the first feature class has no attributes describing vegetation, while the second has

none describing soils. The overlay operation depicted here looks for coincidences at the same location. In other words, for any given location, it looks what information there is in one feature class, then in the other feature class, and then it combines the two in the output dataset. The geometries of the output dataset did not exist before. We really create completely new data.

Figure 23 Schematics of a polygon overlay operation

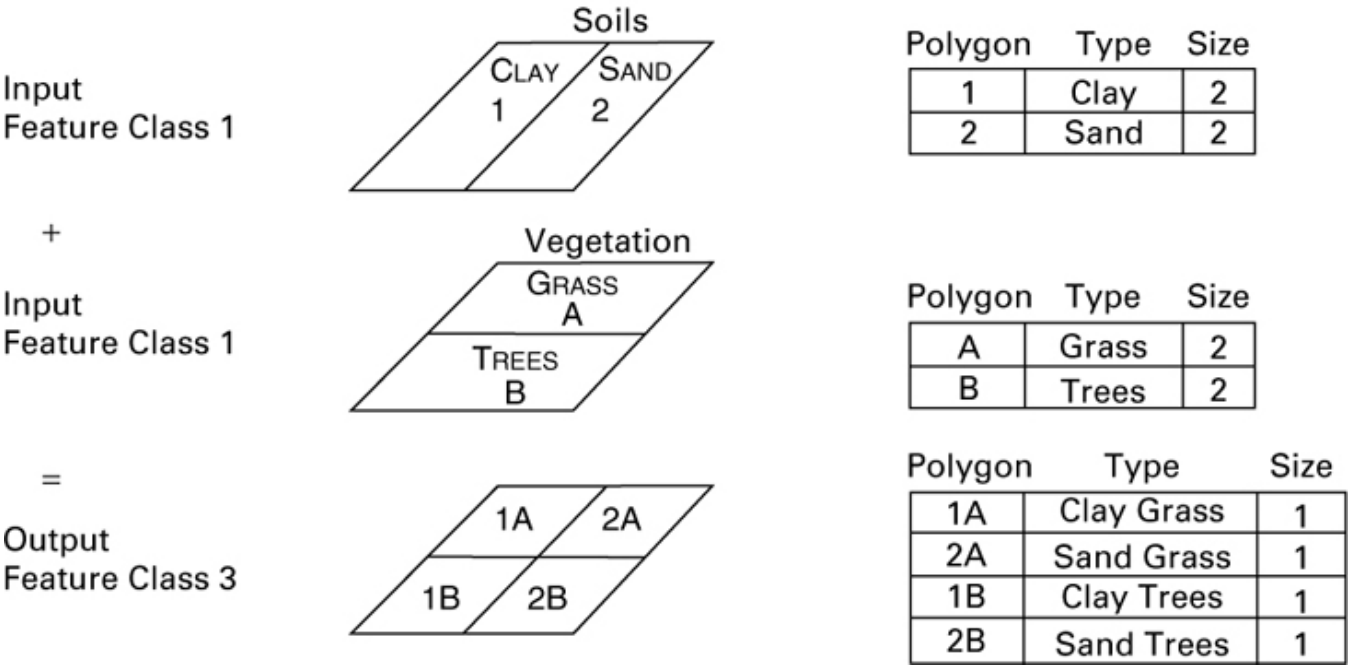
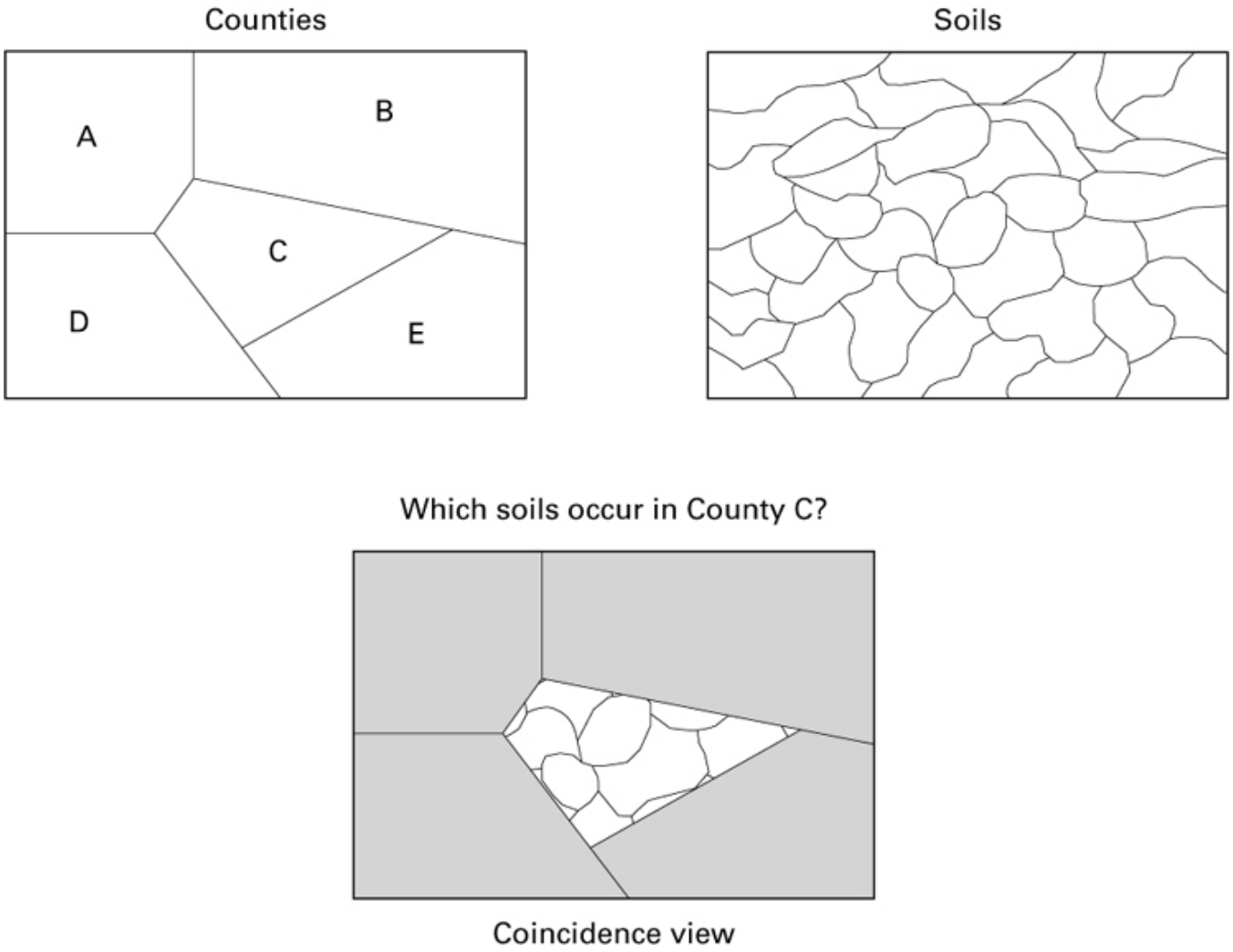
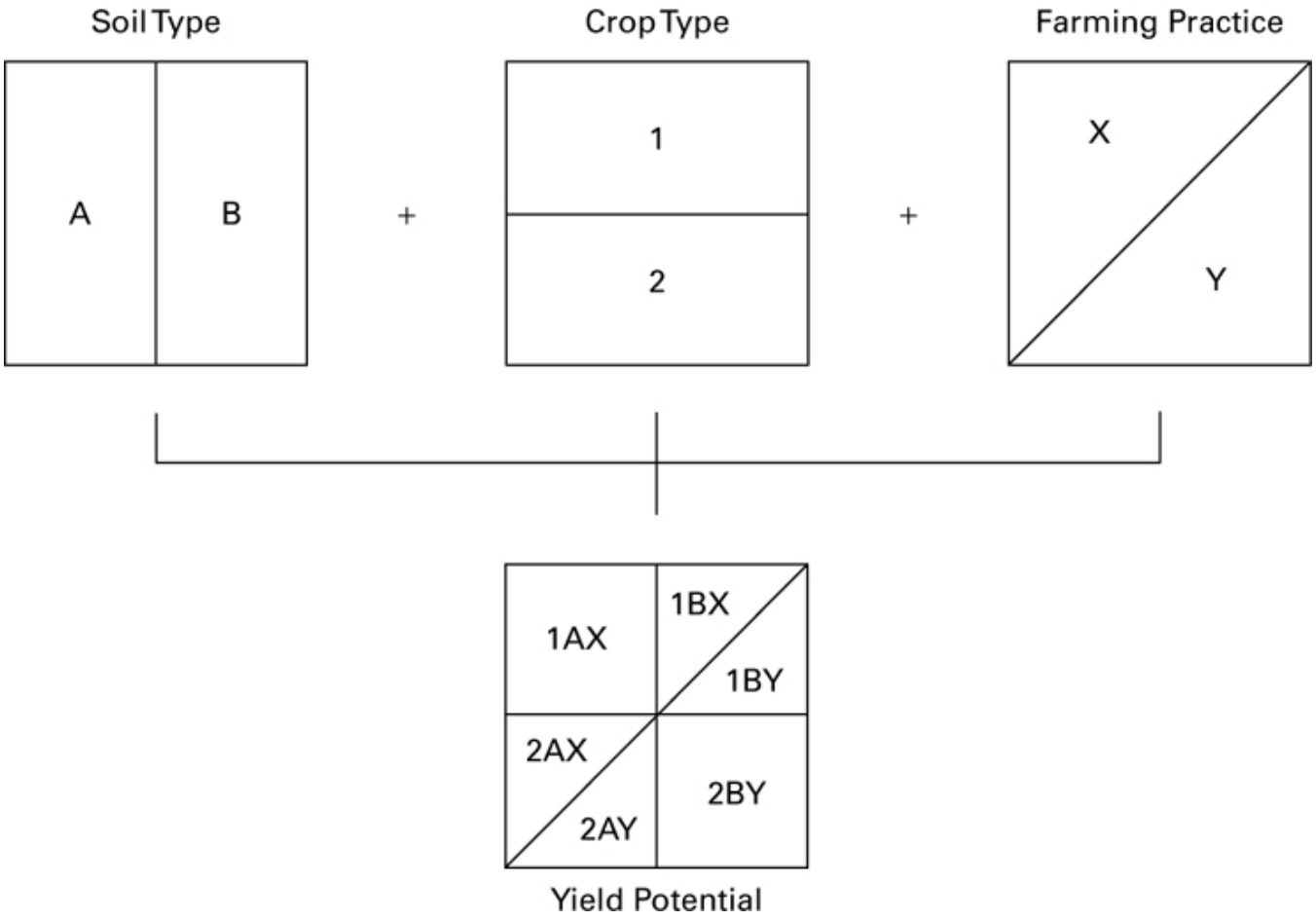


Figure 24 Overlay as a coincidence function



We used this notion of coincidence earlier when we looked at some of the more advanced spatial search functions. Figure 24 is a case in point.

Figure 25 Overlay with multiple input layers



The question ‘Which soils occur in county C?’ appears to be a simple spatial search function. However, what actually happens here is that we overlay the two feature classes and then look for those parts of the soils feature class that fall within the polygon that marks county C in the counties feature class. In Chapter 4, we did not have an answer to how to deal with those soil polygons that lie only partially within county C. Now, we applied the overlay operation like a cookie cutter and created several new pieces of data. When you look back at Figure 23, you see that there is quite a lot happening here. New files or at least tables get created, new geometries have to be calculated, new connections between attributes and geometries have to be formed and maintained. This may seem like a lot for a simple spatial search operation and it is! Keep this in mind, the next time you wait impatiently for the result of your spatial search.

What we did in Figure 24 was to overlay one layer of areal features with another one. A slightly more complicated (realistic) example is depicted in Figure 25. Here we are using three input layers and possibly some weighting scheme to calculate agricultural yield potential. Going beyond the pure area-on-area situation, we

could just as well overlay areal with point or line feature classes, for example to determine which historic site is in what administrative unit or to ascertain all the countries that the river Nile is flowing through. We could even overlay point and line feature classes to learn whether the Christopher St subway station is on the red or the blue line (see Figure 21). In each of these cases we make use of the topological relationships between all the features involved.

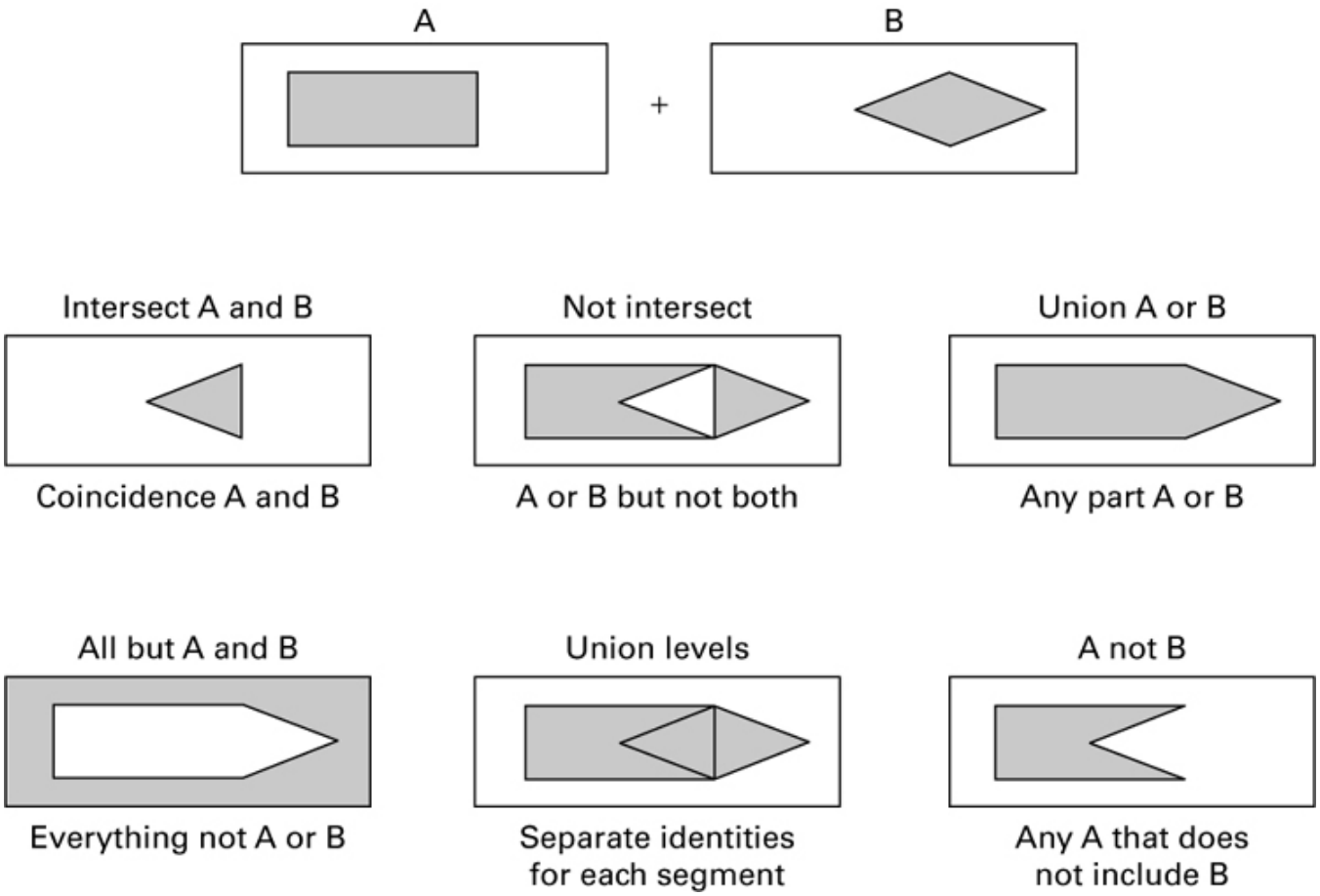
6.2 Spatial Boolean logic

In Chapter 4, we looked briefly at Boolean logic as the foundation for general computing. You may recall that the three basic Boolean operators were NOT, AND and OR. In Chapter 4, we used them to form query strings to retrieve records from attribute tables. The same operators are also applicable to the combination of geometries; and in the same way that the use of these operators resulted in very different outputs, the application of NOT, AND and OR has completely different effects on the combination (or overlay) of layer geometries.

Figure 26 illustrates the effect of the different operands in a single overlay operation. This is why we referred to overlay as a group of functions. Figure 26 is possibly the most important in this book. It is not entirely easy to digest the information provided here and the reader is invited to spend some time studying each of the situations depicted. Again, for pedagogical reasons, there are only two layers with only one feature each. In reality, the calculations are repeated thousands of times when we overlay two geographic datasets. What is depicted here is the resulting geometry only. As in the example of Figure 23 above, all the attributes from all the input layers are passed on to the output layer.

Depending on whether we use one or two Boolean operators and how we relate them to the operands, we get six very different outcomes. Clearly one overlay is not the same as the other. At the risk of sounding overbearing, this really is a very important figure to study. GIS analysis is dependent on the user understanding what is happening here and being able to instruct whatever system is employed to perform the correct overlay operation.

Figure 26 Spatial Boolean logic



The relative success of the overlay operations can be attributed to their cognitive consonance with the way we detect spatial patterns. Overlays are instrumental in answering questions like ‘What else can be observed at this location?’, or ‘How often do we find woods and bison at the same place?’.

6.3 Buffers

Compared to overlay, the buffer operation is more quantitative if not analytical. And while, at least in a raster-based system, we could conceive of overlay as a pure database operation, buffering is as spatial as it gets. Typically, a buffer operation creates a new area around our object of interest although we will see exotic exceptions from this rule. The buffer operation takes two parameters: a buffer distance and the object around which the buffer is to be created. The result can be observed in Figure 27.

A classical, though not GIS-based, example of a buffer operation can be found in every larger furniture store. You will invariably find some stylized or real topographic map with concentric rings usually drawn with a felt pen that center on the location of the store or their storage facility. The rings mark the price that the store charges for the delivery of their furniture. It is crude but surprisingly functional.

Regardless of the dimension of the input feature class (point, line or polygon), the result of a regular buffer operation is always an area. Sample applications for points would be no-fly zones around nuclear power plants, and for lines noise buffers around highways. The buffer distance is usually applied to the outer boundary of the object to be buffered. If features are closer to each other than the buffer distance between them, then the newly created buffer areas merge as can be seen for the two right-most groups of points in Figure 27.

There are a few interesting exceptions to the general idea of buffers. One is the notion of inward buffers, which by its nature can only be applied to one-or higher-dimensional features. A practical example would be to define the core of an ecological reserve (see Figure 28). A combination of the regular and the inverse buffer applied simultaneously to all features of interest is called a corridor function (see Figure 29). Finally, within a street network, the buffer operation can be applied along the edges (a one-dimensional buffer) rather than the often applied but useless as-the-crow-flies circular buffer. We will revisit this in the next chapter.

Figure 27 The buffer operation in principle

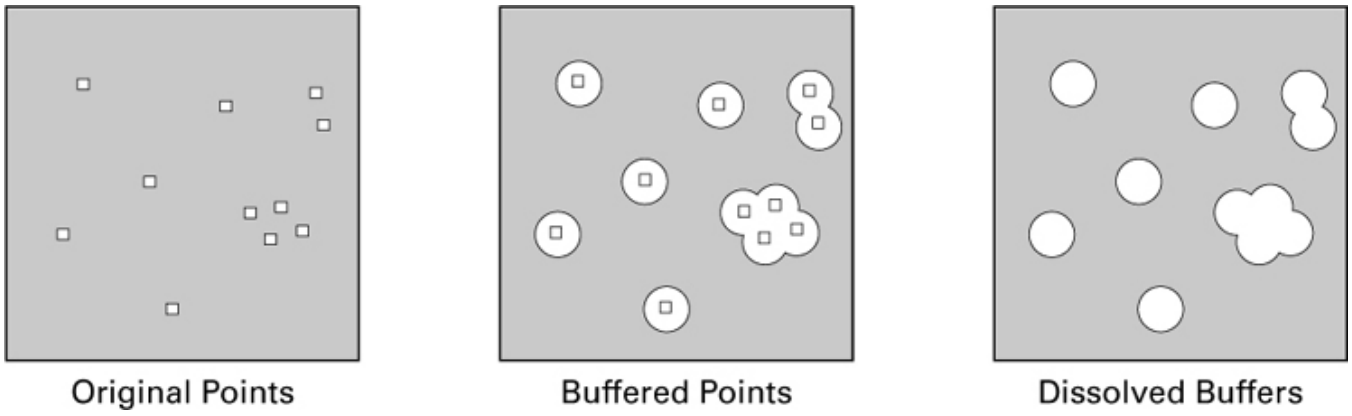


Figure 28 Inward or inverse buffer

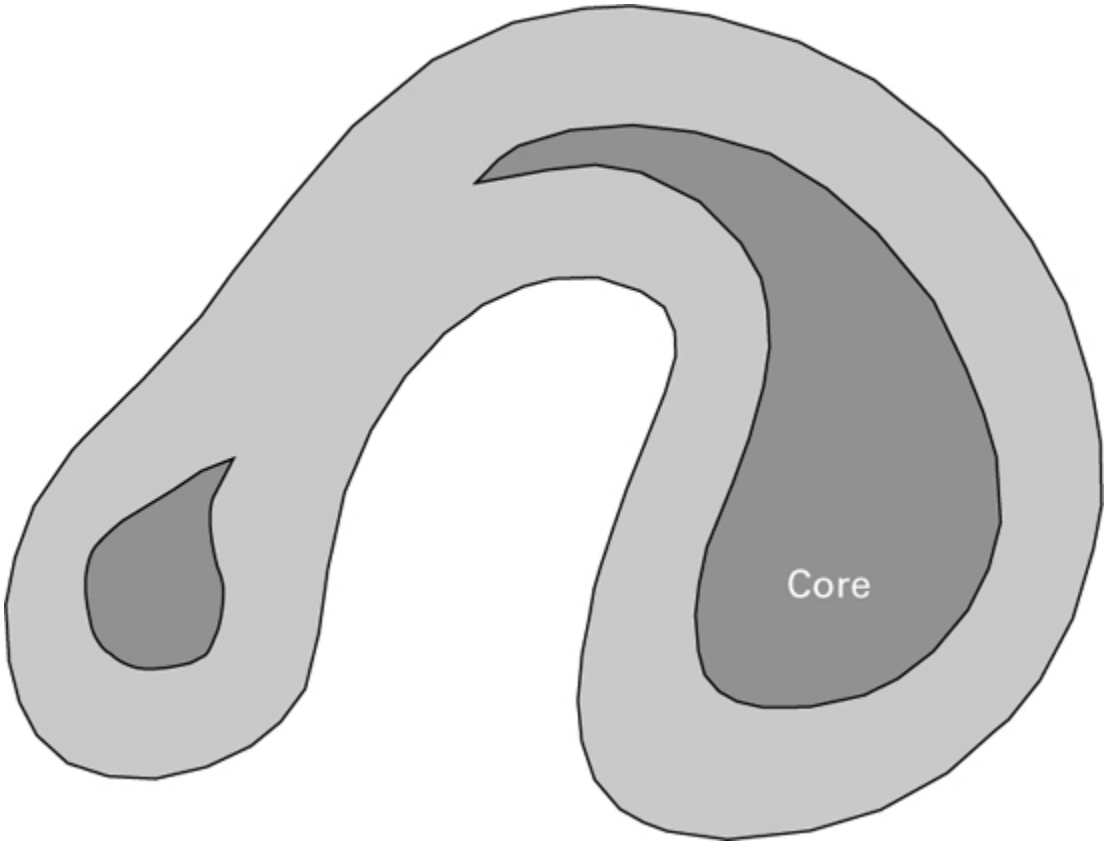
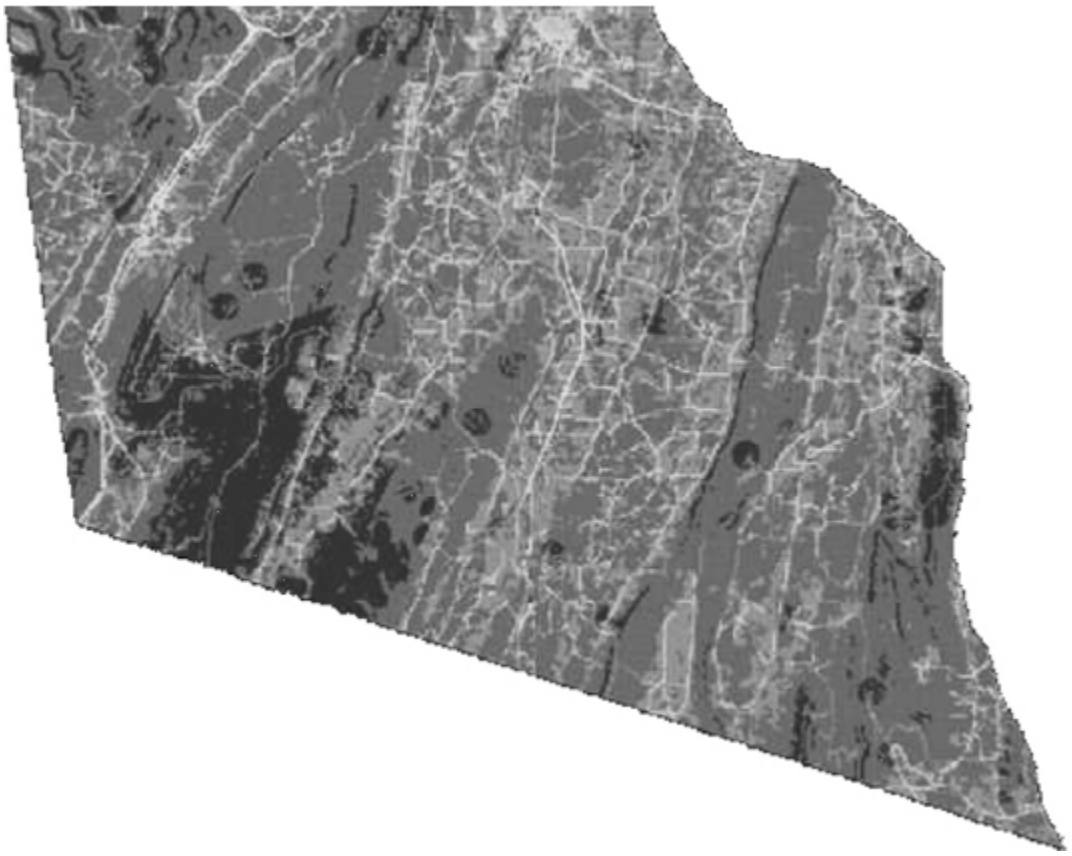


Figure 29 Corridor function



6.4 Buffering in spatial search

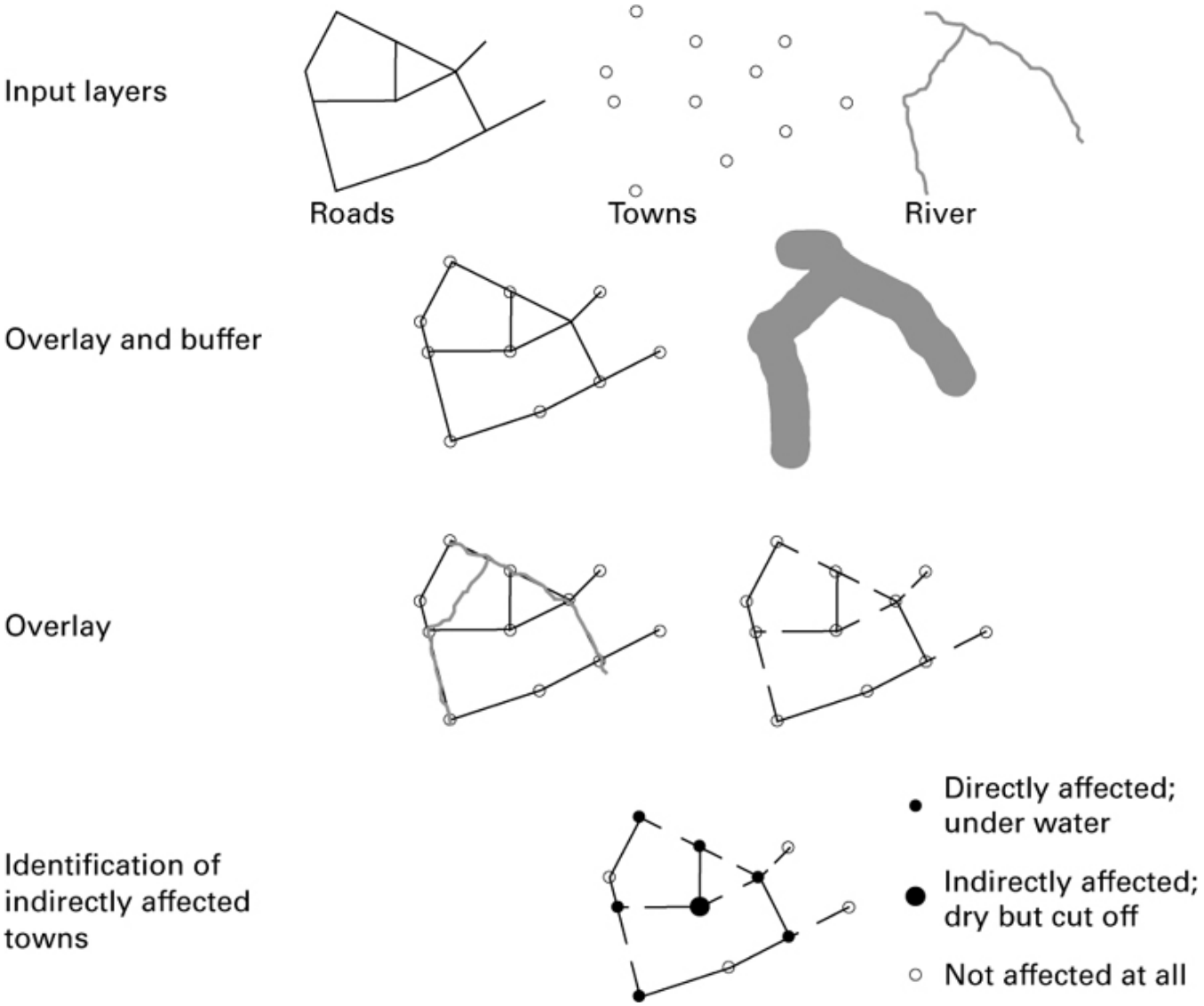
A few paragraphs above we saw how overlay underlies some of the (not overtly) more complicated spatial search operations. The same holds true for buffering. Conceptually, buffers are in this case used as a form of neighborhood. ‘Find all customers within ZIP code 123’ is an overlay operation, but ‘Find all customers in a radius of 5 miles’ is a buffer operation. Buffers are often used as an intermediate select, where we use the result of the buffer operation in subsequent analysis (see next section).

6.5 Combining operations

If the above statement that buffers and overlays make up in practice some 75% of all analytical GIS function-

ality is true, then how is it that GIS has become such an important genre of software? The solution to this paradox lies in the fact that operations can be concatenated to form workflows. The following is an example from a major flood in Mozambique in 2000 (see Figure 30).

Figure 30 Surprise effects of buffering affecting towns outside a flood zone



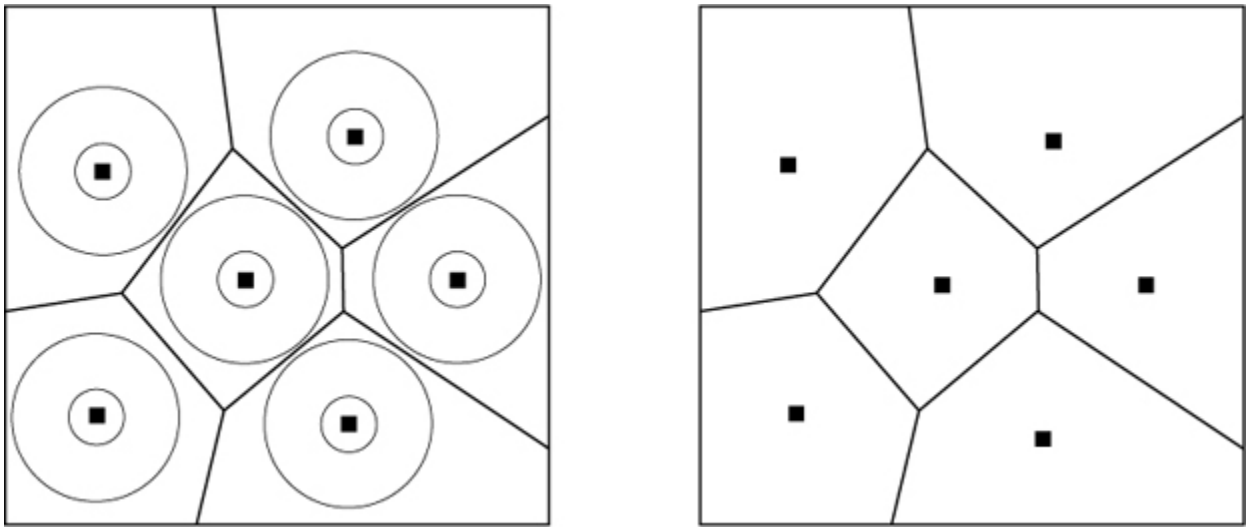
We start out with three input layers towns, roads and hydrology. The first step is to buffer the hydrology layer to identify flood zones (this makes sense only in coastal plains, such as was the case with the Southern African floods in 2000). Step two is to overlay the township layer with the flood layer to identify those towns that are directly affected. Parallel to this, an overlay of the roads layer with the flood layer selects those roads that have become impassable. A final overlay of the impassable roads layer with the towns helps us to iden-

tify the towns that are *indirectly* affected that is, not flooded but cut off because none of the roads to these towns is passable. Figure 30 is only a small subset of the area that was affected in 2000.

6.6 Thiessen polygons

A special form of buffer is hidden behind a function that is called a **Thiessen polygon** (pronounced the German way as 'ee') or Voronoi diagram. Originally, these functions had been developed in the context of graph theory and applied to GIS based on triangulated irregular networks **(TIN**s), which we will discuss in Chapter 9. It is introduced here as a buffer operation because conceptually what happens is that each of the points of the input layer is simultaneously buffered with ever-increasing buffer size. Wherever the buffers hit upon each other, a 'cease line' is created until no buffer can increase any more. The result is depicted in Figure 31.

Figure 31 Thiessen polygons



Each location within the newly created areas is closer to the originating point than to any other one. This makes Thiessen polygons an ideal tool for allocation studies, which we will study in detail in the next chapter.

https://doi.org/10.4135/9780857024442