



Remsoft Inc.
332 Brunswick Street,
Fredericton, NB,
Canada E3B 1H1

1-800-792-9468 or
1-506-450-1511

www.remsoft.com

**GIS 96
Symposium**

**Vancouver
British Columbia**

February 1996

Defining adjacency and proximity of forest stands for harvest blocking

Karl R. Walters, Associate
Forest Management Systems
Remsoft Inc.

332 Brunswick Street, Fredericton
NB, Canada E3B 1H1
1-800-792-9468 or 1-506-450-1511
email: waltek@mindspring.com

Abstract

Harvest blocking is a fundamental decision of timber management. The size, location and timing of harvest blocks directly affects the development and structure of the forest at the current time, but for future time periods as well. In developing automated blocking tools, we have found that obtaining the required adjacency and proximity information to implement these tools properly can be problematic. Part of the problem lies with semantics -- in general, we all know what is meant by adjacency and proximity. Depending on the task at hand (blocking versus scheduling) it takes some forethought about the relationships among stands to avoid adjacency and maximum opening size constraint violations.

In many cases, the required adjacency or proximity information cannot be readily obtained from the geographic information system. For example, using a vector-based system, it is not a trivial task to determine whether two harvest blocks composed of several stands each are more than a minimal distance apart without displaying them and selecting the two closest points visually. A raster system is more amenable to this problem, but suffers from the inefficiencies of many more allocation units (cells versus stands) and the loss of resolution.

Keywords: spatially constrained harvest scheduling, harvest blocking, adjacency, proximity

Introduction

Harvest blocking is the foundation for modern timber management. With increasing concern for the environment have come a host of rules and regulations that make the blocking process a technically challenging task. In years past, the location of harvest blocks was predicated largely on the volume of wood available and the cost of transportation to the processing site; now, these factors are still important but considerations such as opening size, proximity to other openings, and timing of harvest are critical to ensuring economical and sustained harvest levels. In order to comply with various constraints on the harvest, the analyst charged with developing a blocked harvest schedule faces a daunting challenge.

At Remsoft, we have been studying the problem of spatially constrained harvest scheduling for several years and we have developed some software tools to aid in the process. Obviously, geographic information systems (GIS) play an integral role in the harvest blocking process, but we have found that some information, critical to an automated planning process, is difficult or impossible to obtain from commercial GIS products. The purpose of this paper is to introduce to you the concepts of *adjacency* and *proximity* as we use them at Remsoft, and why they are important to spatial management planning.

Definitions

The American Heritage Dictionary defines adjacency as “the state of being adjacent; contiguity.” and proximity as “the state, quality, sense, or fact of being near or next; closeness.” The definitions themselves are fine, but sometimes the precise meaning of the words has been lost in application. In many jurisdictions, regulations have extended the definition of adjacency to include areas that are not physically contiguous (e.g., any area within 100 metres of a forest opening is considered to be adjacent and is thus ineligible for harvesting) or they have limited the scope of adjacency (e.g., if two harvest block boundaries overlap by less than 100 metres the blocks are not considered adjacent). The word *proximity* tends not to be used in regulatory language, even though it is the basis for virtually all green-up delay requirements.

I don’t intend to write a long discourse on semantics and proper usage, but I think it is a valid point to make that many of the difficulties that arise in trying to comply with rules and regulations are the result of imprecise language. Therefore, to maintain clarity throughout this paper, I will maintain that objects that are physically contiguous are *adjacent* whereas objects that are near each other but not touching are *proximate*.

An example problem

Let us consider a hypothetical management problem where the task is to develop a blocked harvest schedule under the following regulations:

- clearcut harvest blocks may not exceed 120 acres in size and,
- no harvesting is permitted within 300 feet of a clearcut harvest block for a period of 10 years (green-up delay).

These regulations are not nearly as complicated as some of those in other jurisdictions (e.g., Maine or

the Pacific Northwest) but they will be sufficient to make my points.

Forming a harvest unit

In New Brunswick, the forest land base is significantly fragmented due to ownership patterns and past logging or catastrophic events; the average stand size is only about 3 or 4 hectares. To make a feasible harvest unit of, say 20 ha, may require 5 or 6 contiguous stands grouped together as a single harvest unit. To determine the location of such a harvest block, our harvest blocking tool starts with a given stand and looks for adjacent stands to group with it. The program requires a comprehensive list of neighboring stands which is derived from GIS databases. However, as we only discovered recently, individual GIS vendors use different implementations of *contiguity* to determine adjacency.

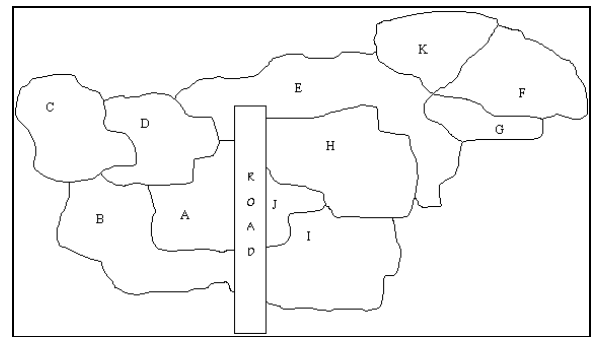


Figure 1. Spatial arrangement of stands in a hypothetical forest.

Consider the spatial arrangement of stands in Figure 1. If we were to create a harvest block composed of 4 adjacent stands, we could use A-B-C-D or E-H-I-J without hesitation. But what about a block composed of stands E-F-H-J? If we use a GIS that defines contiguous polygons to be those with a common *vertex*, then stands E and F are adjacent and the block E-F-H-J is valid. Conversely, if we use a GIS that determines adjacency based on common *arcs*, then block E-F-H-J is not a valid block because polygons E and F do not share a common arc and are thus not considered adjacent.

From a practical standpoint, stands linked by a common point are not really practical components of a harvest block. For example, suppose you have two blocks E-F and G-K. If you were to implement a corridor linking E and F then stands G and K would no longer be contiguous. Although it is probably easy to deal with this problem through manual intervention, it becomes more of a problem with an

algorithmic approach that deals with thousands of blocks at a time. The problem is exacerbated using a raster-based GIS because every cell potentially has 4 single point adjacencies.

Although a harvest unit composed of contiguous stand area is a necessity for most operations, the requirement for contiguity breaks down when a road is involved. Even though stands A, B, I and J are not strictly contiguous, from an operational standpoint, they would make a feasible harvest unit since machinery can readily traverse the road. In this case the stands are not adjacent, but because of the road and their close proximity they can be grouped to form a feasible harvest block. Had stands I and J been located several miles down the road however, the presence of the road would not have justified forming a harvest unit.

Developing a viable harvest unit generally requires contiguous forest stands linked by a common boundary rather than a single point. Obtaining adjacency information that excludes single point adjacencies may be difficult from some geographic information systems. Furthermore, practical considerations such as the possibility of road or stream crossings permit the use of non-adjacent stands in the development of blocks, along with the obvious considerations of stand area to limit opening sizes.

Scheduling harvest units

Once the harvest units have been developed, the timing of harvest for each of them must be determined such that green-up requirements are met. Although one could simply implement buffer strips around each harvest block, there are drawbacks to doing so. Consider Figure 2. First, in the forest on the top, the residual forest in the buffer strips may not yield desirable harvest blocks because they are both long and narrow, and they are not uniform in shape. Second, it requires 5 passes to harvest the entire forest area (one block is half the area of the others). In contrast, the arrangement on the bottom has uniform harvest blocks and requires only 4 passes to harvest the entire forest area (one block is also half the area of the others).

If harvest units are significantly smaller than the maximum opening size, combining 2 or more of these units to create a single harvest block can alleviate some problems with green-up delays. However, as the harvest block increases in size, the greater the extent of forest area tied up due to the green-up requirements. Scheduling the harvest of

these harvest units to minimize the impact of spatial constraints is a computationally difficult problem to solve. Our approach so far has been to combine Monte Carlo integer programming techniques with heuristics to yield good feasible solutions.

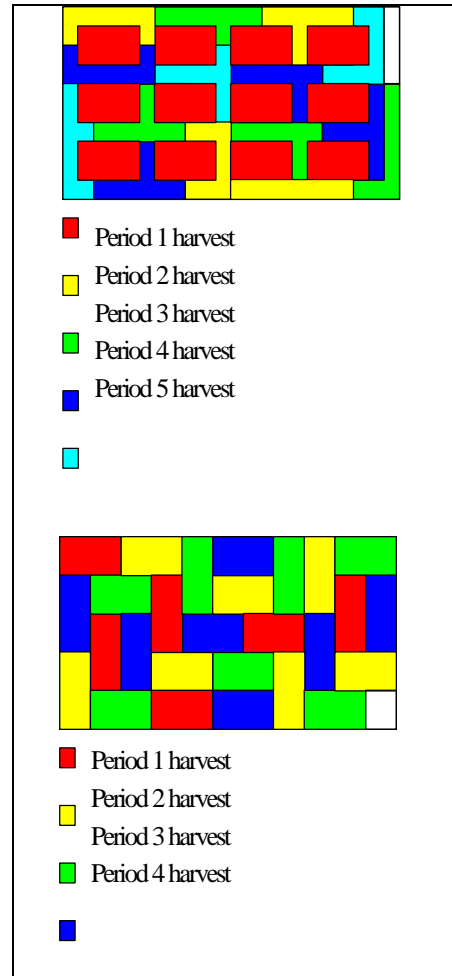


Figure 2. Alternative ways to comply with green-up requirements in blocking.

For the scheduler to work, we must provide it with proximity information on harvest units. Since we know which stands belong to each harvest unit, the stand adjacency information we developed earlier can be used to determine adjacency among harvest units by eliminating *selfs* (adjacency relationships between 2 stands in the same block). Whereas the inclusion of single point adjacencies in the adjacency list for the blocking tool was a disadvantage, including those adjacencies in the scheduling process is desirable.

Consider the polygons illustrated in Figure 3. The equilateral triangles represent the situation where 3 lines cross at a given point. The red polygon has 9

corner point adjacencies and just 3 common arc adjacencies. The square grid represents the situation where 2 lines cross, and there are equal numbers of common arc and corner point adjacencies. The hexagonal grid exhibits no corner point adjacencies at all. Now, if we were to create a harvest block composed of the red polygon and one of its yellow neighbors we will violate the green-up requirements if we do not recognize the single point adjacencies in the triangular and square grids, but the hexagonal grid will present no such difficulties. Furthermore, if the length of one side of a hexagon is equal to the minimum distance allowed between simultaneously harvested blocks, then it is guaranteed that all non-contiguous blocks will be sufficiently spaced from one another.

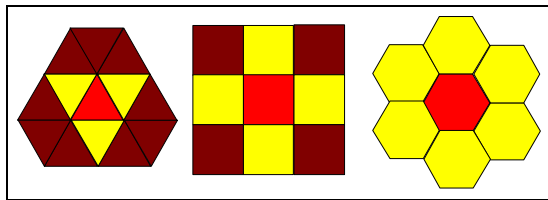


Figure 3. The impact of line intersections on point adjacencies.

Some researchers have used *centroids* to determine proximate blocks because almost all GIS vendors provide ready means of accessing this information. However, centroids are a poor substitute because the green-up restrictions require that all points on the *boundaries* of two polygons be at least a given distance apart, not that the bulk of their *area* be more than a given distance apart. To determine this relationship for all polygons requires point-to-point distance calculations for all the vertices in all arcs – a huge computational effort for a vector-based GIS; although the computational burden for a raster system is considerably less, it is still not trivial.

Some observations

Given that different GIS vendors use either the common point method or the common arc method for determining adjacency in vector-based systems, and that problems arise when either approach is used exclusively for spatial harvest scheduling, we developed a program that scans the nodes of an arc to determine both adjacency and proximity relationships. If two polygons share a common point, they are proximate to each other; if they share two or more common points, they are considered adjacent to one another. In a raster GIS, it is obvious that each cell will have four adjacent neighbors and four

corner-point neighbors. In a vector based system, the number of corner-point proximate polygons depends on how many lines cross each other – a T-intersection generates true adjacencies, not single-point adjacencies. To determine whether the single point adjacencies are significant or not, we processed a Forest Development Survey map from New Brunswick using our adjacency procedure and found that the number of adjacency records fell from 24,720 to 18,738 when single point adjacencies were excluded from the list.

Since the tolerance for comparison can be set to any arbitrary distance, two polygons which have points separated by less than the comparison tolerance (x) are considered to be x -distance proximate to each other. Obviously, the larger the tolerance, the greater the number of comparisons to be made – in fact, the number of comparisons grows exponentially – and so practical limits on processing time require relatively small tolerances. For small tolerance limits, the procedure works quickly and we have used it to generate adjacency relationships among stands that border roads, streams and other line features. We have also used it to establish adjacency relationships among stands on map boundaries which is very useful if maps have not been properly edge-matched or if the routines provided by the GIS are slow to run on a large number of maps.

In the figures that follow, a uniform, square forest was subdivided into a 20 x 20 grid and our automated blocking/scheduling tool was applied to maximize even flow harvest for four planning periods. The minimum harvest block size was one cell and no maximum size limit was established. A one period harvest delay constraint was applied, such that a block adjacent to a block harvested in period one could be harvested no earlier than period three. The algorithm tries to make blocks as large as possible, with shape controls applied to avoid very irregular shaped blocks.

In Figure 4, you can readily determine that the algorithm was quite successful in allocating the harvest to meet the green-up delay requirements. Since corner point adjacencies were not included in the algorithm's adjacency list, it left cells unharvested between the red block in the upper right corner and green block below it, to act as buffers between the two harvest blocks. Similarly, the blue J-shaped block located in the lower middle of the forest is considered a separate block from the nearby larger blue block. However, both of these instances

clearly violate the intent of the adjacency restrictions.

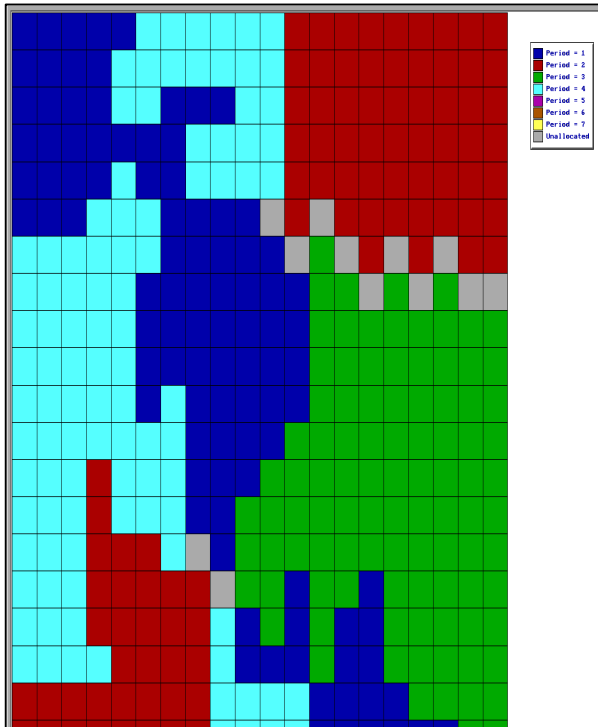


Figure 4. Block configuration 1 - corner point adjacencies omitted.

When the corner point adjacencies were included in the adjacency list the blocking/scheduling algorithm yielded a very different block pattern, as illustrated in Figure 5. Notice that in this block configuration there are no instances where the adjacency restrictions are violated – cells are either left unharvested or there is a minimum one period delay for the intervening cells between blocks. However, the harvest blocks themselves are less desirable than those in Figure 4.

For example, all of the blue cells constitute a single large harvest block, but there are isolated pieces that are problematic from an operational standpoint. Any attempt to connect these small *islands* with the main portion of the block could mean completely severing another island from its parent block – for example, the blue and cyan blocks on the lower boundary of the forest. In other cases, connecting these islands to their parent blocks with corridors would result in adjacency violations as well.

If we extrapolate these results to real world situations, we would expect to find that our blocking/scheduling algorithm would perform well on forests composed of relatively uniform stands of fairly regular shape and with few intersecting lines

(such as property boundaries) without corner point adjacencies considered. The few adjacency violations that resulted could likely be handled easily through manual intervention. Conversely, if the algorithm were applied to a forest composed of many small woodlots or other holdings and/or with highly irregular stand geometries, the number of adjacency violations would increase dramatically, severely limiting the utility of an automated process.

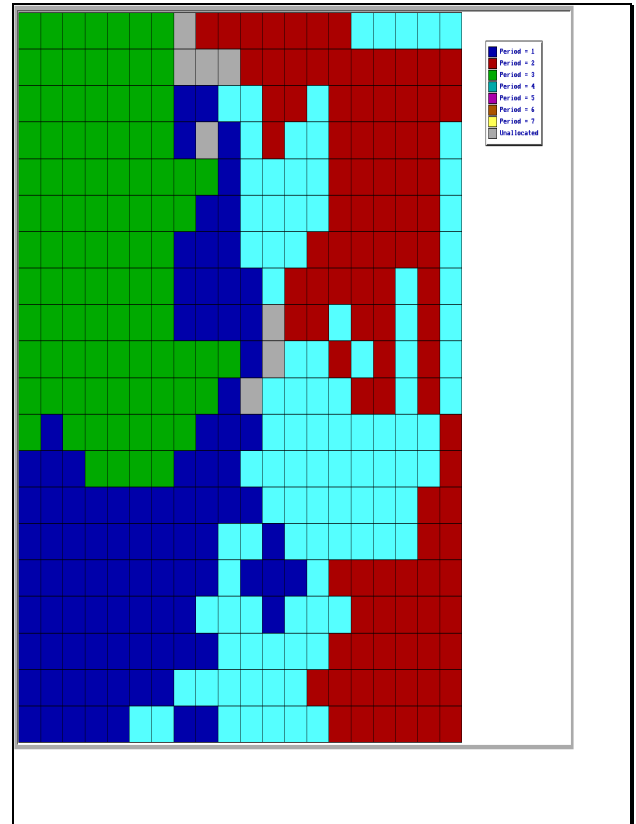


Figure 5. Block configuration 2 – corner point adjacencies included.

Summary

Although we at Remsoft are by no means experts on any GIS, we have been exposed to a number of vector-based geographic information systems and we have yet to see a system that can readily provide adjacency relationships for all stands across multiple map boundaries. Even those systems that provide adjacency information in a relatively accessible form, do not provide methods to differentiate common-point adjacencies from common-arc adjacencies without a significant programming effort. Being able to access data structures directly allowed us to generate adjacency relationships, but the situation

would be vastly improved if system vendors provided these functions directly. Unfortunately, these very basic functions are absolutely necessary to implement automated blocking tools, but the inability to access data structures or system functions to provide the required information hampers deployment of spatial forest management tools.

An automated procedure for developing harvest blocks and scheduling is not a trivial program to implement. Many factors that influence the size and shape of a feasible harvest block are patently obvious to a human observer, but these same factors can

confound an algorithm working under a complex set of guidelines. In our approach, we use a strict definition of adjacency that omits corner-point adjacencies to develop harvest blocks, but includes proximate stands which are obvious candidates for inclusion (e.g., stands which are on either side of a road). Then, when harvest blocks are scheduled under opening size and adjacency constraints, we expand the adjacency list to include not only the corner point adjacent stands but proximate stands that have nodes within a given distance away. Admittedly, our proximate stands list is not perfect, but it is an improvement over proxies like centroids.