# AN OPTIMAL APPROACH AND ALGORITHMS FOR LIMITING FOREST OPENING SIZE IN SPATIALLY CONSTRAINED TIMBER HARVEST SCHEDULING MODELS

David S. Mullen[1]

ABSTRACT.—It is assumed that the reason for including spatial constraints in harvest scheduling models is to place an upper limit on the size of forest openings that occur due to clear-cutting. An optimal approach to enforcing this limitation is to allow adjacent stands to be cut in any planning horizon period, as long as, at any point in time, the combined area of any group of adjacent stands that forms an opening, does not exceed the maximum allowed opening size. Two algorithms are presented that can be used in harvest scheduling models to implement this approach.

## INTRODUCTION

Increasingly, forest managers are being required to include spatial restrictions in their operational harvest schedules. These restrictions are imposed due to concern for maintaining and enhancing wildlife habitat, biodiversity, watershed integrity, and esthetic appeal (Hokans 1983; Rosie 1990; Lockwood 1993; Weintraub 1995).

This paper describes an optimal approach to including spatial constraints in harvest scheduling optimization models. In addition, it provides an informal proof that the approach is optimal and gives algorithms that can be used with heuristic and random search harvest scheduling models. The approach and algorithms were previously described and used in the author's master's thesis (Mullen 1996).

## BACKGROUND

Different approaches have been proposed and used to include spatial constraints in harvest schedule optimization models. The two general approaches that have been used most often are the adjacency constraint approach and the opening size restriction approach. These are discussed in the following sub-sections.

### Adjacency Constraint Approach (Non-Grouping Approach)

With the adjacency constraint approach, spatial restrictions are formulated so that adjacent stands are not allowed to be harvested within a specified green-up time period of each other. The green-up time period is referred to as the exclusion period, and represents the number of planning horizon periods needed for a harvested stand to be regenerated and grow to some arbitrary age or tree height. Because the this approach does not allow stands to be grouped together for harvest in the same period, we will also call this the "non-grouping approach". Following are a few examples of the many models that have been developed that use adjacency constraints.

Nelson et al. (1988) implemented the non-grouping approach in their Monte Carlo integer programming based model. Each time their model schedules a stand, it sets adjacent stands ineligible for harvest in the same period. This implementation only provides for a one-period green-up. Examples of other heuristic and random search

[1] David S. Mullen is Forest Resource Systems Specialist at Rayonier, Southeast Forest Resources, P. O. Box 728, Fernandina Beach, FL 32034. He can be reached by e-mail at david.mullen@rayonier.com.

based models that use adjacency constraints include the SCRAM model of O'Hara, et al. (1989); the ROHO-PATH model of Yoshimoto et al. (1994a), and the ECOPLAN system of Misund et al. (1995). Some of these models allow for multiperiod green-up exclusion. Much work has been done to develop efficient ways to represent adjacency constraints in mathematical programming based models. The work presented by Yoshimoto and Brodie (1994b) and by Torres-Rojo and Brodie (1990) are examples of just two of the many papers that have used mathematical programming models with adjacency constraints.

### Opening Size Restriction Approach (Grouping Approach)

It is often desirable to allow groups of adjacent stands to be harvested together if their combined area does not exceed a maximum allowed opening size. With this approach, the goal is not to prevent adjacent individual stands from being harvested together, but to prevent adjacent groups of stands from being harvested together in patterns that create forest openings above the maximum size limit.

Mathematical programming based models are not suitable for allowing stand grouping. This is because the number of possible combinations of allowable stand groupings becomes astronomical for large scale scheduling problems and the combinations must be explicitly specified in the problem constraints for each possible combination. For this reason, when we want to allow groups of stands to be harvested together, we are forced to abandon mathematical programming models and turn to heuristic and random search models.

Examples of a harvest scheduling models that use opening size restrictions include Dallain's BLOCK model (Dallain 1989), Baskent's GISFORMAN model (Baskent 1990), Lockwood and Moore's simulated annealing based model (Lockwood 1993), and Jamnick and Walters CRYSTAL model (Jamnick 1991). Each of these models uses some sort of grouping algorithm that discourages or prevents opening size restriction violations. Lockwood and Moore's model does not strictly enforce the opening size restrictions, but applies a penalty function to violations. The penalty is proportional to the extent of the violations. This makes the model favor solutions with fewer and less severe violations. In the other models, the grouping algorithms strictly enforce opening size restrictions. All of these models use grouping algorithms that group stands within the same period only. We will refer to this as the "within-period grouping approach."

## THE ACROSS-PERIOD GROUPING APPROACH

As part of the author's master's thesis, a more flexible grouping approach and implementation algorithms were developed and used in three spatially constrained harvest scheduling models (Mullen 1996). In the approach developed, groups of adjacent stands are allowed to include stands harvested in different periods within a green-up time period of each other. Because grouping is done across periods, we will refer to the approach as the "across-period grouping approach". The definition of the across-period grouping approach mirrors the basic goal of all spatial restriction approaches, and can be stated as follows:

> The Across-Period Grouping Approach: A harvest schedule does not violate spatial restrictions if, during each period of the schedule, the combined area of each group of adjacent stands that forms a contiguous opening does not exceed the maximum allowable opening size. Stands in an opening group do not necessarily have to be scheduled in the same period.

This approach does not prohibit adjacent stands from being harvested within a green-up time period of each other as is the case with the non-grouping approach. Nor does it force groups of adjacent stands harvested together as a group to all be harvested in the same planning horizon period as is the case with the within-period grouping approach. The <u>only</u> restriction imposed is that openings over a maximum allowable size never occur.

As an example to show how the across-period grouping approach is less restrictive than the other two approaches, consider a “forest” of three 40-acre stands that are all adjacent to each other. Let the maximum opening size be 100 acres, let the planning horizon consist of three 2-year periods, and let the green-up period be four years (two

periods). Let the management regimes be 1-harvest in period 1, 2-harvest in period 2, 3-harvest in period 3, and 4-do nothing (harvest beyond the end of the planning horizon). Let $\boldsymbol{S} = <S_1, S_2, S_3>$, $S_i \in \{1,2,3,4\}$ denote a schedule. Without any spatial constraints, there are $4^3 = 64$ possible schedules: <1,1,1>, <1,1,2>, . . . , <4,4,4>. If we applied non-grouping adjacency constraints approach, there would be only sixteen allowable schedules: <1,3,4>, <1,4,3>, <1,4,4>, <2,4,4>, <3,1,4>, <3,4,1>, <3,4,4>, <4,1,3>, <4,1,4>, <4,2,4>, <4,3,1>, <4,3,4>, <4,4,1>, <4,4,1>, <4,4,1>, <4,4,4>. Note that spatial constraints are not considered for regime 4. If we apply the within-period grouping approach, schedules with two stands cut in the same period and the other stand cut at least two periods away or by the do-nothing regime could be added to the set of schedules from the non-grouping approach. This would give an additional 14 schedules: <1,1,3>, <1,1,4>, <1,3,1>, <1,3,3>, <1,4,1>, <2,2,4>, <2,4,2>, <3,1,3>, <3,3,1>, <3,3,4>, <3,4,3>, <4,1,1>, <4,2,2>, <4,3,3>. If we apply the across-period grouping approach, all schedules except those having all three stands cut within two-periods are legal. This would add 19 schedules to the 30 that are legal for the within-period grouping approach: <1,2,3>, <1,2,4>, <1,3,2>, <1,4,2>, <2,1,3>, <2,1,4>, <2,3,1>, <2,3,4>, <2,4,1>, <2,4,3>, <3,1,1>, <3,1,2>, <3,2,1>, <3,2,4>, <3,4,2>, <4,1,2>, <4,2,1>, <4,2,3>, <4,3,2>. The non-grouping approach gives us 16 legal schedules. The within-period grouping approach gives us 30 legal schedules, and the across-period grouping approach gives us 49 legal schedules.

## PROOF OF OPTIMALITY OF THE ACROSS-PERIOD GROUPING APPROACH

Following is an informal proof that the across-period grouping approach is optimal compared to the non-grouping approach and the within-period grouping approach.

Let the planning horizon be composed of a consecutive sequence of $P$ $L$-year periods, numbered 1, 2, . . ., $P$. Let there be $M=P+1$ management regimes. Regimes 1 through $P$ each have only one harvest and it occurs in the period matching the regime number. Regime $M$ is the "do-nothing" regime which does not have a harvest in the planning horizon and is not subject to spatial restriction. Let $E$ be the exclusion period or (green-up period) specified as a number of planning horizon periods. For example, if the $L = 4$ and the green-up time is 10 years, then $E$ = CEILING(10/4) = 3. Let $O$ be the maximum allowed opening size. Let $ADJ_i$ be the adjacency list for stand $i$. Let $A_i$ be the land area of stand $i$. To prevent opening size violations, stands must be defined such that for every stand, $A_i \leq O$.

Let the symbol $\alpha$ designate the non-grouping approach. With this approach, no neighbor of any stand harvested in period j may be harvested during the interval of planning horizon periods [j-E+1, j+E-1]. For example, if stand 1 is scheduled in period 5, the exclusion period is 3 periods, and stand 2 is adjacent to stand 1, then stand 2 could not be harvested in the interval of periods from period 5-3+1 = 3 to period 5+3-1 = 7.

Let $\beta$ be the within-period grouping approach. This is the opening size restriction approach that allows stands to be grouped together for harvest within a given period. With this approach, stands may be grouped for harvest in the same period as long as total area of the group does not exceed the maximum allowed opening size. In addition, no neighbor of a stand or group of stands harvested in period j may be harvested during the interval of planning horizon periods [j-E+1, j+E-1]. For a given harvest scheduling problem, let $S^*\beta$ be the set of all schedules that meet the within-period grouping approach spatial restrictions.

Let $\chi$ designate the across-period grouping approach. With this approach, the only restriction imposed is that openings over a maximum allowable size never occur during the planning horizon.

For a given harvest scheduling problem, let $\boldsymbol{S}^*_\alpha$, $\boldsymbol{S}^*_\beta$, and $\boldsymbol{S}^*_\chi$ be the sets of all possible legal schedules that meet the $\alpha$, $\beta$, and $\chi$ spatial restrictions, respectively. To prove that $\chi$ is optimal, we need to show that $\boldsymbol{S}^*_\chi$ contains $\boldsymbol{S}^*_\alpha$ and $\boldsymbol{S}^*_\beta$. Said another way, we need to show that $\boldsymbol{S}^*_\alpha \subseteq \boldsymbol{S}^*_\chi$ and $\boldsymbol{S}^*_\beta \subseteq \boldsymbol{S}^*_\chi$. We can do this by first showing $\boldsymbol{S}^*_\alpha \subseteq \boldsymbol{S}^*_\beta$, and then showing $\boldsymbol{S}^*_\beta \subseteq \boldsymbol{S}^*_\chi$. If we can do that, it will follow that $\boldsymbol{S}^*_\alpha \subseteq \boldsymbol{S}^*_\chi$. The only difference between $\alpha$ and $\beta$ is that $\beta$ allows legal-sized groups of two or more adjacent stands to be cut in the same period, whereas $\alpha$ does not. So, $\boldsymbol{S}^*_\beta$ would contain all schedules in $\boldsymbol{S}^*_\alpha$ plus those schedules where legal sized groups of adjacent stands are scheduled in the same period. Therefore $\boldsymbol{S}^*_\alpha \subseteq \boldsymbol{S}^*_\beta$. The only difference between $\beta$ and $\chi$ is that $\chi$ allows legal sized groups of

stands to be scheduled in different periods within an exclusion period of each other whereas β requires that groups be harvested in the same period. So, $\boldsymbol{S}^*_\chi$ would contain all schedules in $\boldsymbol{S}^*_\beta$ plus those schedules where legal-sized groups of adjacent stands are scheduled in different periods within a green-up period. Therefore, $\boldsymbol{S}^*_\beta \subseteq \boldsymbol{S}^*_\chi$. Because $\boldsymbol{S}^*_\alpha \subseteq \boldsymbol{S}^*_\beta$ and $\boldsymbol{S}^*_\beta \subseteq \boldsymbol{S}^*_\chi$, it follows that $\boldsymbol{S}^*_\alpha \subseteq \boldsymbol{S}^*_\chi$. Since $\boldsymbol{S}^*_\chi$ contains $\boldsymbol{S}^*_\alpha$ and $\boldsymbol{S}^*_\beta$, then χ is optimal.

## ALGORITHMS

This section presents two algorithms that can be used to implement the across-period grouping approach. The algorithms are suitable for use in random search harvest scheduling models, but not to mathematical programming models.

Figure 1 gives the first algorithm. This algorithm, named *is_size_violation*, is intended for use in models that strictly enforce spatial restrictions as hard constraints. It would be called once for each stand as it is added to the schedule to test to see if adding the stand at the period being tested would cause an opening size violation. It returns TRUE if an opening size violation occurs or FALSE if no violation occurs. Figure 2 gives the second algorithm. This algorithm, named *sum_size_violation*, is for use in models that loosely enforce spatial restrictions as soft constraints using a penalty function for opening size violations. It would be called once after the scheduling model has finished building a schedule to determine the sum of the size violation penalty function values. Both algorithms use a common subprogram named *group_adj* that performs a graph traversal to find a group of adjacent stands scheduled within a window of periods that spans a green-up period. The *group_adj* subprogram is given in Figure 3. For each stand being tested, *group_adj* is called *E* times. Each call determines whether the stand being tested participates in an opening with adjacent stands within a window of *E* planning horizon periods. If so, the subprogram returns the set of stand numbers for the group. Figure 4 provides definitions of the variables, arrays, and sets used in the pseudocode listings of Figures 1, 2, and 3.

```
BOOLEAN FUNCTION is_size_violation(i, E, O, P, S, A, ADJ, R)
// Stand i is being added to the schedule. S_i has been set to a regime number in the range [1,P+1].
// All stands, i', that have not yet been added to the schedule have S_i' = 0. To test to see if adding stand i
// to the schedule as regime S_i will cause an opening size constraint violation, execute function
// is_size_violation(..). The return value is TRUE if a violation occurs or FALSE if no violation occurs.
DEFINE LOCAL VARIABLES: i', w, j_beg, j_end, sum, G
IF Si > P ∨ |ADJi| = 0 THEN                    // If S_i is the do-nothing
        RETURN FALSE                           // regime or stand has no
END IF                                         // neighbors, return FALSE.
FOR w = 1 TO E                                 // We have E windows to test.
        j_beg ← S_i - E + w                    // Define exclusion window.
        j_end ← j_beg + E - 1
        G ← {i}                                // Initialize group with stand i.
        group_adj(i, j_beg, j_end, P, G, ADJ, S, R)   // Add adjacent stands cut
                                               // within exclusion window.
        IF |G| > 1 THEN                        // Don't test if only 1 stand.
                sum ← Σ_{i'∈G} Ai'             // Sum area of stands in group.
                IF sum > O THEN                // Have violation if sum of
                        RETURN TRUE            // area for any window > O.
                END IF
        END IF
ENDFOR
RETURN FALSE                                   // Didn't find any violations.
END FUNCTION is_size_violation
```

Figure 1.—Algorithm to use while a schedule is being built to test each stand as it is added to the schedule to see if adding it will cause an opening size violation

```
NUMERIC FUNCTION sum_size_violation(N, E, O, P, Z_a, S, A, ADJ, R)
// Summarize the penalties for adjacency constraint violations for schedule S.
DEFINE LOCAL VARIABLES: local variables: T, G, G' , Z_adj, i, i', j_beg, j_end, size, size_max, w
T ← ∅ ; Z_adj ← 0
FOR i = 1 TO N
        // Stand i doesn't need to be tested if it is scheduled beyond the end of the
        // planning horizon, or if it has no neighbors, or if it is already in T.
        IF ( S_i > P ∨ |ADJ_i| = 0 ∨ i ∈ ⋃_{G∈T} G ) THEN NEXT i
        size_max ← 0
        FOR w = 1 TO E                                  // Have E windows to test.
                j_beg ← S_i - E + w                     // Define exclusion window.
                j_end ← j_beg + E - 1
                G ← {i}                                 // Initialize group with stand i.
                group_adj(i, j_beg, j_end, P, G, ADJ, S, R)   // Add adjacent stands cut
                                                        // within exclusion window.
                IF |G| > 1 THEN                         // Don't care if only 1 stand.
                        size ← Σ_{i'∈G} A_i'            // Sum area in group.
                        IF size > O ∧ size > size_max THEN
                                G' ← G          // Save opening group and size if the acre sum is
                                size_max ← size // greater than that for any previous window's
                        END IF                  // violation.
                END IF
        END FOR
        // If stand i is part of a multistand group larger than the maximum opening size, add a
        // penalty based on the total grouped area of the group from the time window with the highest
        // group area. Note that if stand i was not grouped, then size_max will still be 0, in which case
        // we don't need to do anything.
        IF size_max > O THEN
                Z_adj ← Z_adj + Z_a·(size_max - O)
                T ← T ∪ {G'}                    // Note: T is a set of sets.
        END IF
END FOR
RETURN Z_adj
END FUNCTION sum_size_violation
```

Figure 2.—Algorithm to use after a schedule has been built to summarize opening size constraint violations

```
PROGRAM group_adj(i, j_beg, j_end, P, G, ADJ, S, R)
// group_adj() constructs a set of stands adjacent to stand i that are harvested within the exclusion window
// [j_beg, j_end]. This is a recursive function. The first level call adds neighbors of the initial seed stand. The next
// level call adds neighbors of the stands added in the first level call. Etc. The recursion performs a depth first
// search of the adjacency graph. Upon initial call to group_adj, G contains only the seed stand. Parameter i must
// be passed by value. All others may be passed by address.
DEFINE LOCAL VARIABLE: i' // unit number of a stand adjacent to stand i.
FOREACH i' ∈ ADJ_i
        IF i' ∉ G THEN
                // The tests against S_i' are to group based on new schedule. The tests
                // against R_i' are to group based on recent harvesting activity.
            IF (S_i' > 0 ∧ S_i' ≤ P ∧ S_i' ≥ j_beg ∧ S_i' ≤ j_end) ∨ (R_i' ≥ j_beg ∧ R_i' ≤ j_end) THEN
                // Stand i' is part of group. Add it to G and recursively call group_adj.
                G ← G ∪ {i'}
                group_adj(i', j_beg, j_end, P, G, ADJ, S, R);      // Recurse.
            END IF
        END IF
END FOREACH
RETURN
END PROGRAM group_adj
```

Figure 3.—Algorithm for subprogram *group_adj* which is called by algorithms in Figures 1 and 2

| | | |
|---|---|---|
| $i$, $i'$ | = | stand numbers. |
| $w$ | = | exclusion window index. |
| $j_{beg}$ | = | beginning period in exclusion window. |
| $j_{end}$ | = | ending period in exclusion window. |
| $N$ | = | number of stands in the forest. |
| $E$ | = | exclusion period in number of planning horizon periods. |
| $O$ | = | maximum allowed opening size area. |
| $P$ | = | number of periods in the planning horizon. |
| $size$ | = | sum of acres in a opening group ($G$). |
| $size_{max}$ | = | maximum size found for opening groups seeded at one stand. |
| $Z_a$ | = | value penalty per unit of area for adjacency constraint violations. |
| $Z_{adj}$ | = | sum of value penalties for a group. |
| $S$ | = | array designating schedule. If $S_{i'} = 0$, stand has not been added to the schedule. If $S_{i'} \in [1,P]$, stand is scheduled in period $S_{i'}$ . If $S_{i'} = P+1$, stand is scheduled beyond end of planning horizon. |
| $A$ | = | array of stand areas -- $A_i$ is area of stand $i$. |
| $ADJ$ | = | array of adjacency sets -- $ADJ_i$ is sets of stand numbers of stands adjacent to stand $i$. For example, $ADJ_4 = \{7, 9, 10\}$ would indicate stand 4 is adjacent to stands 7, 9, and 10. |
| $R$ | = | array of recent cut designations. If $R_i < 1$, then it represents the implied recent cut period for stand $i$. If $R_i \geq (E + P + 1)$, then it means stand $i$ was not recently cut. |
| $G$ | = | variable for constructing an opening group. |
| $G'$ | = | variable used to hold largest opening group found. |
| $T$ | = | variable used for constructing the set $T$ of all non-overlapping opening groups while evaluating adjacency violations. |
| $sum$ | = | sum of area of stands in a group $G$. |

Figure 4.—Descriptions of variables, arrays, and sets used in algorithms shown in Fig. 1, 2, and 3

## CONCLUSION

I have presented an optimal approach for including spatial restrictions in harvest scheduling optimization models. Also presented were an informal proof that the approach is optimal and algorithms for implementing the approach. The algorithms are suitable for use in heuristic and random search harvest scheduling models that either strictly enforce spatial restrictions, or use a penalty function to discourage spatial restriction violations. Whereas the approach presented here is optimal, the algorithms are not necessarily optimal for implementing the approach. It is likely that some efficiencies could be added to the algorithms.

## LITERATURE CITED


BASKENT, EMIN ZEKI. 1990. *Spatial wood supply modelling: concept and practice*. M.Sc. thesis, University of New Brunswick, Fredericton, N.B.

DALLAIN, PATRICK. L. 1989. *A block harvest scheduling model for spatial management planning*. M.Sc. thesis, University of New Brunswick, Fredericton, N.B.

HOKANS, R.H. 1983. Evaluating Spatial Feasibility of Harvest Schedules with Simulated Stand Selection Decisions. *Journal of Forestry* 81: 601-603, 613.

JAMNICK, MARK S., and K. WALTERS. 1991. Harvest blocking, adjacency constraints, and timber harvest volumes. In: *Proceedings of the 1991 Systems Analysis in Forest Resources Symposium: 1991 March 3-7* (1991) pp. 255-266., Charleston, SC.

LOCKWOOD, CAREY, AND T. MOORE. 1993. Harvest scheduling with spatial constraints: a simulated annealing approach. *Canadian Journal of Forest Research* 23: 468-478.

MISUND, GUNNAR, B.S. JOHANSEN, AND G. HASLE. 1995. Integration of Geographical Information Technology and Constraint Reasoning - A promising Approach to Forest Management. *SINTEF Informatics*, P. O. Box 124, Blindern, N-0314 Oslo, Norway.

MULLEN, DAVID S. 1996. *A comparison of genetic algorithms and Monte Carlo integer programming for optimization of adjacency constrained timber harvest scheduling problems*. Master of science thesis. Jacksonville, Florida: University of North Florida, Department of Computer and Information Sciences.

NELSON, JOHN, J.D. BRODIE, AND J. SESSIONS. 1988. Integrating short-term spatially feasible harvest plans with long-term harvest schedules using Monte Carlo integer programming and linear programming. In: *The 1988 Symposium on Systems Analysis in Forest Resources*, eds. B. Kent and L. Davis, 224-229. Gen. Tech. Rep. RM-161. U.S. Department of Agriculture, Forest Service, Rocky Mountain Forest Range Experiment Station.

O'HARA, ANTHONY J., B.H. FAALAND, AND B.B. HARE. 1989. Spatially constrained timber harvest scheduling. *Canadian Journal of Forest Research* 19: 715-724.

ROISE, JOSEPH P. 1990. Multicriteria Nonlinear Programming for Optimal Spatial Allocation of Stands. *Forest Science* 36: 487-501.

TORRES-ROJO, JUAN M., AND J. DOUGLAS BRODIE. 1990. Adjacency constraints in harvest scheduling: an aggregation heuristic. *Canadian Journal of Forest Research* 20: 978-987.

WEINTRAUB, ANDRES, ET AL. 1995. Heuristic procedures for solving mixed-integer harvest scheduling—transportation planning models. *Canadian Journal of Forest Research* 25: 1618-1626.

YOSHIMOTO, ATSUSHI, J.D. BRODIE, AND J. SESSIONS. 1994a. A new heuristic to solve spatially constrained long-term harvest scheduling problems. *Forest Science* 40: 365-396.

YOSHIMOTO, ATSUSHI, J. DOUGLAS BRODIE, AND J. SESSIONS 1994b. Comparative analysis of algorithms to generate adjacency constraints. *Canadian Journal of Forest Research* 24: 1277-1288.