

Adjacency constraints in harvest scheduling: an aggregation heuristic

JUAN M. TORRES-ROJO¹ AND J. DOUGLAS BRODIE

Department of Forest Resources Management, Oregon State University, Corvallis, OR 97331-5703, U.S.A.

Received July 27, 1989

Accepted January 21, 1990

TORRES-ROJO, J. M., and BRODIE, J. D. 1990. Adjacency constraints in harvest scheduling: an aggregation heuristic. *Can. J. For. Res.* 20: 978-986.

An heuristic for adjacency constraint aggregation is proposed. The heuristic is composed of two procedures. Procedure 1 consists of identifying harvesting areas for which it is not necessary to write adjacency constraints. Procedure 2 consists of writing one adjacency constraint for each one of the harvesting areas not identified in procedure 1. Such adjacency constraints consider all the adjacency relations between the harvesting area and its surrounding areas. The heuristic is based on the concept of penalties and the four-color theorem. The aggregated constraints present fewer variables per constraint than the aggregator described by B.J. Meneghin, M.W. Kirby, and J.G. Jones (1988. *USDA For. Serv. Rocky Mt. For. Range Exp. Stn. Gen. Tech. Rep. RM-161*, pp. 46-53) and can easily be generated mechanically from the adjacency matrix. In addition, the proposed heuristic does not require the tedious task of identifying type 1 and 2 constraints as with Meneghin's algorithm. Hence the combinatorial work to compute the aggregated constraints is reduced significantly. Comparisons showed that the proposed heuristic requires about a third of the constraints required by the conventional adjacency constraint formulation and about the same number of constraints as the procedure suggested by B.J. Meneghin, M.W. Kirby, and J.G. Jones (1988).

TORRES-ROJO, J. M., et BRODIE, J. D. 1990. Adjacency constraints in harvest scheduling: an aggregation heuristic. *Can. J. For. Res.* 20 : 978-986.

Une méthode heuristique est proposée pour l'agrégation de contraintes contiguës. La méthode heuristique est composée de deux procédures. La première consiste à identifier des surfaces de coupe pour lesquelles il n'est pas nécessaire d'écrire des contraintes contiguës. La deuxième consiste à écrire une contrainte contiguë pour chacune des surfaces de coupe qui ne sont pas identifiées dans la première procédure. De telles contraintes contiguës considèrent toutes les relations contiguës entre les surfaces de coupe et les surfaces avoisinantes. La méthode heuristique est basée sur le concept des pénalités et le «four-color theorem.» Les contraintes agrégées présentent moins de variables par contrainte que l'agrégat décrit par B.J. Meneghin, M.W. Kirby et J.G. Jones (1988. *USDA For. Serv. Rocky Mt. For. Range Exp. Stn. Gen. Tech. Rep. RM-161*, p. 46-53) et peuvent être facilement générées mécaniquement à partir de la matrice adjacente. En plus, la méthode heuristique proposée ne requiert pas la tâche fastidieuse d'identifier les contraintes de type 1 et de type 2 comme dans l'algorithme de Meneghin. Ainsi, le travail combinatoire de calculer les contraintes contiguës est réduit de façon significative. Les comparaisons montrent que la méthode heuristique proposée requiert environ le tiers des contraintes requises par la formulation conventionnelle des contraintes contiguës et à peu près le même nombre de contraintes que la procédure suggérée par B.J. Meneghin, M.W. Kirby et J.G. Jones (1988).

[Traduit par la revue]

Introduction

Tactical planning requires consideration of spatial relationships among harvesting areas. Such spatial considerations often reflect legal or biological requirements for the management of forests, or in other instances, they simply are logistic requirements for the practical implementation of forest plans. Sometimes these spatial considerations are requirements for multiple objective forest planning, such as wildlife habitat management or necessary restrictions on sediment production for the protection of fisheries. These types of spatial requirements are often modeled through adjacency constraints. Such constraints restrict the selection of the harvesting areas so that adjacent harvesting areas sharing a common border (or a common corner) cannot be harvested in the same period.

Adjacency constraints add a large number of rows to any integer programming or mixed integer programming formulation on an area-based harvest scheduling problem. So these formulations are often limited by the number of constraints rather than the number of variables. Hence constraint aggregation of adjacency restrictions can help fit those large complex formulations into standard linear pro-

gramming or integer programming solution packages. More important potential applications of the aggregated constraints are improving the efficiency of heuristics to solve special integer programming problems such as the multi-constrained knapsack problem, or its use when the solution to the integer programming problem is obtained through relaxation, where a small number of constraints is desired.

Aggregation of equations with integer coefficients (such as adjacency constraints) has been studied intensively because of its simplicity and with the aim of deriving aggregators with manageable coefficients that are able to improve the computational efficiency of current solution algorithms (Fishburn and Kochenberg 1985). There exist some reasonable concerns about the improvement in computational efficiency by using aggregated constraints (Onyekwelu 1983). Some authors have even suggested that disaggregating constraints can lead to improved performance. Nevertheless, the solution of some highly structured integer programming problems has been improved by using aggregated constraints (Kannan 1983).

Techniques of constraint aggregation often have the associated problem of increasing the number of variables. Such is the case of the aggregators derived by Bradley (1971) or Padberg (1972), which require an additional variable for each inequality constraint that is aggregated. Other aggregators increase not only the number of variables but

¹On graduate leave from Instituto Nacional de Investigaciones Forestales, Agrícolas y Pecuarias, Mexico.

also the value of the coefficients in the aggregated constraint. Such is the case of Padberg's aggregator or the aggregator derived by Fishburn and Kochenber (1985). In some cases these values grow very rapidly with the number of aggregated constraints, which might represent a problem in practical applications. For instance, the Padberg's aggregator requires coefficients of the order of six digits to aggregate only 15 constraints.

Meneghin *et al.* (1988) considered the specific problem of aggregation of adjacency constraints. They developed a procedure to reduce the number of adjacency constraints in linear programming or integer programming formulation. Their approach combines simple adjacency restrictions (type 1 constraints) into multiple adjacency restrictions (type 2 constraints) to form aggregated adjacency constraints. The procedure requires less than half the constraints required by the conventional adjacency constraint formulation.

We present a procedure to reduce the number of adjacency constraints in a formulation of the harvest scheduling problem that can be implemented easily into a computer code. Our procedure is based on the concept of penalties and a principle used in map coloring and graph theory called the four-color theorem (Ringel 1959; Saaty and Kainen 1977). The procedure is simple and yields aggregated constraints with fewer variables per constraint than the ones reported by Meneghin *et al.* (1988). In addition, our approach also requires less than half the constraints required by the conventional adjacency constraint formulation, and in contrast with Meneghin's algorithm, it does not need the tedious two-step procedure of identifying type 1 and 2 constraints before constructing the aggregated constraint.

In the next section (section 2) we present our heuristic for adjacency constraints aggregation and the rules to compute the aggregators. In the third section some aggregators are derived for an example problem. Section four describes the basis of the aggregator. Finally the last section shows some comparisons among our aggregator, the conventional procedure, and the algorithm proposed by Meneghin *et al.* (1988).

An heuristic for constraint aggregation

Our heuristic is based on recognizing the following observations of the adjacency constraints: (i) If we were able to write an adjacency constraint per harvesting area (i.e., one constraint that describes all the adjacency relations of the reference harvesting area and all its adjacent areas), we would have at most the number of adjacency constraints equal to the number of harvesting areas. (ii) When we have an adjacency constraint per harvesting area, some constraints are redundant. Consider that the *i*th area (*a_i*) is surrounded by areas already described with adjacency constraints. Then area *a_i* does not need to be described by an additional adjacency constraint, since its adjacency relationships with its adjacent areas have already been described with the corresponding surrounding areas.

The first observation suggests that if we are able to write one adjacency constraint for each one of the *N* harvesting areas, the number of adjacency constraints is *N*. The second observation indicates that the number *N* of aggregated constraints can be reduced further.

Based on these observations, our heuristic consists of two steps. The first step, procedure 1, consists of identifying

areas for which it is not necessary to write adjacency constraints, given that their surrounding areas describe their adjacency relationships. Once we identify these areas, the second step, procedure 2, consists of applying a set of rules to write one adjacency constraint for each one of the areas not identified in procedure 1.

Procedure 1 is simple and will be illustrated through an example. Consider the pattern of harvesting areas depicted in Fig. 1. Following this pattern we can form the adjacency matrix (i.e., the matrix that shows us the adjacency relationships among all harvesting areas) shown in Table 1. Recalling that each *X* in Table 1 represents an adjacency relation, let us call *NR_i* the number of *X*'s in row *i*, and *NC_i* the number of *X*'s in column *i*. Thus for row 1, Fig. 2, *NR₁* = 3, and for column 1, *NC₁* = 3. If for the *i*th row *NR_i* = *NC_i*, then area *i* can be identified. Such identification indicates that the adjacency relations of area *i* can also be described by other areas. Hence it is redundant to write an adjacency constraint for that area. If area *i* can be identified in this way, then row *i* is deleted.

Procedure 1 consists of performing this identification of areas. Every time an area is identified its corresponding row in the adjacency matrix is deleted, and the procedure continues until we check all the areas. For instance, for the adjacency matrix depicted in Fig. 2, *NR₁* = *NC₁* = 3, so row 1 might be eliminated and rows 2-4 will still keep the adjacency relations of area 1 and areas 2-4. If row 1 is eliminated (i.e., area 1 has been identified), *NR₂* = 4 and *NC₂* = 3, so row 2 cannot be eliminated. And since *NR₃* = *NR₄* = 4 and *NC₃* = *NC₄* = 3, neither row 3 nor row 4 can be eliminated. However, *NR₅* = *NC₅* = 5. Therefore, row 5 can be eliminated. Following the same pattern, row 6 cannot be eliminated but row 7 can. Thus, rows 8 and 9 cannot be eliminated.

Note that to obtain the relationships *NC₂* = *NC₃* = *NC₄* = 3, we considered row 5, but this was before deleting it. The basic idea of this procedure is to identify areas that can be described by surrounding areas that have not been identified.

At the end of procedure 1 and considering the adjacency matrix depicted in Table 1, we identified areas 1, 5, and 7 as areas for which we do not need to write adjacency constraints.

Procedure 2 consists of writing one adjacency constraint for each one of the harvesting areas not identified in procedure 1. Each one of these constraints should include all the adjacency relations between the harvesting area and its surrounding areas. To write such a constraint can be an easy task if for the *i*th area all its adjacent areas are also adjacent to each other. For instance, consider Fig. 1. An adjacency constraint for area 9 that describes all the adjacency relationships between area 9 and its surrounding area is

$$[1] \quad X_7 + X_8 + X_9 \leq 1, \quad X_i \in \{0, 1\}$$

where

$$X_i = \begin{cases} 1 & \text{if area } i \text{ is harvested} \\ 0 & \text{otherwise} \end{cases}$$

which is called a triplet. Figure 2 shows three spatial patterns for which writing one adjacency constraint per harvesting area is simple. These patterns were defined by Meneghin *et al.* (1988) as type 1 inequalities. Consider Fig. 2, pattern *a*. To write an adjacency constraint for area 1, we have a pair:

$$X_1 + X_2 \leq 1$$

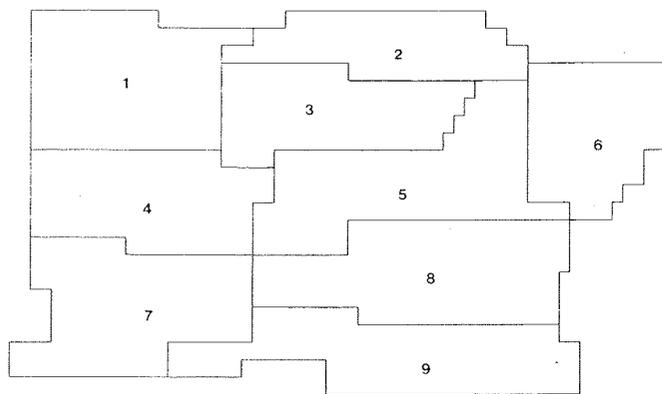


FIG. 1. Example forest with nine harvesting areas.

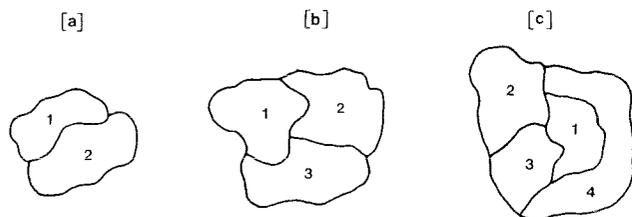


FIG. 2. Three different patterns where it is possible to write type 1 constraints.

To write a constraint for the same area 1 under pattern *b* we have a triplet:

$$X_1 + X_2 + X_3 \leq 1$$

and for writing a constraint for area 1 under pattern *c* we have a quadruplet:

$$X_1 + X_2 + X_3 + X_4 \leq 1$$

It turns out that all the type 1 inequalities described by Meneghin *et al.* (1988) represent an adjacency constraint for a given area with all its adjacency relationships considered if, and only if, all the areas in the constraint are the total number of areas adjacent to the area for which we are writing the constraint. However in most of the cases, not all the areas adjacent to a given harvesting area are adjacent to each other. Moreover, it is very unlikely that a type 1 inequality considers all the adjacent areas to a reference area unless it is a quadruplet, such as area 1 in pattern *c* (Fig. 2). For cases where type 1 inequalities do not work, our procedure consists of weighting the coefficients of the areas adjacent to the area for which we are writing the constraint. Consider again the pattern depicted in Fig. 1. If we want to write an adjacency constraint for area 1, the constraint

$$[2] \quad X_1 + X_2 + X_3 + X_4 \leq 1$$

is not valid given that areas 2 and 4 are not adjacent to each other, and this constraint [2] does not allow the feasible combination of harvesting areas 2 and 4 in the same period. However, the constraint

$$[3] \quad 5X_1 + 3X_2 + 4X_3 + 2X_4 \leq 5$$

keeps the desired adjacency relations and considers all the areas adjacent to area 1. Constraint [3] was partially constructed following the idea of weighting the coefficients of each of the harvesting areas that violate other constraints. These weights will be called penalties because of the way our procedure estimates them. The weighting (penalization) procedure is simple and better understood through an example. Hence, next we will define some notation and the

TABLE 1. Adjacency matrix for pattern in Fig. 1

Reference area	Adjacent areas								
	1	2	3	4	5	6	7	8	9
1		X	X	X					
2	X		X		X	X			
3	X	X		X	X				
4	X		X		X		X		
5		X	X	X		X			X
6		X			X				
7				X				X	X
8					X		X		X
9							X	X	

rules of the procedure, and then we will use those rules in an example problem.

We define reference area as the area for which we are writing the adjacency constraint. If only one additional adjacent area to the reference area is considered in the aggregated constraint, we say we have a first-level aggregation. If two adjacent areas are considered in the aggregated constraint, we have a second-level aggregation, and so on. So, the number of adjacent areas (NA) to the reference area equals the level of aggregation. If in any level of aggregation an area is adjacent only to the reference area and it is not adjacent to any of the areas included in the aggregated constraint, such an area is called colorable, otherwise it is called noncolorable.

Procedure 2 is sequential, and it computes the final weights of the aggregated constraints. The strategy follows a penalization procedure whose rules are as follows:

- (1) The right-hand side value (RHS) for any aggregated constraint equals the coefficient of the reference area in the constraint.
- (2) Each penalty has a value of 1, i.e., the value of the penalized coefficient increases by 1.
- (3) Each variable (harvesting area) that enters into the aggregated constraint adds a penalty to the reference area if the following is true: (a) It is the last variable to be added into the constraint. As a general rule, at most four noncolorable areas and all colorable areas will be included in each constraint. Note that the number of noncolorable areas can be less than four. (b) It is not the last variable, and before it enters, $NA \geq RHS$ (i.e., the number of adjacent areas to the reference area is greater than or equal to the current value of the RHS). Recall that the RHS is incremented as the coefficient of the reference area is incremented.
- (4) If the entering variable has an adjacency relation with any of the areas already considered in the aggregated constraint (i.e., it is a noncolorable area), a penalty is added to the entering variable for each one of its adjacent areas (variables) already in the aggregated constraint, but only one penalty is added to the reference area. In addition, one penalty is added to each of the areas adjacent to the area just entered. For instance, according to this rule, if the entering variable is adjacent to two areas already in the constraint, the entering variable is penalized twice, its adjacent areas (2) are penalized once, and the reference area is penalized once.
- (5) If the areas adjacent to the area just entered have an adjacency relation with any other area already in the

constraint, we have a transitivity effect. Such an area is penalized again, and a penalty is added to its adjacent area(s). This procedure is repeated for all adjacent areas. Each time a transitivity effect occurs a penalty is added to the reference area. For instance, if an entering variable, say area *A*, has two adjacent areas already in the constraint, say areas *B* and *C*, and in addition area *C* is adjacent to area *D*, which is also already in the constraint, then we have the transitivity effect:

$$A \rightarrow C \rightarrow D$$

Then area *C* is penalized again, and its adjacent area *D* and the reference area are also penalized.

- (6) If the constraint already has five areas (variables) including the reference area and there exists an area(s) adjacent only to the reference area (colorable area), such an area(s) enters the constraint without penalty, and the reference area is penalized just once.
- (7) The set of areas (variables) has to be unique to each constraint. Otherwise the constraint is considered redundant and is dropped. Likewise, the set of areas in a given constraint cannot be a subset in any other constraint.

Example

Consider that we want to write aggregated adjacency constraints for the pattern depicted in Fig. 1. The first step is to find the areas whose adjacency relations can be expressed by surrounding areas. Hence, we apply procedure 1 to identify such areas. From the last section we know that we do not need to write adjacency constraints for areas 1, 5, and 7. We then apply the rules of procedure 2 to write aggregated adjacency constraints for areas 2, 3, 4, 6, 8, and 9. Assume we want to compute an aggregated constraint for area 9 in Fig. 1. This area is adjacent to areas 7 and 8, but the two adjacent areas are adjacent to each other. Starting with the first level of aggregation we have

$$[4] \quad X_7 + X_9 \leq 1$$

For the second level of aggregation we have to penalize both areas, and the penalty is as follows. When X_8 enters, it is immediately penalized (rule 4) because it has an adjacent area in constraint [4], namely area 7 (X_7). Likewise, following rule 4, X_7 is penalized because of its adjacency relation with area 8. Following the same rule, a penalty is added to the reference area X_9 . In addition, following rule 3a, X_9 is penalized again because X_8 is the last variable to enter the aggregated constraint. Thus, X_9 is penalized twice; the first time because X_8 enters (rule 3a) and the second time because of the adjacency relations between areas 7 and 8. Figure 3 shows this penalization procedure, which yields the following aggregated constraint:

$$[5] \quad 2X_7 + 2X_8 + 3X_9 \leq 3$$

Observe that in this case the constraint

$$X_7 + X_8 + X_9 \leq 1$$

has the same effect and has smaller coefficients than ours. It turns out that if all the adjacent areas to a reference area form a type 1 constraint, i.e., pairs, triplets, and quadruplets (Meneghin *et al.* 1988), these constraints always have smaller coefficients than our aggregators. In these cases such aggregators might be used instead of the ones yielded through this heuristic.

Suppose that we want to write an adjacency constraint for area 1 (from procedure 1 we know that we do not need

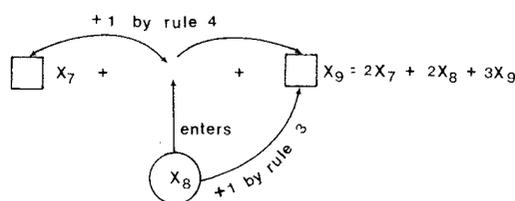


FIG. 3. Transitivity effect to compute the penalties.

to write an adjacency constraint for this area; however, we will use it to illustrate the penalization procedure). We start with the first aggregation level:

$$X_1 + X_2 \leq 1$$

By adding X_3 and the same steps we followed earlier, the second aggregation level yields

$$[6] \quad 3X_1 + 2X_2 + 2X_3 \leq 3$$

Note that in this case the second penalty to the reference area X_1 followed rule 3b not rule 3a, as did the last example. The third aggregation level has a transitivity effect that is described in two steps. First, following rule 4 we enter X_4 with a penalty because of its adjacency relation with X_3 . Under the same rule, X_3 and the reference area are also penalized. In addition, following rule 3a the reference area is penalized again because X_4 is the last entering variable. So, at the end of the first step we have

$$5X_1 + 2X_2 + 3X_3 + 2X_4 \leq 5$$

For the second step we observe that area 3 is adjacent to area 2 (the transitivity effect), and area 2 is already in the constraint. Hence, we penalize area 3 again and adjacent area 2, adding the corresponding penalty to the reference area (rule 5). Thus the final aggregator for area 1 is

$$[7] \quad 6X_1 + 3X_2 + 4X_3 + 2X_4 \leq 6$$

Notice that this aggregator violates the infeasible combination of harvesting areas 3 and 4 in the same period. However, this deficiency is corrected by other constraints. To prove it let us compute the aggregator for area 4. Following the procedure for area 1, the second level of aggregation for the aggregator of area 4 yields

$$[8] \quad 2X_1 + 2X_3 + 3X_4 \leq 3$$

For the third aggregation level we have the transitive effect, which is described in two steps. In the first step X_5 enters into constraint [8]. It is penalized immediately because it has an adjacent area (X_3). Hence X_3 and the reference area are penalized (rule 4), yielding

$$[9] \quad 2X_1 + 3X_3 + 4X_4 + 2X_5 \leq 4$$

Note that at this step, X_4 (reference area) is penalized just once (rule 4) since X_5 is not the last area adjacent to enter into the constraint. In addition, before X_5 enters, $RHS > NA$. Hence, no one of the requirements of rule 3 is met to penalize the reference area again. In the second step, we observe that there is an adjacency relation between X_1 and X_3 (the transitive effect). Hence X_3 has to be penalized again, and as a consequence the reference area and X_1 have to be penalized too, yielding the final aggregator for the third level of aggregation:

$$[10] \quad 3X_1 + 4X_3 + 5X_4 + 2X_5 \leq 5$$

Finally we enter area 7 (X_7) into constraint [10]. It is the last entering variable (area), so the reference area is penalized

TABLE 2. Aggregated constraints for each area in Fig. 1, considering four noncolorable areas per constraint

Reference area	Aggregated constraint
1	$6X_1 + 3X_2 + 4X_3 + 2X_4 \leq 6$
2*	$4X_1 + 9X_2 + 6X_3 + 4X_5 + 2X_6 \leq 9$
3*	$6X_1 + 5X_2 + 9X_3 + 4X_4 + 3X_5 \leq 9$
4*	$3X_1 + 4X_3 + 6X_4 + 2X_5 + X_7 \leq 6$
5	$5X_2 + 6X_3 + 3X_4 + 10X_5 + 2X_6 + X_8 \leq 10$
6*	$2X_2 + 2X_5 + 3X_6 \leq 3$
7	$X_4 + 4X_7 + 2X_8 + 2X_9 \leq 4$
8*	$X_5 + 2X_7 + 4X_8 + 2X_9 \leq 4$
9*	$2X_7 + 2X_8 + 3X_9 \leq 3$

*Constraints were needed to explain all adjacency relations in the pattern shown in Fig. 1.

once; and because X_7 has no adjacency relation with other areas in constraint [10] but the reference area (i.e., it is a colorable area), it enters without penalty, yielding the final aggregator:

$$[11] \quad 3X_1 + 4X_3 + 6X_4 + 2X_5 + X_7 \leq 6$$

Note that this aggregated constraint violates the requirement that X_3 and X_5 cannot be harvested in the same period. However, it avoids the violation incurred in the aggregated constraint [7], where X_3 and X_4 were allowed to be harvested in the same period.

The aggregator for area 8 can be obtained as follows. Applying the same steps as in the last aggregator, the second aggregation level yields:

$$2X_7 + 3X_8 + 2X_9 \leq 3$$

For the third aggregation level we enter X_5 without a penalty (it is colorable), and given that it is the last entering variable, the reference area is also penalized (rule 3a), yielding the final aggregator for area 8:

$$[12] \quad X_5 + 2X_7 + 4X_8 + 2X_9 \leq 4$$

Finally we describe how to compute the aggregator for area 3. For this area the second level of aggregation yields

$$[13] \quad 2X_1 + 2X_2 + 3X_3 \leq 3$$

For the third aggregation level we enter X_4 . Variable X_4 enters into constraint [13] with a penalty because of its adjacency relation with X_1 . Hence, following rule 4, X_1 and the reference area are penalized. However, X_1 is adjacent to X_2 (which is already in constraint [13]). Hence by the transitivity effect, X_1 has to be penalized again, and by rule 5, X_2 and the reference area are also penalized, yielding

$$[14] \quad 4X_1 + 3X_2 + 5X_3 + 2X_4 \leq 5$$

The fourth level of aggregation has a transitivity effect that is described in two steps. In the first step, X_5 enters with a double penalty because of its adjacency relations with areas 2 and 4, which are already in constraint [14]. At this step the reference area is penalized twice, one penalty because X_5 enters and it is the last entering variable (rule 3a) and the second penalty (rule 4) because X_5 has adjacent areas in constraint [14]. Likewise, following rule 4, the areas adjacent to X_5 , namely 2 and 4, are also penalized. Thus at this step, the aggregated constraint is

$$[15] \quad 4X_1 + 4X_2 + 7X_3 + 3X_4 + 3X_5 \leq 7$$

Since X_4 was penalized and it is adjacent to X_1 , there is a transitivity effect. Thus, following rule 5, X_1 , X_4 , and the reference area are penalized. Likewise, since X_2 was penalized and it is adjacent to X_1 , there is another transitivity effect, and by rule 5, X_2 , X_1 , and the reference area are penalized again, yielding the final aggregator:

$$[16] \quad 6X_1 + 5X_2 + 9X_3 + 4X_4 + 3X_5 \leq 9$$

As the reader can verify, the violation incurred in constraint [11], which permits the harvest of areas 3 and 5 in the same period, has been corrected by constraint [16]. Although constraint [16] permits the harvest of areas 2 and 5, the reader can verify that the constraint for area 2 corrects this infeasible combination. Table 2 shows the aggregated constraints for each one of the areas in Fig. 1. The constraints marked with an asterisk correspond to required constraints, and the ones without an asterisk correspond to the redundant constraints identified in procedure 1.

Thus, for describing the adjacency relations of the pattern depicted in Fig. 1, we need just the six constraints marked with an asterisk in Table 2, instead of the 15 constraints required with the conventional procedure. Something important to notice in procedure 2 is that the order in which the adjacent areas are picked to enter the constraint, or which adjacent areas are picked (if there are more than four adjacent areas to the reference area), does not affect the final adjacency relations obtained with the procedure. However, for some situations, constructed patterns following our rules could lead to a set of constraints that violates some of the adjacency relations. To avoid this possibility, we include as another requirement for the computation of aggregated constraints maintenance of the order in the selection of the variables each time they enter the aggregated constraints; i.e., if six possible variables can enter the constraint, first enter area 1, then area 2, and so on.

Basis of the heuristic to write aggregated adjacency constraints

Procedure 1 is a systematic mechanism that identifies harvesting areas whose adjacency relationships can be defined by their surrounding areas. Following the pattern in Fig. 1, the reader can verify that areas 1, 5, and 7 are completely surrounded by areas for which we have to write adjacency constraints. For instance, if the aggregated constraints for areas 2-4 consider area 1, then the constraint for area 1 is not necessary since its adjacency relations with its surrounding areas are implicit in the constraints for areas 2-4. In other words, the number of areas adjacent to area 1 equals the number of areas adjacent to area 1 left (unidentified) to describe the adjacency relationships of that area.

Procedure 1 can be interpreted in the following way. For any adjacency matrix, let the number of X 's in row i (NR_i) represent the areas adjacent to the reference area i , and let the number of X 's in column i (NC_i) represent the number of areas adjacent to area i left to describe the adjacency relations between area i and its surrounding areas. Then we can avoid writing the constraint for area i in $NR_i = NC_i$; i.e., the number of areas adjacent to area i equals the number of areas left to describe the adjacency relations of area i . As can be observed, procedure 1 is just a mechanized way to identify areas surrounded by areas not identified.

TABLE 3. Aggregated constraints for each area in Fig. 4, considering one noncolorable area per constraint

Reference area	Aggregated constraint
1	$2X_1 + X_2 + X_4 \leq 2$
2	$X_1 + 3X_2 + X_3 + X_5 \leq 3$
3	$X_2 + 2X_3 + X_6 \leq 2$
4	$X_1 + 3X_4 + X_5 + X_7 \leq 3$
5	$X_2 + X_4 + 4X_5 + X_6 + X_8 \leq 4$
6	$X_3 + X_5 + 3X_6 + X_9 \leq 3$
7	$X_4 + 2X_7 + X_8 \leq 2$
8	$X_5 + X_7 + 3X_8 + X_9 \leq 3$
9	$X_6 + X_8 + 2X_9 \leq 2$

In section 2 we described the rules of procedure 2 to compute aggregated adjacency constraints when the reference area has up to four adjacent areas (recall that colorable areas are not considered in the counting). However, it is very likely that the reference area has more than four noncolorable adjacent areas. We can apply the same rules when the reference area has more than four (noncolorable) adjacent areas. The problem with this approach is that the procedure becomes more complex if those additional areas are adjacent to areas already considered in the aggregated constraint. In this case the coefficients start to increase rapidly, basically because of the transitivity effect of the areas already in the constraint.

If we consider that all the areas adjacent to a given reference area might not be adjacent to each other, then some areas in the aggregated constraint do not appear in the aggregated constraint of other areas. For instance, following the pattern in Fig. 1, the aggregated constraint for area 1 includes area 4 (inequality 7). However, any adjacency constraint for area 2 (which is included in inequality 7) does not include area 4; i.e., we can harvest areas 2 and 4 in the same period, or seen as a chromatic scheduling problem, we say that we can color areas 2 and 4 with the same color in the map represented by Fig. 1. Thus the problem of deciding how many areas we should include in the aggregated adjacency constraint is related to a chromatic scheduling problem. Chromatic scheduling is the area of graph theory related to the scheduling of discrete events that span the same period of time, where some events can occur simultaneously but others cannot (Wood 1969). Gross and Dykstra (1988) used the principles of chromatic scheduling to investigate the minimum number of evenly spaced harvest entries needed to bring a forest into area regulation. We will use these principles to determine the minimum number of (noncolorable) areas adjacent to a reference area that we should include in writing an adjacency constraint for a reference area.

In graph theory the map coloring problem consists of finding the minimum number of colors sufficient for coloring a planar map such that two contiguous areas do not have the same color. This number of colors is called the chromatic number. The chromatic number can be a minimum of two for special map patterns or one if the map consists only of one area. The chromatic number is related to our problem, since if we know the chromatic number of the map that represents the areas of our harvest scheduling problem, such a number reduced by one, should be the minimum number

of adjacent areas (noncolorables) that we should include (whenever it is possible) in each one of the adjacency constraints. It is important to note that this minimum number of adjacent areas considers only the noncolorable areas. Colorable areas are always included in writing the aggregated adjacency constraint for a given reference area.

To clarify the statement that the chromatic number of the pattern reduced by one determines the minimum number of noncolorable areas we should include in the aggregated constraint, we will use two examples. First, consider a pattern of harvesting areas similar to a checkerboard (Fig. 1). This pattern has a chromatic number of 2,² which means that only two colors are sufficient to color each area and no two adjacent areas are the same color. Applied to our problem, we should be able to write adjacency constraints for each area such that in every aggregated constraint all the areas adjacent to the reference area are not adjacent to each other, i.e., all of them are colorable areas.

Following the rules of procedure 2, we can construct the aggregated adjacency constraints for each of the areas in this example. These aggregated adjacency constraints are shown in Table 3. In all cases we did not stop adding areas to the aggregated constraint once two areas were already in the aggregated constraint (recall that the chromatic number of this example is 2) since the additional areas are adjacent only to the reference area (colorable) and not to the other areas already in the aggregated constraint (recall rule 6). In the hypothetical case that one candidate area to enter the aggregated constraint was adjacent to the reference area and any area already in the constraint (considering a map with chromatic number of 2), we could ignore that candidate area because if we include it we would have two noncolorable areas in the constraint. Such a number of noncolorable areas is more than the minimum required for a pattern with a chromatic number of 2.

A second example might help clarify this idea. For our example problem in Fig. 1, the reader can easily verify that the chromatic number of the pattern is 3 (i.e., only three colors are needed to color the map and no two adjacent areas will have the same color). Thus, given the chromatic number of 3, we should be able to construct adjacency constraints for this example such that once an entering area is adjacent not only to the reference area but also to any other area already in the constraint (i.e., a noncolorable area), we stop entering noncolorable areas and just enter areas adjacent exclusively to the reference area (colorables). When we enter the first noncolorable area in the constraint, we automatically have two noncolorable areas in the aggregated constraint; and for the pattern with a chromatic number of 3, this is the minimum required.

Following the rules of procedure 2, we computed the adjacency constraints for each area in Fig. 1 and stopped the addition of noncolorable areas once the first noncolorable area is added to the constraint. Table 4 shows these constraints. Observe that there are no constraints for areas 6 and 9 since its adjacency relations are described by other constraints and the inclusion of these constraints would imply a violation of rule 7 (be redundant). Likewise, observe that for area 2 we could write a constraint similar to that

²Only the infinite checkerboard has a chromatic number of 2 since technically the area around the checkerboard has to have another color.

of area 1. However, this constraint would violate rule 7. Also observe that for area 1 we did not include the adjacent area 4, for area 2 we did not include the adjacent areas 1 and 6, and for area 3 we did not include the adjacent areas 2 and 5. In this way many other adjacent areas were not included in other constraints. The reason to avoid these areas in the pattern with a chromatic number of 3 is because we need just two noncolorable areas in each aggregated constraint, regardless if the reference area has more than two noncolorable adjacent areas. In the constraint for area 7 we included a fourth area, namely area 4, given that this area is adjacent only to reference area 7 but not to areas 8 and 9 (rule 6). In other words, area 4 can have the same color as areas 8 or 9. The same idea is used to include area 5 in the constraint for area 8.

Gross and Dykstra (1988) showed the exaggerated computing time required to find chromatic numbers under different constraints. In general, the chromatic number is difficult to find for many patterns, and it would not be practical to compute it every time we want to write aggregated constraints for a specific problem. Hence we will rely on the principles of graph theory to generalize our heuristic without having to compute the chromatic number of each pattern. One principle of graph theory called the four-color theorem states that "four colors are sufficient to color any map drawn in a plane or a sphere so that no two regions with a common boundary line are colored with the same color" (Ringel 1959). Based on this theorem we can state that at least three areas adjacent (recall that only noncolorable areas are considered in the counting) to the reference area would be necessary to identify the adjacency relations of each reference area.

Observe what could happen if we use the chromatic number to define the minimum number of noncolorable areas in an aggregated constraint. In the case of a chromatic number of 2 (Fig. 4), if we do not include any of the constraints (see Table 3), all the adjacency relations are kept. Thus, using procedure 1 we can eliminate constraints 1, 3, 5, 7, and 9, and the adjacency relations are kept. However, consider Table 4. In this case the aggregated adjacency constraints were computed considering just two noncolorable areas in each constraint, i.e., the number of colors (coloring number) corresponded to the chromatic number. If we do not include constraint 1 we violate the restriction that areas 1 and 2 are adjacent to each other. Hence we cannot eliminate arbitrarily any constraint in this case. Consider again the pattern in Fig. 1. If instead of using the chromatic number 3 we assume the pattern is five colorable, i.e., we will stop entering variables in the constraint once there are four noncolorable areas in the constraint, then we end up with the constraints shown in Table 4. Note that the rules of procedure 2 assume the pattern is five colorable. In this case, as the reader can verify in Table 2, we can eliminate arbitrarily any constraint, and the adjacency relations are kept.

By increasing the number of noncolorable areas in the aggregated constraint beyond the chromatic number, we have a better identification of the adjacency relations, which makes it feasible to apply procedure 1 and eliminate some constraints without incurring any violations of adjacency relationships. Therefore, assuming the validity of the four-color theorem, we would have a better identification of the adjacency relations for each reference area if we assume the patterns are five colorable. In this way, when we apply procedure 1 in complicated patterns that require four colors,

we reduce the possibility of deleting constraints needed to represent the required adjacency restrictions. Notice that if we do not apply procedure 1, three noncolorable areas per constraint are sufficient in each aggregated constraint (when the reference area has more than three noncolorable areas). However, if we apply procedure 1, there is a possibility of violating some adjacency relationships. Hence, we recommend the use of four noncolorable areas when possible to write the aggregated adjacency constraints.

To identify any difference in the number of aggregated constraints computed using different numbers of noncolorable areas considered in each aggregated adjacency constraint, we simulated 10 forests varying from 15 to 60 units. The examples were completely different, and we tried to simulate arrangements of up to 12 adjacent areas to only 1 area. We used 4, 5, and 6 as coloring numbers, i.e., 3, 4, and 5 as number of noncolorable areas included in each aggregated constraint. In all the examples procedure 1 was performed to eliminate redundant constraints. Hence, the initial number of constraints was the same for each example regardless of the coloring number. In all cases the derived adjacency constraints were able to identify the optimal solution without violating any adjacency relation. So it was not necessary to increase the number of constraints when small coloring numbers were used.

From these results and acting conservatively, we recommend using a coloring number of 5 to compute the aggregated constraints, since our objective is to reduce the number of constraints to a number less than the number of harvesting areas. By using a coloring number of 4 there is a possibility of violating adjacency relations for complicated patterns. By using larger coloring numbers we reduce that possibility, although we increase the number of variables per constraint and the size of the coefficients.

Note that the assumption that the pattern of harvest units is five colorable works correctly only if the exclusion period (the minimum age difference, in years, between adjacent stands) is equal to the return period (the number of years between any two cutting operations on a particular map). Nevertheless, these two numbers might differ. In such cases, the aggregated adjacency constraints obtained through this heuristic might violate some adjacency relations since the number to use in place of the chromatic number is more complicated than simply using 5. Gross (1989) has derived the number to use in place of 5 for various ratios of exclusion period to return period. These numbers could be used to set the minimum number of noncolorable areas to include in the constraint.

Another questionable statement about the rules of procedure 2 is the existence of a rationale for the penalty system. To give an idea of this rationale, assume a system of equations of the form

$$\sum_{j=1}^m X_{ij} = 1$$

$$i = 1, 2, \dots, n; m, n \geq 2; X_{ij} \in \{0, 1\}$$

where n is the number of equalities and m is the number of variables. The common aggregator for this system of equations is given by

$$[17] \quad \sum_{i=1}^n \sum_{j=1}^m a_i X_{ij} = \sum_{i=1}^n a_i$$

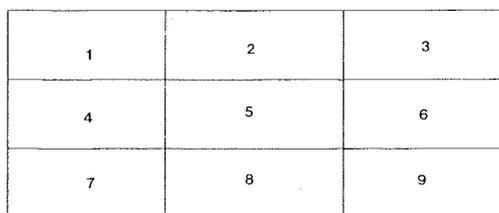


FIG. 4. Checkerboard pattern of an example forest.

TABLE 4. Aggregated constraints for each area in Fig. 1, considering two noncolorable areas per constraint

Reference area	Aggregated constraint
1	$3X_1 + 2X_2 + 2X_3 \leq 3$
2	$3X_2 + 2X_3 + 2X_5 \leq 3$
3	$2X_1 + 3X_3 + 2X_4 \leq 3$
4	$2X_3 + 3X_4 + 2X_5 \leq 3$
5	$2X_2 + 3X_5 + 2X_6 \leq 3$
7	$X_4 + 4X_7 + 2X_8 + 2X_9 \leq 4$
8	$X_5 + 2X_7 + 4X_8 + 2X_9 \leq 4$

where the a_1, \dots, a_n is an increasing sequence of positive integers that satisfy

$$a_i > (m - 1)(a_1 + \dots + a_{i-1}) \text{ for } i \geq 2$$

However, this aggregator yields values for a_i that increase rapidly (Fishburn and Hochenberg 1985). Our aggregator is based on the idea that we can form aggregators less restrictive than aggregator [17], since the adjacency constraints are inequalities.

Consider the pattern depicted in Fig. 1. All the conventional adjacency constraints that include area 1 are

$$[18] \quad X_1 + X_2 \leq 1$$

$$[19] \quad X_1 + X_3 \leq 1$$

$$[20] \quad X_1 + X_4 \leq 1$$

Assuming $a_i = 1$ and applying the aggregator in [17], a possible aggregator for constraints [18] and [20] is given by

$$[21] \quad 2X_1 + X_2 + X_4 \leq 2$$

which works because areas 2 and 4 are not adjacent to each other. Now consider if we attempt to aggregate constraints [19] and [20]. Since area 3 is adjacent to areas 2 and 4, we need to consider the set of constraints

$$[22] \quad X_2 + X_3 \leq 1$$

$$X_3 + X_4 \leq 1$$

Then, our penalty procedure tries to consider the set of constraints [22], keeping the increasing sequence of coefficients whenever areas adjacent to areas already considered in the aggregated constraint are added. Hence the penalties keep the increasing sequence of coefficients for the noncolorable areas.

Comparisons

To compare the performance of our algorithm, we formulated adjacency constraints for five examples by using two other algorithms. The first algorithm was the conventional formulation that consists of writing pair-wise adjacent relationships in each constraint such that only two areas are involved in every adjacency constraint. For instance, following this formulation and the pattern depicted in Fig. 4, the constraints that involve area 1 are

$$X_1 + X_2 \leq 1$$

$$X_1 + X_4 \leq 1$$

Thus for the pattern in Fig. 4 and following the conventional formulation, we need 12 constraints to express all the adjacency relationships.

The second algorithm was the one described by Meneghin *et al.* (1988). This algorithm consists of three basic steps: (i) Identify all type 1 constraints. This step is basically to identify all the pairs, triplets, or quadruplets. (ii) Identify

all type 2 constraints. This step consists of applying some rules to identify groups of pairs (triplets or quadruplets) of type 1 constraints for each type 1 constraint identified in the first step. (iii) According to another set of rules, select the best set of type 2 constraints and following an additional set of rules, write the adjacency constraints for the selected combinations of type 2 constraints. The procedure is almost tedious since the process of identifying type 1 constraints and all possible combinations of these constraints to form a type 2 constraint requires some combinatorial work. For the pattern depicted in Fig. 4 and following this procedure to formulate adjacency constraints, we need just four constraints to describe all the adjacency relations. Those constraints correspond to inequalities 2, 4, 6, and 8 in Table 3.

In our example forests, each forest varied in complexity not only in the number of stands but also in the adjacency relations. For each example forest, we formulated adjacency constraints using the two algorithms described earlier and our heuristic. The number of constraints used in each case was then counted. Table 5 shows these results. As can be observed, our procedure requires about a third of the number of constraints required by the traditional procedure and about the same number of constraints as required by Meneghin's algorithm. A close look at Table 5 could lead to the misinterpretation that Meneghin's algorithm performs better as we increase the number of units in the forest. However, this is not true. Such differences depend basically on the structure of the forest. For structures similar or close to a checkerboard (small chromatic number) and where the number of adjacency relations per area is small, our algorithm performs better. It requires fewer constraints and smaller coefficients for the aggregated constraints. However, when the number of adjacency relations increases and especially in those cases where it is possible to form a large number of quadruplets (not very common in real forest structures), Meneghin's algorithm is better since it yields fewer constraints than ours and smaller coefficients in each constraint. In these cases, the difference in the number of constraints was about 20% of the number of constraints required by our algorithm. Observe that in such situations Meneghin's algorithm uses its property of grouping many adjacency relations for several units in one single constraint. However, in patterns where there are few quadruplets, this property is not exploited. In most of the cases we obtained fewer variables per constraint than Meneghin's constraints, although the coefficients obtained with our heuristic were often larger than those of Meneghin's algorithm. We did not write any computer code to compute the adjacency con-

TABLE 5. Number of adjacency constraints required for three different algorithms in some example problems

No. of units	Conventional formulation	Meneghin's formulation	Proposed heuristic
15	32	10	9
20	37	13	12
30	71	22	22
35	73	23	25
40	90	26	31

straints from the adjacency matrix for any of the algorithms. However, it is evident that the time required to formulate adjacency constraints by our procedure is not shorter than that of the conventional procedure, but it is if compared to Meneghin's algorithm, given that our procedure does not involve any combinatorial selection.

Another important feature to note in our algorithm is the ease with which it is mechanically generated from the adjacency matrix. This characteristic is highly evident for procedure 1, but not so much for procedure 2. However, if we consider that there exists a small number of combinations of values of coefficients for the four noncolorable areas in each constraint (when the reference area has at least four noncolorable areas), then we can store those combinations and select one any time we enter a variable in the aggregated constraint. Hence, procedure 2 can also be mechanically generated.

Reduction of adjacency constraints in area-based forest planning is important not only to fit large formulations into standard linear programming or integer programming solution packages but also to fit large problems to some heuristics developed to provide good feasible solutions to the area-based harvest scheduling problem. For instance, our procedure can be applied to the heuristic developed by Nelson *et al.* (1988) or the one developed by Sessions (1988) to reduce the size of the adjacency matrix that is stored, since this matrix can be reduced if we use aggregated constraints. Another important application of aggregated constraints is our attempt to obtain approximate optimal solutions to the area-based problem through relaxation, namely surrogate relaxation and Lagrangean relaxation. In these cases, the smaller the number of constraints the faster the computation of the multipliers.

In summary, we have shown that the reduction of adjacency constraints is an important tool in area-based forest planning. It can be applied to heuristics or optimal solution algorithms to reduce storage limitations or to reduce solution time for highly structured integer programming

problems (Kannan 1983), which are the main problems in area-based formulations of the harvest scheduling problem.

Acknowledgement

Research was partially supported by the Forestry Research Laboratory, Oregon State University, and the Coastal Oregon Productivity Enhancement Program.

- APPEL, K.I., and HAKEN, W. 1976. Every planar map is four colorable. *Bull. Am. Math. Soc.* **82**: 711-712.
- BRADLEY, G.H. 1971. Transformation of integer programs to knapsack problems. *Discrete Math.* **1**(1): 29-45.
- FISHBURN, P.C., and KOCHENBERG, G.A. 1985. Aggregating assignment constraints. *Nav. Res. Log.* **32**(4): 653-663.
- GROSS, T.E. 1989. Use of graph theory to analyze constraints on the juxtaposition of timber stands. M.S. thesis, School of Forestry, Northern Arizona University, Flagstaff, AR.
- GROSS, T.E., and DYKSTRA, D.P. 1988. Harvest scheduling with nonadjacency constraints. *In Proceedings, Society of American Foresters National Convention, 16-19 Oct. 1988, Washington, DC. Society of American Foresters, Washington, DC.* pp. 310-315.
- KANNAN, R. 1983. Polynomial time aggregation of integer programming problems. *J. Assoc. Comput. Mach.* **30**(1): 133-145.
- MENEGHIN, B.J., KIRBY, M.W., and JONES, J.G. 1988. An algorithm for writing adjacency constraints efficiently in linear programming models. *In The 1988 Symposium on Systems Analysis in Forest Resources, Mar. 29-Apr. 1, 1988. Edited by B. Kent and L. Davis. USDA For. Serv. Rocky Mt. For. Range Exp. Stn. Gen. Tech. Rep. RM-161.* pp. 46-53.
- NELSON, J., BRODIE, J.D., and SESSIONS, J. 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. Edited by B. Kent and L. Davis. USDA For. Serv. Rocky Mt. For. Range Exp. Stn. Gen. Tech. Rep. RM-161.* pp. 224-229.
- ONEYKWELU, D.C. 1983. Computational viability of a constraint aggregation scheme for integer programming problems. *Oper. Res.* **31**(3): 795-801.
- PADBERG, M.W. 1972. Equivalent knapsack-type formulations of bounded integer linear programs: an alternative approach. *Nav. Res. Log.* **19**(4): 699-708.
- RINGEL, G. 1959. Färbungsprobleme auf Flächen und Graphen. *Mathematische Monographien Numer 2.* VEB Deutscher Verlag der Wissenschaften, Berlin.
- SAATY, T.L., and KAINEN, P.C. 1977. *The four-color problem.* McGraw-Hill Inc., Maidenhead, Great Britain.
- SESSIONS, J. 1988. User's guide for Scheduling and Network Analysis Program (SNAP). [Draft] USDA Forest Service, Division of Timber Management, Portland, OR.
- WOOD, D.C. 1969. A technique for colouring a graph applicable to large scale timetabling problems. *Comp. J.* **12**: 317-319.