Evaluating Approaches for Solving the Area Restriction Model in Harvest Scheduling

Marcos Goycoolea, Alan Murray, Juan Pablo Vielma, and Andres Weintraub

Abstract: We survey three integer-programming approaches for solving the area restriction model for harvest scheduling. We describe and analyze each of these approaches in detail, comparing them both from a modeling and computational point of view. In our analysis of these formulations as modeling tools, we show how each can be extended to incorporate additional harvest scheduling concerns. In our computational analysis we illustrate the strengths and weaknesses of each formulation as a practical optimization tool by studying harvest scheduling in four North American forests. FOR. SCI. 55(2):149–165.

Keywords: adjacency, area restriction model, green-up, optimization, integer programming

FOREST FIRMS FACE THE CHALLENGE of adapting current harvest scheduling methodologies to meet growing environmental concerns, particularly those of protecting species and natural habitat, while minimally foregoing economic gains. A common approach for incorporating such wildlife concerns consists of restricting the local extent of harvest activity by imposing a limit on the maximum contiguous area of clearcut regions (Thompson et al. 1973).

To model these types of problems, forest planners rely on spatially explicit models to represent forests and harvest scheduling plans. More specifically, geographic information systems are used to partition forests into stands representing subregions with homogeneous characteristics, and decisions are made at a stand level. In this way, the design of a harvest schedule is abstracted as a combinatorial optimization problem, with decision variables corresponding to stands for potential harvest at a particular time and for which the objective may be maximizing profits or an alternative, subject to operational and environmental constraints. The constraint we are concerned with in this article is prohibiting contiguous groups of stands from exceeding the prescribed clearcut area limit.

Approaches to solve this problem initially were relatively simple and relied on an experienced planner to manually predefine disjoint feasible "clusters" (contiguous groups of stands), each satisfying the maximum contiguous area restriction. This was done in such a way that if two clusters were adjacent, then their combined area would exceed the maximum opening size restriction. Thus, the harvest scheduling problem was reduced to selecting from these predefined clusters a subset that would maximize revenue, subject to the constraint that no two adjacent clusters be simultaneously selected. It is clear that this approach was highly dependent on how the clusters were predefined. Murray (1999) referred to this approach as the unit restriction model. A number of exact formulation approaches for this model have been pursued: Barahona et al. (1992) and Weintraub et al. (1994) proposed a column generation scheme, Murray and Church (1996, 1997) proposed using cliques to directly solve a set packing problem, and, Hoganson and Borges (1998) proposed a dynamic programming approach. Heuristic and meta-heuristic approaches have also been proposed, including Tabu search (Murray and Church 1995) and simulated annealing (O'Hare 1989, Nelson and Brodie 1990, Murray and Church 1995).

In addition, Hokans (1983) and Lockwood and Moore (1993) proposed incorporating the construction of clusters from stands in the decision model, as opposed to having these clusters be predefined. Murray (1999) referred to this approach as the area restriction model (ARM). Murray and Weintraub (2002) empirically showed that solution profit value can be improved through the addition of cluster forming in the ARM. However, the ARM is a significantly more complex problem. Until recently, most solution approaches were heuristic or meta-heuristic in nature (see Hokans 1983, Lockwood and Moore 1993, Barrett et al. 1998, Clark et al. 2000, Richards and Gunn 2000, Boston and Bettinger 2002, Caro et al. 2003). Lately, however, several exact integer programming formulations have been proposed for solving the problem. One formulation, which we henceforth call the Path formulation, is based on enumerating all possible ways in which a harvesting cluster may be infeasible and defining constraints that prohibit each of these infeasibilities (see McDill et al. 2002, Crowe et al. 2003, Gunn and Richards 2005a). A second formulation, which we henceforth call the Cluster Packing formulation, is based on defining variables for all possible feasible harvesting clusters and defining

Marcos Goycoolea, School of Business, Universidad Adolfo Ibanez, Diagonal Las Torres 2640, Penalolen, Santiago, Chile—Phone: 56-2-331-1274; marcos.goycoolea@uai.cl. Alan Murray, Arizona State University—alan.t.murray@asu.edu. Juan Pablo Vielma, Georgia Institute of Technology—jvielma@isye.gatech.edu. Andres Weintraub, Universidad of Chile—aweintra@dii.uchile.cl.

Acknowledgments: We acknowledge Klaus Barber (US Forest Service), John Nelson (University of British Columbia), and Evelyn Richards (University of New Brunswick), for making the forests we used available for analysis. We also acknowledge the help of the referees who have greatly contributed to improving this document. Part of this research was funded by Fondecyt under Grants 1060807 and 11075028, the Millennium project for Complex Engineering Systems and the Basal project CMM, Universidad de Chile.

Manuscript received March 30, 2007, accepted December 16, 2008

constraints that prevent any selected pair from being overlapping or adjacent (see McDill et al. 2002, Murray et al. 2004, Goycoolea et al. 2005, Martins et al. 2005, Vielma et al. 2007). A third formulation, which we call the Bucket formulation, is based on defining buckets a priori and then assigning stands to each bucket so that each bucket represents a harvested cluster (see Constantino et al. 2006).

In this article we review and compare these three mixed-integer programming formulations for the ARM and discuss some basic extensions. We consider both modeling and computational issues. In terms of modeling, we discuss different spatial considerations that can be incorporated into the formulations, such as green-up, minimum size, maximum average size, shape constraints, and others. In terms of computational issues our focus is on performance.

It is well known that most integer programming problems can be formulated in different ways and that the properties of these different formulations can significantly affect their performance. One such issue is the formulation size. Another less trivial issue is tightness of the formulation. There are several ways of defining the notion of tightness. Given that we consider formulations defined on different variables; we compare their linear programming (LP) relaxation values, that is, the optimal value obtained when the formulation is solved without imposing the integrality constraints. This process defines a natural bound on the optimal value and indicates how good a proxy the relaxation value is to the actual optimal, which is useful as a way of validating solutions found with heuristic methods and is crucial for the effectiveness of Branch and Bound algorithms. For more information, see Nemhauser and Wolsey (1988). A final measure that we consider is the amount of time it takes to find an optimal (or near-optimal) solution. If this solution cannot be achieved in a reasonable amount of time, we consider instead what is the best solution found after an allotted time.

This article is organized as follows. In the second section we introduce the ARM and the notation that will be used. We then describe the three integer programming formulations of the ARM. We discuss implementation issues relating to these formulations and compare the theoretical quality of the LP relaxation bounds they provide. In the third section we discuss extensions of the ARM and how these may be incorporated into the integer programming formulations. In the fourth section we present computational results obtained by testing these approaches using four forest planning applications. We focus on analyzing the size of the different formulations, the strength of the LP relaxation bounds they provide, the difficulty of solving each formulation to optimality, and the effect of incorporating model extensions. Finally, in the last section the relative advantages of each approach are discussed and possible future directions of research are noted.

The ARM

Throughout this article we convene on the following notation to formally describe the problem. Let S be the number of stands in the forest, and let T represent the number of time periods considered. Assume that each time period represents Y years. To each stand s we assign the following attributes: a_s is area of stand s, $p_{s,t}$ is revenue obtained when stand s is harvested in time t, $\alpha_{s,t}$ is the volume of timber obtained when stand s is harvested in time t, and g_s is the initial age of stand s (i.e., the age of stand s in time $t = 1$).

Let E represent all pairs of adjacent stands. That is, if r, s are adjacent stands, then $\{r, s\} \in E$. Let $N(s)$ represent the set of all stands that are adjacent to s. That is, stand r belongs to set $N(s)$ if and only if r is adjacent to s or, equivalently, if $\{r, s\} \in E$.

To illustrate these concepts consider Figure 1. In this example, $S = 15$. If we define two stands as being adjacent if they touch, then we have that the pair $\{1, 4\}$ is in E but that the pair $\{1, 9\}$ is not. Note also for this example that $N(4) = \{1, 5, 9, 10\}$.

We say that a set C of contiguous stands defines a *cluster* of stands. For each cluster C we define its total area as $a_C = \sum_{s \in C} a_s$. Likewise, we define the revenue obtained from harvesting cluster C at time t as $p_{C,t} = \sum_{s \in C} p_{s,t}$. Finally, let $\alpha_{C,t} = \sum_{s \in C} \alpha_{s,t}$ equal the volume of timber obtained when harvesting cluster C at time t.

The ARM consists of selecting, for each time period, a set of stands to be harvested so as to maximize revenue subject to the following constraints:

Volume yield constraints. A typical requirement in forestry operations is that a "forest produces a nondeclining even-flow of timber" (Buongiorno and Gilless, 2003, p. 70) or a "reasonable yield pattern" (Ware and Clutter, 1971, p. 436). For each time period t, the amount of timber extracted should be no more than U_t times (and no less than L_t times) what was extracted in the previous time period. Variants of this constraint might consider imposing of minimum and maximum volume requirements in different time periods.

Average ending age constraints. The average age of standing timber at the end of the planning horizon should be at least $\bar{G}$ years.

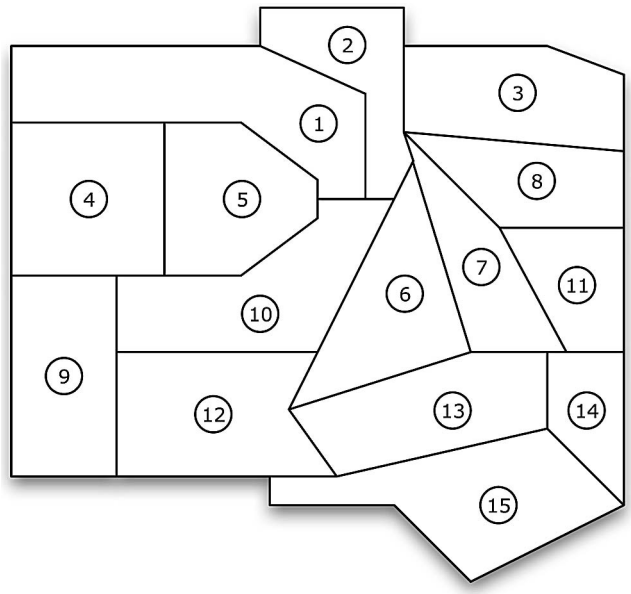

Figure 1.

Harvest-once constraints. Each stand should be harvested at most once during the planning horizon. Here it is implicitly assumed that we are only concerned with the first harvest and that replanted stands will not reach a profitable age in the planning horizon.

Maximum clearcut size constraints. No contiguous group of stands harvested in a same time period may exceed an area limit of A.

Henceforth we say that a cluster C is *feasible* if it satisfies $a_C \leq A$. Otherwise we say that C is *infeasible*. With this definition, the maximum clearcut constraint can be equivalently described as prohibiting infeasible clusters from being harvested in all of the time periods. Note that the definition of a feasible cluster coincides with the definition of g*eneralized management unit* as proposed by McDill et al. (2002).

The maximum clearcut size constraint can be visualized using the hypothetical forest illustrated in Figure 1. For simplicity, assume that all stands have an area of 10 ha, that a maximum opening size of $A = 30$ ha is stipulated, and that we are considering a single time period. Given these assumptions, the cluster defined by stands 4, 5, and 9 is feasible. However, if any additional neighboring stand were added, the resulting cluster would be infeasible. For example, cluster {4, 5, 9, 10} is infeasible, as it has a total area of 40 ha. In this way, if we momentarily ignore volume flow and average ending age constraints, it can be seen that the set of shaded stands illustrated in Figure 2 defines a feasible solution.

We now describe three different integer programming formulations that have been proposed for this problem. The focus of this article is on these maximum clearcut size constraints.

The Path Formulation

The Path formulation, originally proposed in McDill et al. 2002, works by defining for each stand s and each time period t a binary variable $y_{s,t}$ taking value 1 if stand s is to be harvested in period t and value 0 if not. To see how the maximum clearcut size condition can be imposed using these variables, let us consider again the hypothetical forest of the previous section, and a candidate solution in which the stands {1, 2, 3, 4, 5} are all harvested in time period $t = 1$. Because the total area of this cluster is 50, and the maximum clearcut size is $A = 30$, this cluster would be infeasible. Thus, to ensure a feasible solution, we need to prohibit that all these stands be harvested simultaneously in $t = 1$. For this, it suffices to add the constraint,

$$y_{1,1} + y_{2,1} + y_{3,1} + y_{4,1} + y_{5,1} \leq 4.$$

This idea can be generalized in the following manner. Let Λ represent the set of all infeasible clusters. That is, $(C \in \Lambda)$ if and only if C corresponds to a contiguous set of stands such that $(a_C > A)$. To prohibit all infeasible clusters from being selected add the family of constraints,

$$\sum_{s \in C} y_{s,t} \leq |C| - 1 \quad \forall C \in \Lambda, \quad \forall t = 1, \ldots, T.$$

A concern regarding this formulation is that the set Λ of infeasible clusters may be exponentially large, thus rendering this formulation impractical. However, it is possible to greatly reduce the size of this formulation by using only those constraints that are strictly necessary. Consider two infeasible clusters C and D such that $C \subseteq D$. It is easy to see that by prohibiting the harvest of cluster C, the harvest of cluster D is also prohibited. Going back to the previous example, observe that to prevent {1, 2, 3, 4, 5} from being harvested in $t = 1$ it would suffice to add the constraint,

$$y_{1,1} + y_{2,1} + y_{3,1} + y_{4,1} \leq 3.$$

Formally, we can generalize this concept as follows. Define a cluster $C \in \Lambda$ to be *minimally infeasible* if C is infeasible and if for every $s \in C$ either $C - \{s\}$ is a feasible cluster or $C - \{s\}$ is not contiguous, that is, if removing any stand from C we obtain another cluster that is either feasible or the resulting set of stands is disconnected. If C is minimally infeasible, then C cannot contain any other infeasible clusters. Thus, if we define $\Lambda^+ = \{C \in \Lambda : C$ is minimally infeasible$\}$ we can instead use the so-called Path inequalities,

$$\sum_{s \in C} y_{s,t} \leq |C| - 1 \quad \forall C \in \Lambda^+, \quad \forall t = 1, \ldots, T.$$

This leads us to the Path formulation of the ARM (McDill et al. 2002):

$$\max \sum_{t=1}^{T} \sum_{s=1}^{S} p_{s,t} y_{s,t} \tag{1}$$

subject to

$$\sum_{t=1}^{T} y_{s,t} \leq 1 \quad \forall s = 1, \ldots, S \tag{2}$$

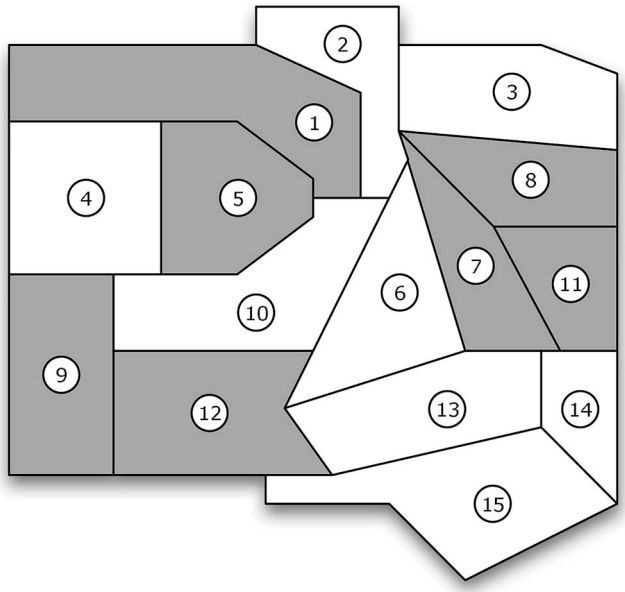

Figure 2.

$$\sum_{s \in C} y_{s,t} \leq |C| - 1 \quad \forall C \in \Lambda^+, \quad \forall t = 1, \ldots, T \qquad (3)$$

$$\sum_{s=1}^{S} \alpha_{s,t+1}\, y_{s,t+1} \leq U_t \sum_{s=1}^{S} \alpha_{s,t}\, y_{s,t}$$

$$\forall t = 1, \ldots, T-1 \qquad (4a)$$

$$\sum_{s=1}^{S} \alpha_{s,t+1}\, y_{s,t+1} \geq L_t \sum_{s=1}^{S} \alpha_{s,t}\, y_{s,t}$$

$$\forall t = 1, \ldots, T-1 \qquad (4b)$$

$$\sum_{s=1}^{S} a_s \left(g_s + T \cdot Y - \sum_{t=1}^{T} (t \cdot Y + g_s) y_{s,t} \right) \geq \overline{G} \sum_{s=1}^{S} a_s \qquad (5)$$

$$y_{s,t} \in \{0, 1\} \quad \forall s = 1, \ldots, S, \quad \forall t = 1, \ldots, T \qquad (6)$$

The objective function 1 consists of maximizing the total harvest revenue. Constraints 2 impose that each stand be harvested at most once during the planning horizon. Constraints 3 impose the maximum clearcut size condition. Constraints 4 impose the volume flow constraints. Constraints 5 impose the average ending age condition. Constraints 6 impose integrality on decision variables.

We say that set K of stands defines a *clique* if every pair of stands $r, s \in K$ are adjacent to each other. A clique K is *maximal*, if no other clique contains it. Observe that the sets {1, 4, 5, 10} and {6, 7, 13} in Figure 1 each define a maximal clique. Let Π denote the set of all maximal cliques. Crowe et al. (2003) suggested adding the following family of constraints to strengthen the problem formulation:

$$\sum_{s \in K} a_s y_{s,t} \leq A \quad \forall K \in \Pi, \quad \forall t = 1, \ldots, T. \qquad (7)$$

Observe that constraints 7 need only be defined for those cliques whose total area exceeds the prescribed limit. Computational results presented in Crowe et al. (2003) suggest that this strengthening approach is not very effective in practice. However, by building on this idea, Gunn and Richards (2005a) proposed another way of tightening the Path formulation by adding only one constraint per stand. The basic idea is based on the following observation: if a stand s is harvested in period t, then the total area adjacent to stand s that can be harvested is bounded by $(A - a_s)$. The constraints they proposed are as follows:

$$\sum_{r \in N(s)} a_r y_{r,t} \leq M_s (1 - y_{s,t}) + (A - a_s)$$

$$\forall s = 1, \ldots, S, \quad \forall t = 1, \ldots, T. \qquad (8)$$

As an example, consider again the forest depicted in Figure 1. For stand 4, time period t, this constraint would take the form:

$$10y_{1,t} + 10y_{5,t} + 10y_{9,t} + 10y_{10,t} \leq M_4(1 - y_{4,t}) + 20.$$

Gunn and Richards (2005a) described different ways by which to compute suitable coefficients M_s, as well as a lifting algorithm by which to strengthen these constraints. Gunn and Richards (2005a) observed that by replacing constraints 3 with constraints 8, they were able to generate optimal feasible solutions for several test cases. However, they pointed out that such an approach will not always yield feasible solutions. Consider the forest depicted in Figure 3, and assume that the area of stands 1, 2, 3, and 4 is 20 ha each. Further, assume that $A = 60$.

A cluster consisting of stands 1, 2, 3, and 4 exceeds the maximum opening size, yet no constraint of type 8 excludes

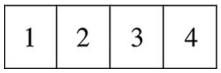

Figure 3.

it. Regardless, constraints 8 can be effective for improving the performance of the Path formulation when combined with constraints 3, or constraints 8 can be used alone to generate approximate solutions (when individual stand sizes are large relative to the maximum clearcut size) as proposed by Gunn and Richards (2005a).

An important issue pertaining to the use of the Path formulation is the enumeration of the minimally infeasible clusters that are needed to define constraints 3. McDill et al. (2002) proposed what they called the Path algorithm for achieving this. In Appendix I we present an alternative algorithm that is easier and that we have found to work better in practice. In Appendix I we also discuss how this algorithm can be modified to generate the set of all maximal cliques and other sets of stands.

The Cluster Packing Formulation

An alternative way to formulate the ARM problem is based on focusing on feasibilities rather than infeasibilities. This focus can be achieved by defining a 0-1 decision variable for every feasible cluster (taking value 1 if we should select it and 0 otherwise), and defining constraints in such a way as to ensure that the selected clusters are disjoint and separated one from another.

More formally, for each feasible cluster C and every time period t, let binary variable $x_{C,t}$ indicate whether cluster C is to be harvested in time period t or not. We say that two feasible clusters C and D are *incompatible* if they intersect or are adjacent. The Cluster Packing model imposes the maximum area constraint by forcing selected clusters to be compatible with one another.

A natural way to achieve this compatibility is to explicitly prohibit incompatibility by using pairwise constraints (McDill et al. 2002, Goycoolea et al. 2005). That is, if two feasible clusters C and D are incompatible, one could impose for each time period t constraints of the form

$$x_{C,t} + x_{D,t} \leq 1.$$

Let Ω represent the set of all feasible clusters. For each set of stands U, let $\Omega(U)$ represent the set of all feasible clusters containing at least one stand in U and define $\Omega(s) = \Omega(\{s\})$ for each stand s. The Pairwise Cluster Packing

formulation for the ARM is defined as

$$\max \sum_{t=1}^{T} \sum_{C \in \Omega} P_{C,t}\, x_{C,t} \tag{9}$$

subject to

$$\sum_{t=1}^{T} \sum_{C \in \Omega(s)} x_{C,t} \leq 1 \quad \forall s = 1, \ldots, S \tag{10}$$

$$x_{C,t} + x_{D,t} \leq 1$$

$$\forall C, D \in \Omega \text{ with } C, D \text{ incompatible} \quad \forall t = 1, \ldots, T \tag{11}$$

$$\sum_{C \in \Omega} \alpha_{C,t+1} x_{C,t+1} \leq U_t \sum_{C \in \Omega} \alpha_{C,t} x_{C,t}$$

$$\forall t = 1, \ldots, T-1 \tag{12a}$$

$$\sum_{C \in \Omega} \alpha_{C,t+1} x_{C,t+1} \geq L_t \sum_{C \in \Omega} \alpha_{C,t} x_{C,t}$$

$$\forall t = 1, \ldots, T-1 \tag{12b}$$

$$\sum_{s=1}^{S} a_s \left(g_s + T \cdot Y - \sum_{t=1}^{T} (t \cdot Y + g_s) \sum_{C \in \Omega(s)} x_{C,t} \right) \geq \bar{G} \sum_{s=1}^{S} a_s \tag{13}$$

$$x_{C,t} \in \{0, 1\} \quad \forall C \in \Omega, \quad \forall t = 1, \ldots, T \tag{14}$$

The objective function, 9, is to maximize total harvest revenue. Constraints 10 impose that each stand should be harvested at most once during the planning horizon. Constraints 11 impose that incompatible pairs of clusters should not be selected in the same time period, and constraints 12 impose the volume flow constraints. Constraints 13 impose the average ending age condition. Constraints 14 impose integrality on decision variables. If we replace constraints 14 with the constraint,

$$0 \leq x_{C,t} \leq 1 \quad \forall C \in \Omega, \quad \forall t = 1, \ldots, T, \tag{14b}$$

we obtain what is called the LP relaxation of this formulation. In Appendix I we describe how to algorithmically generate the set Ω required for formulating the problem.

Observe that there may exist two feasible clusters whose union defines yet another feasible cluster. This can happen if the clusters are disjoint and adjacent to each other and if their combined area does not exceed the prescribed maximum. A common concern is that constraints 11 would prohibit these from being simultaneously harvested. It should be noted that this is not a problem because the cluster defined by taking the union of these smaller clusters also has a variable associated to it. Although this formulation is valid, it is possible to obtain others that, building on the same idea, provide much tighter LP bounds.

To see this, consider a pair of adjacent stands r, s, that is, a pair $\{r, s\}$ in the set E. These pairs will henceforth be referred to as *edges.* In a feasible ARM solution there can be at most one selected cluster intersecting the edge $\{r, s\}$. In fact, if there are two clusters intersecting the set $\{r, s\}$, then either they both contain a same stand, or, otherwise, they are disjoint but adjacent to each other. In both cases, they are incompatible. From the preceding analysis, the following family of constraints is valid for the ARM (see Martins et al., 2005):

$$\sum_{C \in \Omega(\{r,s\})} x_{C,t} \leq 1 \quad \forall \{r, s\} \in E, \quad \forall t = 1, \ldots, T. \tag{11b}$$

If we replace constraints 11 with constraints 11b, we obtain what we call the Edge Cluster Packing formulation, which is tighter than the Pairwise Cluster Packing formulation.

An even tighter formulation was developed by Martins et al. (2005) and Goycoolea et al. (2005). As before, let Π denote the set of all maximal cliques in the forest. Constraints 11b can be substituted with the constraints

$$\sum_{C \in \Omega(K)} x_{C,t} \leq 1 \quad \forall K \in \Pi, \quad \forall t = 1, \ldots, T. \tag{11c}$$

The reason is that two clusters intersect a common edge if and only if they intersect a common clique. When this substitution is used, we obtain the Clique-Cluster Packing formulation. Going back to the example depicted in Figure 1, {6, 7, 13} defines a clique. This means that among the feasible clusters {6, 7, 13}, {2, 6, 10}, {3, 8, 7}, {7, 13, 14}, {6, 12}, {13, 14}, and all others that intersect {6, 7, 13}, only one can be chosen. Recall that an algorithm by which to enumerate all maximal cliques is described in Appendix I.

Not only is the Clique-Cluster Packing formulation tighter than the Edge Cluster Packing formulation, but also results detailed in Goycoolea et al. (2005) show that the Clique-Cluster Packing formulation is substantially faster to solve. Goycoolea et al. (2005) also introduced a methodology called *constraint projection* for deriving additional families of valid inequalities for this problem (see Nemhauser and Wolsey 1988). Finally, Goycoolea et al. (2005) describe a methodology for strengthening constraints 11c by a clique-lifting procedure. Computational tests, however, indicate that these constraints are very effective and do not need such strengthening.

In the computational results section, we shall see that the LP relaxation bounds afforded by the Cluster Packing formulation are strictly better than those afforded by the Path formulation in all but one of the instances tested (for example, in Table 5, see El Dorado with $T = 1$). A natural question is whether or not the Cluster Packing formulation can have a worse LP relaxation bound than the Path formulation. The answer, remarkably, is no. That the LP relaxation value of the Path formulation will always be larger than or equal to that of the Cluster Packing formulation is proved as a mathematical theorem in Appendix II. That in practice it is very often strictly larger can be seen in the computational results. An important implication of this result is that the Clique Cluster Packing formulation is better suited than the Path formulation as a tool for validating heuristics and estimating quality of solution values.

The Bucket Formulation

Consider an instance of ARM and observe that in any given time period, one can never simultaneously harvest

more clusters than there are stands in the forest. This constraint naturally follows from the fact that harvested clusters must be disjoint and nonadjacent to each other and suggests yet a third way in which the ARM can be formulated.

For each stand i define a "bucket" B_i. The idea is that stands should be assigned to these buckets in such a way that (a) each stand is assigned to at most one bucket, (b) the total area of stands assigned to a same bucket does not exceed the prescribed maximum, and (c) buckets should be nonadjacent to each other. Consider such an assignment of stands to buckets and consider any given bucket. If the bucket is nonempty and the stands in the bucket are all connected to each other, then the bucket defines a feasible cluster. Otherwise, break up the stands of the bucket into connected components. Given condition b, each such component defines a feasible cluster. Finally, because of condition c, all of the clusters obtained from the stand-to-bucket assignment will be compatible with each other, and so one obtains feasible ARM solutions through these assignments. Moreover, it is easy to see that given any feasible ARM solution, it is possible to construct such a stand-to-bucket assignment by simply letting each feasible cluster define a bucket.

Let us now formalize this idea for multiple time periods by means of an integer programming model. For each stand s, period t and bucket B_i define a binary decision variable y_s^{it} indicating whether or not stand s will be assigned to bucket B_i in time period t. For each $e \in E$, $t = 1, \ldots, T$, and $i = 1, \ldots, S$ define a binary decision variable w_e^{it} indicating whether at least one of the end points of edge e has been assigned to bucket B_i in period t.

With these variables it is possible to use the following integer programming formulation for the ARM problem (see Constantino et al. 2006):

$$\max \sum_{t=1}^{T} \sum_{s=1}^{S} \sum_{i=1}^{S} p_{s,t}\, y_s^{it} \tag{15}$$

subject to

$$\sum_{t=1}^{T} \sum_{i=1}^{S} y_s^{it} \le 1 \quad \forall s = 1, \ldots, S \tag{16}$$

$$\sum_{i=1}^{S} w_e^{it} \le 1 \quad \forall e \in E, \quad \forall t = 1, \ldots, T \tag{17}$$

$$y_s^{it} - w_e^{it} \le 0 \tag{18}$$

$$\forall e \in E, \quad \forall s \in e, \quad \forall i = 1, \ldots, S, \quad \forall t = 1, \ldots, T$$

$$\sum_{s=1}^{S} a_s y_s^{it} \le A \quad \forall i = 1, \ldots, S \tag{19}$$

$$\sum_{i=1}^{S} \sum_{s=1}^{S} \alpha_{s,t+1} y_s^{i,t+1} \le U_t \sum_{i=1}^{S} \sum_{s=1}^{S} \alpha_{s,t} y_s^{i,t}$$

$$\forall t = 1, \ldots, T-1 \tag{20a}$$

$$\sum_{i=1}^{S} \sum_{s=1}^{S} \alpha_{s,t+1} y_s^{i,t+1} \ge L_t \sum_{i=1}^{S} \sum_{s=1}^{S} \alpha_{s,t} y_s^{i,t}$$

$$\forall t = 1, \ldots, T-1 \tag{20b}$$

$$\sum_{s=1}^{S} a_s \left(g_s + T \cdot Y - \sum_{t=1}^{T} (t \cdot Y + g_s) \sum_{i=1}^{S} y_s^{it} \right) \ge \bar{G} \sum_{s=1}^{S} a_s \tag{21}$$

$$y_s^{it} \in \{0, 1\}$$

$$\forall i = 1, \ldots, S, \quad \forall s = 1, \ldots, S, \quad \forall t = 1, \ldots, T \tag{22}$$

$$w_e^{it} \ge 0$$

$$\forall e \in E, \quad \forall i = 1, \ldots, S, \quad \forall t = 1, \ldots, T \tag{23}$$

The objective function, 15, is to maximize the total harvest revenue. Constraints 16 impose that each stand can be assigned at most to a single bucket throughout the planning horizon. Constraints 17 impose that clusters are pairwise nonadjacent. This is achieved by imposing that at most one bucket intersects each edge in the forest. Constraints 18 impose the relationship between the y and w variables. That is, each time a stand is assigned to a bucket, the variable corresponding to the edges that it touches must be triggered. Constraints 19 impose that the total area of stands in a bucket cannot exceed A. Constraints 20 impose the volume flow condition, and constraints 21 impose the average ending age condition. Constraints 22 impose integrality of the y variables. Constraints 23 impose non-negativity of the w variables. Note that it is not necessary to impose integrality for the w variables, as a feasible schedule will be fully determined by the y variables.

As with the Cluster Packing formulation, it is observed in Constantino et al. (2006) that a tighter formulation can be obtained replacing edges by cliques. For this, define binary decision variables W_K^{it} for each $i = 1, \ldots, S$, each $t = 1, \ldots, T$, and each $K \in \Pi$, indicating whether at least one of the stands in clique K has been assigned to bucket B_i in period t. Then, substitute constraints 17, 18, and 23 by

$$\sum_{i=1}^{S} W_K^{it} \le 1 \tag{17b}$$

$$\forall K \in \Pi, \quad \forall t = 1, \ldots, T$$

$$y_s^{it} - W_K^{it} \le 0 \tag{18b}$$

$$\forall K \in \Pi, \quad \forall s \in K, \quad \forall i = 1, \ldots, S, \quad \forall t = 1, \ldots, T$$

$$W_K^{it} \ge 0 \tag{23b}$$

$$\forall K \in \Pi, \quad \forall i = 1, \ldots, S, \quad \forall t = 1, \ldots, T$$

A very important feature of this formulation is that the number of rows and columns is independent of the maximum clearcut size A. As the value of A increases, all that needs to change in this formulation is the right-hand side of

constraints 19. In the Path formulation, the number of constraints increases exponentially with A, and in the Cluster Packing formulation, the number of columns increases exponentially with A. This makes the Bucket formulation especially significant for problems in which the average stand size relative to the maximum clearcut size is small.

Despite this fact, however, the formulation can still be very large in practice. To make up for this, Constantino et al. (2006) proposed a series of preprocessing ideas to make the formulation practical for real applications. Without going into all the details, we briefly summarize the two key ideas for effective preprocessing:

- Assume that the clusters defined from bucket B_i will only be made up of stands s_j with $j = i, \ldots, S$. In this way, each cluster will be uniquely represented by the bucket corresponding to its lowest stand index.
- Consider a bucket L_i and time period t. Only define y_s^{it} variables for stands s such that there exists a feasible cluster C containing both s and i. In this way, stands that are too far away from s will not be included in bucket B_i.

Incorporating Additional Spatial Constraints into the ARM

There are many possible extensions to the basic ARM. Such extensions are often fundamentally important to different planning applications. In this section we discuss several such extensions and ways of incorporating them into the three different formulations.

Harvest Costs

Although income from timber sales can be approximated as a summation of the volumes from all harvested stands, costs are not necessarily linear. There are usually fixed costs involved in harvesting an area, such as establishing a workforce, setting up operations, and moving equipment (Weintraub et al. 1999). Other costs include fencing, which depends on the perimeter length of the harvested area. McDill et al. (2002) point out that the cost savings achieved by joint management of adjacent stands can be significant.

In the Cluster Packing approach it is straightforward to consider fixed costs or nonlinearities because costs for each cluster are derived a priori. It is not clear how this issue can be imposed in the other formulations in a simple way.

Average Clearcut Size Constraints

In certain cases, imposing average clearcut size constraints can provide much flexibility. For example, instead of imposing that clusters *never* exceed a size of 40 ha, it could be imposed that on *average* they never do so. Recent voluntary initiatives, such as that of the American Forest and Paper Association (2001), promote the use of such average constraints, as opposed to strict area limitations.

Let $\overline{A_t}$ equal the maximum allowed average clearcut size for time period t. A bound on the average clearcut size can be imposed using the following constraint in the Cluster Packing formulation (see Murray et al. 2004):

$$\sum_{C \in \Omega} (a_{C,t} - \overline{A_t}) x_{C,t} \leq 0 \quad \forall t = 1, \ldots, T. \tag{24}$$

Observe that imposing 24 in the cluster packing formulation still requires that a maximum area be imposed when the feasible cluster variables are defined.

This constraint can also be imposed in the bucket formulation by use of the following constraints:

$$\sum_{s=1}^{S} \sum_{i=1}^{S} a_s y_s^{it} \leq \overline{A_t} \sum_{i=1}^{S} y_i^{it} \quad \forall t = 1, \ldots, T$$
$$y_s^{it} - y_i^{it} \leq 0 \quad \forall i = 1, \ldots, S, \quad \forall t = 1, \ldots, T \tag{25}$$

The validity of this constraint follows immediately from the observation that $\sum_{i=1}^{S} y_i^{it}$ defines a lower bound on the total number of clusters selected for harvesting. Observe that an advantage of the bucket formulation over the cluster packing formulation in terms of this constraint is that a strict maximum area requirement need not be imposed in the bucket formulation. It is not clear how this constraint can be imposed in the path formulation.

Green-Up Constraints

Green-up constraints extend maximum clearcut area constraints to multiple time periods. That is, they impose that once an area is harvested, adjacency conditions should still be enforced for that area during a certain amount of time, which is called a green-up period. The length of this period depends on the species that are replanted and the length of each period. For example, the American Forest and Paper Association (2001) promotes adopting "green-up requirements, under which past clearcut harvest areas must have trees at least 3 years old or 5 ft high . . . before adjacent areas may be clearcut."

We distinguish between two types of green-up constraints, namely dynamic and static green-up constraints. The main difference between the two is that dynamic green-up constraints enforce the adjacency condition at the stand level, whereas static green-up constraints enforce the condition at the cluster level. It is important to note that this is a fundamental difference that has not been made explicit in other articles. For example, the notion of green-up used by Goycoolea et al. (2005) and Constantino et al. (2006) is that of static green-up. However, the notion of green-up used by Gunn and Richards (2005b), Barrett and Giles (2000), and Davis et al. (2001) is that of dynamic green-up.

Consider a green-up period equal to Δ. That is, if a stand is harvested in period t, it will be considered in a clearcut state for all periods in $\{t, \ldots, t + \Delta - 1\}$.

Dynamic green-up constraints limit the combined area of contiguous stands in a clearcut state independent of the moment in which the stands were harvested. Dynamic green-up constraints can be easily implemented by adding prescription variables (see Barrett and Giles 2000, Davis et al. 2001, Gunn and Richards 2005b). Under our assumptions of only one possible treatment, the formulation obtained has two sets of variables. One indicates when a stand is harvested and the other when a stand is in a clearcut state.

The details for implementing this formulation in the ARM model are as follows. We first introduce additional binary variables $z_{s,t}$ for each stand s and each period t and then modify some of the constraints. The idea of these variables is that they should indicate whether stand s is harvested in time t, whereas the variables $y_{s,t}$ and $x_{C,t}$ should indicate whether a stand or cluster is in a clearcut state during period t.

For the Path formulation constraints 2 need to be replaced by

$$y_{s,t} = \sum_{q=t-\Delta+1}^{t} z_{s,q} \quad \forall t = 1, \ldots, T, \quad \forall s = 1, \ldots, S. \tag{26}$$

For the Cluster Packing formulations constraints 10 need to be replaced by

$$\sum_{C \in \Omega(s)} x_{C,t} = \sum_{q=t-\Delta+1}^{t} z_{s,q} \quad \forall t = 1, \ldots, T, \quad \forall s = 1, \ldots, S. \tag{27}$$

For both formulations it is necessary to add constraints,

$$\sum_{t=1}^{T} z_{s,t} \leq 1 \quad \forall s = 1, \ldots, S. \tag{28}$$

In addition, changes need to be made to both models, because the objective function, volume flows, and average ending age constraints should be stated in terms of the z variables as opposed to the y and x variables. For example, objective functions 1 and 9 should be replaced by

$$\max \sum_{t=1}^{T} \sum_{s=1}^{S} p_{s,t} z_{s,t}. \tag{29}$$

Modifying the Bucket formulation is analogous to modifying the Path formulation.

Static green-up constraints (Goycoolea et al. 2005, Constantino et al. 2006) are different in the sense that they mimic the effect of green-up constraints over the unit restriction model. The main difference with the dynamic green-up approach is that in static green-up all contiguous stands in a clearcut state are forced to be harvested in the same time period. A methodological advantage of this approach is that green-up constraints can be easily imposed at the cluster level without the introduction of any additional variables.

Static green-up constraints can be implemented in the Clique-Cluster Packing formulation by extending the clique inequalities to

$$\sum_{C \in \Omega(K)} \sum_{q=t}^{t+\Delta-1} x_{C,q} \leq 1 \quad \forall K \in \Pi, \quad \forall t = 1, \ldots, T-1, \tag{30}$$

in the Path formulation by adding

$$\sum_{u \in N(s)} \sum_{q=t-\Delta+1, q \neq t}^{t+\Delta-1} y_{u,q} \leq M(1 - y_{s,t}) \quad \forall s = 1, \ldots, S, \quad \forall t = 1, \ldots, T \tag{31}$$

where M is some suitable large number and in the Bucket formulation by extending 17b to

$$\sum_{i=1}^{S} \sum_{q=t}^{t+\Delta-1} W_K^{it} \leq 1 \quad \forall K \in \Pi, \quad \forall t = 1, \ldots, T.$$

We now illustrate the difference between the dynamic and static green-up constraints using the following example. Again consider Figure 3 that consists of a narrow forest made up of four stands. Assume that in this example the green-up requirement is of two time periods, that all stands span 10 ha, and that the maximum clearcut size is 20 ha. Thus, if stand 1 is harvested in period 1, stand 2 in period 2, stand 3 in period 3, and stand 4 in period 4, we observe that for $t = 1$ there will be a single clearcut cluster $\{1\}$, for $t = 2$ a clearcut cluster $\{1, 2\}$, for $t = 3$ a clearcut cluster $\{2, 3\}$, for $t = 4$ a clearcut cluster $\{3, 4\}$, and for $t = 5$ a clearcut cluster $\{4\}$. Given that in all time periods the clusters satisfy the maximum clearcut size condition, the solution is feasible for the dynamic green-up constraints. However, this same solution would not be valid for the static green-up constraints.

It is interesting to note that the introduction of such green-up constraints and the choice between the dynamic or static versions can have important consequences in terms of spatial analysis. In fact, many of the specialized Cluster Packing constraints have to be carefully considered. This is because the shapes of clearcut areas will be constantly changing in time when dynamic green-up constraints are used. For example, consider the average clearcut size constraints described earlier. Should the average size be considered in terms of the clearcut clusters as they evolve in time? Or should the average size only be considered in terms of when clusters are actually harvested? How should fixed costs be defined when dynamic green-up constraints are used?

Restricting the Set of Feasible Areas

In certain applications it may be of importance to restrict the actual shapes of harvested clusters. For example long, elongated clusters may be undesirable from an operational point of few, and clusters with holes in the middle may be undesirable in terms of wildlife protection. In some cases it may even be desirable to limit the *minimum* size of a clearcut area (Andalaft et al. 2003). This condition is applied when determination of fixed costs is difficult. In the Cluster Packing formulation all such constraints are easily imposed by discarding clusters having nondesirable shapes or characteristics through a preprocessing phase. It is not clear how this issue can be imposed in the other formulations in a simple way.

Using Alternative Definitions of Adjacency

It has long been recognized that adjacency conditions can be interpreted in a variety of ways. The classic approach is merely to infer adjacency when two stands share a common border or edge (point-touch adjacency). However, adjacency could be defined on the basis of many conceived notions of proximity, both spatial and aspatial (for example, see Walters 1996). There are several reasons that it may be desirable to consider different definitions of adjacency. As has been pointed out before, when point-touch adjacency definitions are used, maximum clearcut size constraints tend to leave managed forests very fragmented, reducing them to small, disconnected patches, with each having a reduced interior area (Gustafson and Crow 1998, Borges and Hoganson 2000, Rebain and McDill 2003). Although some species might prefer early successional stage habitats (which often occur in forest edges), it is well known that these can be very detrimental to many other species.

Part of the problem is that a single definition of adjacency is used in current models to simultaneously model two different aspects of harvest scheduling. On the one hand, it is used to define when a group of stands make up a cluster, and on the other, it is used to define how far apart different clusters should be from each other. Point-touch adjacency metrics seem to be a reasonable measure with which to define clusters because (for operational reasons) one would like the stands that make up a common cluster to be tightly packed together. However, it seems a poor choice for defining cluster incompatibility as one would like clusters to be well separated one from another, possibly by a wide buffer.

An alternative to point-touch definitions of adjacency for use in defining cluster incompatibility consists of defining clusters to be adjacent when they are within a certain distance of each other, say, M meters (distance adjacency). In this way, if we consider a single time period model, it is easy to see that harvested clusters will be separated from one another by a wide buffer and that the diameter of the standing forest will be at least M meters wide at all points, except, possibly, near the edges. If, in addition, it is imposed that clusters should be M meters away from the edge of the forest, then the resulting harvest plans will be such that the standing forest is entirely connected and that the diameter condition will hold throughout. In this way we would obtain a forest in which the unharvested region is connected by corridors of width M meters throughout. By properly combining distance adjacency metrics with green-up constraints, it is possible to ensure that these connectivity conditions extend in time.

Goycoolea et al. (2005) define this variation of the ARM, which considers two simultaneous definitions of adjacency, as the extended area restriction model. As they point out, formulating this model using the cluster packing formulation is trivial. In fact, it is simply a matter of defining the clusters with one definition and using the latter to define the cliques that define the compatibility constraints. It is not clear how this issue can be imposed in the path and bucket formulations.

Computational Tests

Introduction

To compare the performance of the different integer programming approaches, tests were conducted on four forest regions. The goal of the experiments was to assess the following:

1. The size of the different formulations. We focus on the number of columns, rows, and nonzeroes resulting in the formulation of real forest instances. Special attention is paid to how the size changes relative to the maximum clearcut size condition.
2. The strength of the LP relaxation bounds. The proximity of the LP relaxation bound to the value of the best known feasible solution is compared. Special attention is paid to how this proximity changes with the number of time periods considered and with the use of volume flow, average ending age constraints, and green-up constraints.
3. The amount of time required to solve problems with each formulation. The time required to solve each problem to optimality is compared as well as the time required to find feasible solutions having a reasonable gap (1%).

The first data set, El Dorado, corresponds to a United States national forest region in northern California. Sets Shullkel and Lemon Creek correspond to Canadian national forest regions in Nova Scotia and British Columbia. The last set, NBCL5, corresponds to a publicly owned tract of industrial forest region in New Brunswick. Each data set contains information describing the age, timber volume, and area for each stand. All of these data sets, with the exception of Lemon Creek[1], are available on the Internet (Integrated Forest Management Lab 2006). Furthermore, note that a choice of maximum clearcut size of 16.19 ha was chosen instead of 48.56 ha for Shullkel and Lemon Creek. Similarly a choice of 32.37 ha was chosen for NBCL5. These sizes were chosen because these instances have many stands that are very small. Thus, limiting the maximum clearcut size assures that the problems could be formulated in at most 2 GB of RAM. In Table 1 some basic information concerning these instances is summarized.

These forests were selected because their characteristics make them induce relatively hard-to-solve ARM models. For example, El Dorado has a fairly homogeneous age distribution, which usually results in harder ARM models (McDill and Braze 2000, McDill et al. 2002). Computational results should then be considered illustrative of types of forests.

To carry out the tests, two stands were defined as adjacent if they touched in a line. The revenue was assumed to be proportional to the timber volume for each stand. For multiple time period runs a discount rate of 3% was applied to each period. When using volume flow constraints, we defined $Lt = 1 - 15/100$, and $Ut = 1 + 15/100$, for all $t = 1, \ldots, T$. Unless specifically stated otherwise, no green-up constraints were imposed. When we used average ending age constraints, the average ending age was required to be

Table 1. Problem instance information

Instance	Stands	Total area	Minimum stand area	Average stand area	Maximum stand area	Maximum clearcut size
		ha				
El Dorado	1,363	21,147.03	4.05	15.52	47.09	48.56
Shullkel	1,039	4,498.75	0.13	4.33	112.36	16.19
Lemon Creek	6,675	40,814.17	2.84	6.11	97.74	16.19
NBCL5	5,881	60,393.1	0.99	10.27	99.95	32.37

of 40 years. Stands having an area larger than the maximum imposed limit were ignored. Note that the Path formulation strengthening of Crowe et al. (2003) and Gunn and Richards (2005a) as described earlier was not used.

All runs were made on a Pentium IV (Xeon) running at 2.40 GHz and with 2 GB of RAM using Linux. All programs were written in the C++ programming language, and ILOG CPLEX 9.0 was used for all linear and integer programs solutions. Summaries of the runs can be found in Tables 3–6 and in Figure 4. Unless otherwise noted, all runs were made using the default CPLEX settings and a maximum time limit of 10,000 seconds (≈2.8 hours) was imposed. Finally, note that the times reported do not take into account the amount of time taken to build the problem, e.g., the time required to enumerate all clusters and cliques and set up the formulation. However, the enumeration algorithms described for the ARM ran very fast. In fact, completely formulating (e.g., enumerating clusters, cliques, and defining the constraints) each of the problems took less than 1 minute for most combinations of instances and approaches. More specifically, the average formulation times for the Path, Clique Cluster Packing, and Bucket formulations were 28.97, 23.39, and 73.08 seconds, respectively.

Size of the Different Formulations

Our first experiment consisted of formulating our four test cases using the different formulations and comparing the resulting sizes. As a preliminary step it was necessary to enumerate all of the feasible and minimally infeasible clusters, as well as all of the maximal cliques for each forest. In Table 2 we list the number of each of the enumerated objects when the maximum clearcut size areas indicated in Table 1 were used.

As can be seen, the number of clusters (both feasible and minimally infeasible) per stand is roughly 10–30, and the number of maximal cliques is 1–2 per stand. Further, it can be observed that the number of feasible clusters is similar to the number of minimally infeasible clusters in each forest. Thus, it would be expected that the total number of rows and columns should be very similar for the Path and Clique Cluster Packing formulations. This expectation is confirmed in Table 3, in which we describe the number of rows, columns, and nonzero coefficients for each of the three formulations when the specific case in which there are three time periods and both volume and ending age constraints is considered.

For example, consider instance El Dorado. It can be seen that the total number of rows and columns for the cluster formulation is 71,700 and for the Path formulation is 67,344. On the other hand, it can be seen that the Bucket formulation results in much larger problems for these tests. In El Dorado the number of rows and columns adds up to 379,296. It can also be seen that the cluster formulation is by far the densest formulation (in terms of nonzeroes).

Our next experiment consisted of increasing the maximum clearcut size to see how this affects the problem size.

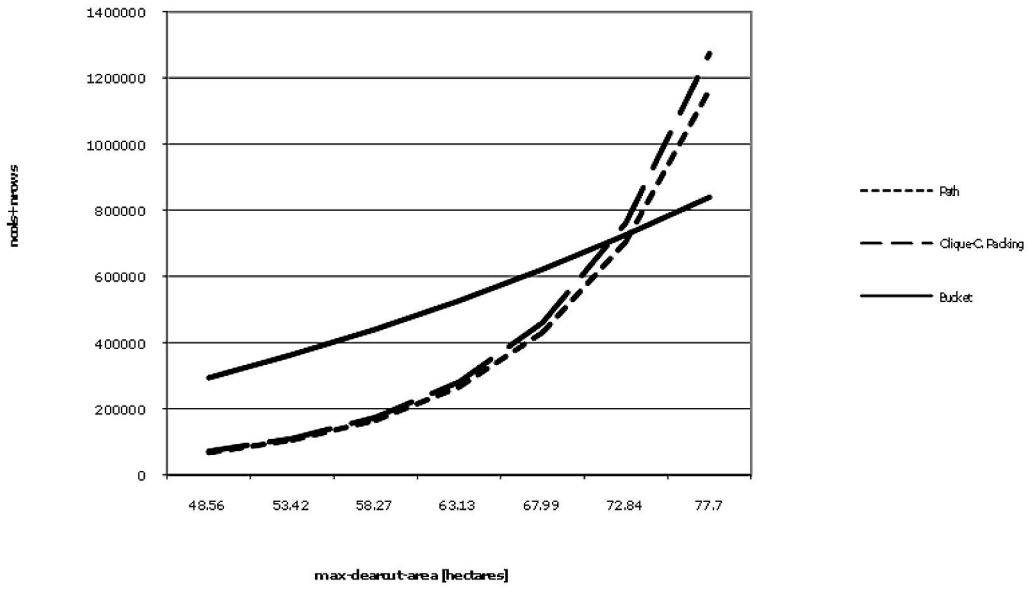

Figure 4. Impact of maximum clearcut size parameter.

Table 2. Number of clusters and cliques

Instance	Stands	Feasible clusters	Minimally infeasible clusters	Maximal cliques
El Dorado	1,363	21,411	20,629	2,033
Shullkel	1,039	29,630	12,889	1,093
Lemon Creek	6,675	87,336	175,743	10,766
NBCL5	5,881	94,912	52,136	5,967

As expected, problem size increased exponentially with both the Path and Clique Cluster Packing formulations. Increasing the size of the maximum clearcut condition by 60% resulted in an increased problem size of roughly 500%. Not surprisingly, problem size increased linearly with the Bucket formulation. That the Bucket formulation grows at all is because the preprocessing becomes less effective as the maximum clearcut size value is increased. This effect can be observed in the case of El Dorado (three time periods, with volume flow and average ending age constraints) in Figure 4. By observing the graph we can see that if the maximum clearcut size increases to 72.84 ha or more, then the Bucket formulation becomes the most compact (in terms of total number of columns and rows) of the three. It is interesting to observe how the average cluster size grows with the maximum clearcut size condition. Table 4 shows that in the specific case of El Dorado, the size of feasible clusters nearly doubles when the average clearcut size condition is increased by 60%.

Strength of the LP Relaxation

Our second experiment consisted of solving the LP relaxation of each test case with 1, 3, and 5 time periods. We also solved for El Dorado with 12 time periods. For each problem we considered up to four variants; one without using volume flow, ending age, or any green-up constraints (labeled as "*T*," indicating the number of time periods considered), one using both volume flow and average ending age constraints ("*T*, volume + age"), and two using volume flow, average ending age, and dynamic green up constraints ("*T*, volume + age, green-up," in which the last parameter indicates the length of the green-up period). For each run we recorded the time it took to solve the LP relaxation (root LP time) and, in addition, the value of the best solution obtained. Then, for each instance we kept the best solution found among all the different runs. For each run we compared the value of the LP relaxation with the value of the best known feasible solution for the corresponding instance (root LP gap). If x is the value of the best known solution for a given instance and r is the value of the LP relaxation obtained using a given formulation, then the corresponding root LP gap would have value $[(r/x) - 1] \times 100$. For example, the best-known feasible solution for El Dorado ($T = 1$) has value $x = 2154060$, and the LP relaxation for the Bucket formulation has value $r = 2249236.2$. Thus, the value of root LP gap is $[(2249236.2/2154060) - 1] \times 100 = 4.4184$. The results are illustrated in Table 5.

Several observations arise. It can be seen that the cluster formulation yields, by far, the tightest root LP gaps. For example, consider the Lemon Creek instance ($T = 3$, volume + age). The root LP gap for the Path formulation is 1.65%, and for the Bucket formulation this value is 1.18%. However, the value for the Clique-Cluster Packing formulation is 0.68%. In general, however, all three formulations perform rather well with the averages gaps (over all test cases) for the Path, Clique-Cluster Packing, and Bucket formulation being 3.13, 0.73, and 2.45%, respectively.

In Theorem 1 it was proved that the LP relaxation gap of the Clique-Cluster Packing formulation could never be less tight than that of the Path formulation. However, no such relationship was established between the Bucket formulation and any other formulation. From Table 5 it can be seen that neither the Path nor Bucket formulation is stronger than the other. In fact, for El Dorado ($T = 5$) it can be seen that the Bucket formulation is tighter than the Path formulation. However, for Shulkell ($T = 1$) it can be seen that the converse holds. On the other hand, the table suggests that it may be true that the Clique-Cluster Packing formulation is always tighter than the Bucket formulation (this is, observed in every instance considered).

The time required to solve the LP relaxation of the Bucket formulation is considerably longer than that required to solve the Path and Clique-Cluster Packing formulations. We attempted to solve the same instances with the barrier method (as opposed to the default CPLEX settings), and although several problems took considerably less time to solve, others took much longer, thus making it unclear which method was the better. It would seem that this difficulty should be further studied.

As the number of time periods increases, the root LP gaps tend to decrease in the Path and Bucket formulations. A clear example is the El Dorado problem with the Path formulation, for which the root LP gap values for $T = 1$, $T = 3$, $T = 5$, and $T = 12$ are 5.81, 5.67, 3.21, and 0.79%. The Clique-Cluster Packing formulation, on the other hand, seems to peak when $T = 3$. For example, consider the Lemon Creek instance, in which the root LP gaps for $T = 1$, $T = 3$, and $T = 5$ are 0.17, 1.23, and 0.61%, respectively.

Adding the volume and ending age constraints decreases the root LP gap values in all formulations. For example, in the Path formulation, observe that El Dorado ($T = 5$) has a root LP gap of 3.21% and that El Dorado ($T = 5$, volume + age) has a root LP gap of 0.25%. However, adding the extended green-up constraints (green-up = 2, green-up = 3) increases the root LP gaps in all formulations. In El Dorado ($T = 3$, volume + age) it can be seen that the Path formulation has a root LP gap of 0.74%. However, when green-up constraints are introduced, this value goes up to 7.16%.

The impact of imposing green-up constraints seems to be much less severe in the Clique-Cluster Packing formulation than in the Path and Bucket formulations. For example, consider the Shulkell ($T = 5$, volume + age) instance. The root LP gaps for the Path, Clique-Cluster Packing, and Bucket formulations are 0.34, 0.02, and 0.25%, respectively. However, with introduction of green-up = 3 constraints, these values increase to 7.45, 0.15, and 6.78%, respectively.

Table 3. Size for three periods

	Path			Clique-Cluster Packing			Bucket		
Instance	Columns	Rows	Nonzero	Columns	Rows	Nonzero	Columns	Rows	Nonzero
El Dorado	4,089	63,255	297,952	64,233	7,467	1,280,747	114,309	264,987	760,437
Shullkel	3,117	39,711	233,113	88,890	4,323	1,822,338	60,264	135,111	418,464
L.Creek	20,025	533,909	2,359,277	262,008	38,978	5,234,423	684,171	1,579,148	4,435,775
NBCL5	17,643	162,294	915,292	284,736	23,787	5,352,639	294,981	633,042	1,942,489

Table 4. Impact of maximum clear-cut size parameter (El Dorado)

Maximum clearcut size (ha)	Average cluster size	Maximum cluster size (no. stands)
48.56	3.57926	7
53.42	4.01706	8
58.27	4.46242	8
63.13	4.91272	9
67.99	5.36555	10
72.84	5.81671	10
77.7	6.26803	11

Solving the Integer Programming Problem

Our third experiment consisted of solving the integer programming problem (with a time limit of 10,000 seconds) for each of the problem variants considered in the previous section. The main tool used for measuring performance was the gap. Given an incomplete run of a branch and bound algorithm, let zu be the value of the current upper bound and let zl be the value of the best-known feasible solution. We compute gap as $[(zu/zl) - 1] \times 100$. For each optimization run we recorded the time at which the first feasible solution within a provable 1% of optimality was obtained ("First 1% solution", in seconds) and the best provable gap obtained after the 10,000-second time limit expired ("Final gap/opt time"). If a problem was solved to optimality within the time limit, the number of seconds required to complete the solution is indicated in parentheses. If no feasible solution was found or no feasible 1% solution was found, we indicate this with a dash in the corresponding column. The results obtained are summarized in Table 6.

We made the following observations. The Clique-Cluster Packing formulation significantly outperformed the Path and Bucket formulations in problems without volume, ending age, and green-up constraints, having managed to solve most of these to optimality. The other formulations had a very difficult time even finding solutions within 1% for these problems. For example, for El Dorado ($T = 1$), the Clique-Cluster Packing formulation found a 1% solution in 2.93 seconds and solved the problem to optimality in 4.3 seconds. However, after 10,000 seconds the Path formulation had only managed to establish a solution within 2.07% of optimality and the Bucket formulation had only managed a solution within 2.58% of optimality.

The performance of the Clique-Cluster Packing formulation relative to that of the Path formulation is most significant when the difference between the LP relaxation bounds is large (see El Dorado, $T = 1$). On the other hand, when the difference between the LP relaxation bounds is small, the performance of both methods is more similar, with the Path formulation sometimes performing better (see Shulkell, $T = 5$, volume + age).

Problems with green-up constraints proved considerably harder to solve for all of the formulations. For the Lemon Creek problem we were simply unable to obtain feasible solutions with any of the formulations.

In problems with no green-up constraints, but with volume and age constraints, the relative performance of the Path and Clique-Cluster Packing formulations seems to depend on the number of time periods. In fact, for $T = 1$ instances the Clique-Cluster Packing formulation seems to perform better, yet for $T = 5$ instances (and the $T = 12$ instance) the Path formulation seems to be the top performer.

In most of the runs, the Bucket formulation did not perform as well as the other two formulations. In fact, in more than half of the runs it was unable to even find a feasible solution. This was very surprising; especially when these results are compared to those reported in Constantino et al. (2006), in which very different results are presented for the specific case of El Dorado. This discrepancy led us to contact the authors of Constantino et al. (2006). We found that our implementations differed in many aspects. First, we had used different parameters for the discount rates, volume flows, and average ending age constraints. More importantly, however, we found that the LP formulations we were ultimately solving were very different in terms of constraints and number of variables. The reason for this was that the preprocessing routine was dependent on the way stands in the forest were numbered. This finding suggests that permuting the ordering of the stands could lead to alternative formulation preprocessing schemes.

Final Remarks

In this article we have reviewed the three major integer programming formulations that have been proposed for the ARM and have discussed how different spatial concerns can be addressed using each approach. In addition, through computational tests, we have shown how these methodologies perform on a set of real problems of small to medium size. It is important to keep in mind that because of the limited sample size of our experiments, the results should be regarded as being illustrative rather than as statistically significant in their conclusions.

All of these formulations, which have been developed during the last decade, constitute an important first step in being able to solve the ARM to optimality. However, it seems clear from our analysis that several important issues need to be resolved before these methodologies can be

Table 5. Strength of linear programming relaxations and time required to solve

No. time periods, and extensions of formulation	Path		Clique-Cluster Packing		Bucket	
	Root lp gap (%)	Root lp time (s)	Root lp gap (%)	Root lp time (s)	Root lp gap (%)	Root lp time (s)
El Dorado						
$T = 1$	5.81	0.84	0.16	0.57	4.42	802.71
$T = 3$	5.67	17.9	0.57	22.86	3.51	10,000+
$T = 3$, volume + age	0.74	12.11	0.08	8.35	0.47	444.21
$T = 3$, volume + age + green-up = 2	7.16	12.22	0.38	6.97	5.11	10,000+
$T = 5$	3.21	92.91	0.44	31.77	2.01	8,916.97
$T = 5$, volume + age	0.25	13.05	0.05	13.19	0.13	394.81
$T = 5$, volume + age + green-up = 2	3.37	247.7	0.96	215.79	2.63	10,000+
$T = 5$, volume + age + green-up = 3	8.31	538.1	0.52	500.92	5.16	10,000+
$T = 12$	0.79	41.38	0.08	33.68	0.53	1,541.38
$T = 12$, volume + age	0.47	16.12	0.46	31.61	0.48	706
$T = 12$, volume + age + green-up = 2	5.89	621.43	5.74	120.85	5.86	10,000+
$T = 12$, volume + age + green-up = 3	12.44	1,125.19	10.83	2259.42	11.98	10,000+
Shulkell						
$T = 1$	2.16	0.29	0.02	0.37	4.00	33.5
$T = 3$	1.06	3.54	0.06	2.97	0.77	107.55
$T = 3$, volume + age	0.38	7.06	0.03	9.85	0.55	148.7
$T = 3$, volume + age + green-up = 2	7.75	22.18	0.13	27.72	7.14	5,644.02
$T = 5$	0.34	14.71	0.02	4.49	0.25	82.47
$T = 5$, volume + age	0.03	12.67	0.01	14.08	0.08	89.8
$T = 5$, volume + age + green-up = 2	1.16	45.41	0.37	109.2	1.29	4,138.67
$T = 5$, volume + age + green-up = 3	7.45	155.86	0.15	103.1	6.78	10,000+
Lemon Creek						
$T = 1$	14.16	24.26	0.17	5.32	5.71	10,000+
$T = 3$	7.24	52.24	1.23	29.46	3.72	10,000+
$T = 3$, volume + age	1.65	381.12	0.68	2,303.17	1.18	10,000+
$T = 5$	3.75	11,012.59	0.61	1,759.53	2.51	10,000+
$T = 5$, volume + age	0.25	2,021.5	0.08	1,555.07	0.19	10,000+
NBCL						
$T = 1$	0.78	1.58	0.01	1.5	2.49	286.34
$T = 3$	0.21	13.71	0.01	10.21	0.42	64.6
$T = 3$, volume + age	0.04	120.4	0.01	31	0.08	267.5
$T = 3$, volume + age + green-up = 2	0.34	63.05	0.02	54.07	0.75	1,285.87
$T = 5$	0.07	48.29	0.00	14.43	0.18	176.59
$T = 5$, volume + age	0.01	267.42	0.01	66.82	0.02	504.05
$T = 5$, volume + age + green-up = 2	0.09	1122.17	0.01	32.73	0.12	496.51
$T = 5$, volume + age + green-up = 3	0.24	1,936.1	0.03	108.8	0.43	1,368.18

extended to larger problems. Some suggested research directions follow.

In Table 5 it can be seen that all three formulations provide very tight LP bounds. However, solving these formulations to optimality via branch and bound is painfully slow (see Table 6). Although much research has been conducted in proposing customized heuristics for these problems, no efforts have been pursued that combine such heuristics (lower bounds) and LP to validate their quality (upper bounds). The tight bounds obtained in the section on strength of the LP relaxation suggest that this may be a fruitful procedure in practice. After all, solving the LP relaxation of these problems is very fast, and it is natural to assume that the LP relaxation of much larger problems can be solved. If the bound is as tight on these problems and good customized heuristics are developed, there may be no need to do branch-and-bound to obtain provably good solutions.

As shown in the section on size of the different formulations, all three integer programming formulations result in extremely large problems. Larger integer programming problems are hard to solve, primarily because they require enormous amounts of memory to store. With current methodologies, problems with more than 30,000 stands or problems in which feasible clusters average more than 10 stands each seem largely intractable. It would seem that to solve such problems decomposition-based integer programming methodologies, such as cutting plane or column generation algorithms, are necessary.

As shown in the aforementioned sections, it would seem that further research in computationally effective ways of implementing the Bucket formulation are necessary. This approach is very promising as an alternative way of dealing with problems in which stands are very small relative to the maximum allowed clearcut area. So far, our computational experiments have been largely disappointing. It has been noted that one way of dealing with this issue is through more advanced preprocessing schemes. This conclusion follows from the observation that the CPLEX preprocessor eliminates a large amount of variables and constraints even after the preprocessing steps described in the section on the Bucket formulation.

Table 6. Solving the integer programming problem

	Path		Clique-Cluster Packing		Bucket	
	First 1% solution	Final gap (opt time)	First 1% solution	Final gap (opt time)	First 1% solution	Final gap (opt time)
El Dorado						
$T = 1$	—	2.07%	2.93	(4.3 s)	—	2.58%
$T = 3$	—	3.62%	171.94	(5,417.4 s)	—	15.26%
$T = 3$, volume + age	27.19	0.13%	73.42	0.01%	1,433.18	0.44%
$T = 3$, volume + age + green-up = 2	—	7.52%	173.48	(1,732.6)	—	—
$T = 5$	—	1.71%	269.99	0.09%	—	11.38%
$T = 5$, volume + age	36.02	0.03%	101.72	0.07%	841.14	0.23%
$T = 5$, volume + age + green-up = 2	—	2.77%	2,379.93	0.80%	—	—
$T = 5$, volume + age + green-up = 3	—	8.68%	1,817.17	0.19%	—	—
$T = 12$	73.61	0.28%	109.48	(192.31 s)	3,533.57	0.79%
$T = 12$, volume + age	1,684.58	0.45%	—	1.98%	—	2.69%
$T = 12$, volume + age + green-up = 2	—	5.79%	—	5.72%	—	—
$T = 12$, volume + age + green-up = 3	—	—	—	10.69%	—	—
Shulkell						
$T = 1$	19.21	(6,315.2 s)	0.9	(1.0 s)	—	1.80%
$T = 3$	37.04	0.36%	12.82	(22.8 s)	1,376.34	0.50%
$T = 3$, volume + age	120.03	0.11%	74.45	(497.3 s)	3,960.02	0.47%
$T = 3$, volume + age + green-up = 2	—	4.92%	38.44	(1,190.9 s)	—	—
$T = 5$	16.26	0.06%	17.53	(28.2 s)	148.93	0.12%
$T = 5$, volume + age	17.12	(786.6 s)	31	(2,878.1 s)	223.39	0.19%
$T = 5$, volume + age + green-up = 2	—	1.01%	655.44	0.27%	—	—
$T = 5$, volume + age + green-up = 3	—	6.04%	340.99	0.07%	—	—
Lemon Creek						
$T = 1$	—	13.50%	17.75	(35.4 s)	—	—
$T = 3$	—	7.35%	9,850	0.91%	—	—
$T = 3$, volume + age	—	3.06%	7,143.51	0.62%	—	—
$T = 5$	—	5.19%	8,544.3	0.41%	—	—
$T = 5$, volume + age	8,831.06	0.52%	1,716.68	0.06%	—	—
NBCL						
$T = 1$	11.9	0.22%	4.15	(4.4 s)	—	1.14%
$T = 3$	20.41	0.04%	30.56	(65.9 s)	286.89	0.18%
$T = 3$, volume + age	132.78	(235.7 s)	55.21	(250.3 s)	1,334.25	0.05%
$T = 3$, volume + age + green-up = 2	260.66	0.14%	208.19	(474 s)	—	—
$T = 5$	53.6	0.02%	43.25	(71.8 s)	216.08	0.07%
$T = 5$, volume + age	340.68	(3,948.79 s)	104.22	(1,103.7 s)	999.91	(3,302.3 s)
$T = 5$, volume + age + green-up = 2	1,341.34	0.06%	293.78	0.01%	2,067.39	0.48%
$T = 5$, volume + age + green-up = 3	2,086.22	0.16%	703.52	0.02%	—	—

—, no feasible solution or no feasible 1% solution was found.

Some work in this direction is currently being pursued by Mills and McDill (2006).

As observed in the sections on strength of the LP relaxation and size of the different formulations, incorporating green-up constraints has a very adverse effect on the LP relaxation gaps. This, in turn, makes the integer programming problems much harder to solve. Deriving new classes of strong valid inequalities might be an important direction in being able to better solve this variant of the ARM.

Incorporating old-growth stands into optimization-based forest harvest planning is a key issue that to date has not been successfully tackled. The Cluster-based approach of Martins et al. (2005) and the path-based approach of Rebain and McDill (2003) are both still very exploratory with only small problem instances being solvable to date. Both studies raise very relevant questions, provide insightful analysis of obtained solutions, and suggest a need for further studies. Interesting methodological approaches to these problems are also that of Ohman (2000) and Wei and Hoganson (2006) It is not clear how these methodologies can be made to work better, or whether instead there is another way of adapting current ARM formulations to consider this type of constraint.

Endnotes

[1] The original Lemon Creek data contained so many stands that none of the one-period formulations could be generated even when using a computer with 12 GB of RAM. For this reason the data set was modified by aggregating stands together until no stand had an area less than 2.83 ha. This modified instance is what will be denoted as Lemon Creek from here on.

Literature Cited

American Forest and Paper Association. 2001. *Sustainable forestry initiative standard.* Available online at www.afandpa.org/; last accessed on Feb. 24, 2007.

Andalaft, N., P. Andalaft, M. Guignard, A. Madgenzo, A. Wainer, and A. Weintraub. 2003. A problem of forest harvesting and road building solved through model strengthening and lagrangean relaxation. *Oper. Res.* 51(4):613–628.

Barahona, F., R. Epstein, and A. Weintraub. 1992. Habitat dispersion in forest planning and the stable set problem. *Oper. Res.* 40(1):S14–S20.

BARRETT, T., AND J. GILLESS. 2000. Even-aged restrictions with sub-graph adjacency. *Ann. Oper. Res.* 95:159–175.

BARRETT, T., J. GILLESS, AND L. DAVIS. 1998. Economic and fragmentation effects of clearcut restrictions. *For. Sci.* 44(4):569–577.

BORGES, J.G., AND H.M. HOGANSON. 2000. Structuring a landscape by forestland classification and harvest scheduling spatial constraints. *For. Ecol. Manag.* 130:269–275.

BOSTON, K., AND P. BETTINGER. 2002. Combining tabu search and genetic algorithms heuristic techniques to solve spatial harvest scheduling problems. *For. Sci.* 48:35–46.

BUONGIORNO, J., AND J.K. GILLESS. 2003. *Decision methods for forest resource management.* Elsevier Science, San Diego, CA.

CARO, F., M. CONSTANTINO, I. MARTINS, AND A. WEINTRAUB. 2003. A 2-Opt Tabu search procedure for the multi-period forest harvesting problem with adjacency, green-up, old growth and even flow constraints. *For. Sci.* 49(5):738–751.

CLARK, M.M., R.D. MELLER, AND T.P. MCDONALD. 2000. A three-stage heuristic for harvest scheduling with access road network development. *For. Sci.* 46:204–218.

CONSTANTINO, M., I. MARTINS, AND J. BORGES. 2006. A new mixed integer programming model for harvest scheduling subject to maximum area restrictions. *Oper. Res.* 56(3):542–551.

CORMEN, T.H., C.E. LEISERSON, AND R.L. RIVEST. 1990. *Introduction to algorithms.* MIT Press, Cambridge, MA.

CROWE, K., J. NELSON, AND M. BOYLAND. 2003. Solving the area-restricted harvest scheduling model using the branch and bound algorithm. *Can. J. For. Res.* 33(9):1804–1814.

DAVIS, L.S., K.N. JOHNSON, T.E. HOWARD, AND P.S. BETTINGER. 2001. *Forest management*, 4th ed. McGraw-Hill, New York, NY.

DIESTEL, R. 2006. *Graph theory.* Springer-Verlag, New York, NY.

INTEGRATED FOREST MANAGEMENT LAB. 2006. Forest Management Optimization Site. Available online at www.unbf.ca/fmos/; last accessed Feb. 24, 2007.

GOYCOOLEA, M., A. MURRAY, F. BARAHONA, R. EPSTEIN, AND A. WEINTRAUB. 2005. Harvest scheduling subject to maximum area restrictions: Exploring exact approaches. *Oper. Res.* 53:490–500.

GUNN, E. A. AND E.W. RICHARDS. 2005a. Solving the adjacency problem with stand-centered constraints. *Can. J. For. Res.* 35(4):832–842.

GUNN, E. A., AND E.W. RICHARDS.2005b. *Construction of stand-centered adjacency constraints and properties.* Technical report.

GUSTAFSON, E.J., AND T.R. CROW. 1998. Simulating spatial and temporal context of forest management using hypothetical landscapes. *Environ. Manag.* 22(5):777–787.

HOGANSON, H.M., AND BORGES, J.G. 1998. Using dynamic programming and overlapping subproblems to address adjacency in large harvest scheduling problems. *For. Sci.* 44:526–538.

HOKANS, R.H. 1983. Evaluating spatial feasibility of harvest schedules with simulated stand-selection decisions. *J. For.* 81:601–603, 613.

LOCKWOOD, C., AND T. MOORE. 1993. Harvest scheduling with spatial constraints: a simulated annealing approach. *Can. J. For. Res.* 23:468–478.

MARTINS, I., M. CONSTANTINO, AND J. BORGES. 2005. A column generation approach for solving a non-temporal forest harvest model with spatial structure constraints. *Eur. J. Oper. Res.* 161(2):478–498.

MCDILL, M.E., AND J. BRAZE. 2000. Comparing adjacency constraint formulations for randomly generated forest planning problems with four age class distributions. *For. Sci.* 47(3):403–418.

MCDILL, M.E., S. REBAIN, AND J. BRAZE. 2002. Harvest scheduling with area-based adjacency constraints. *For. Sci.* 48(4):631–642.

MILLS, S., AND M.E. MCDILL. 2006. Incorporating deer exclusion fence considerations into area-restricted harvest schedule models, in *Proc. of 12th Symposium for Systems Analysis in Forest Resources,* Burlington, VT.

MURRAY, A.T. 1999. Spatial restrictions in harvest scheduling. *For. Sci.* 45(1):1–8.

MURRAY, A.T., AND R.L. CHURCH. 1995. Heuristic solution approaches to operational forest planning problems. *OR Spekt.* 17:193–203.

MURRAY, A.T., AND R.L. CHURCH 1996. Analyzing cliques for imposing adjacency restrictions in forest models. *For. Sci.* 42:166–175.

MURRAY A.T., AND R.L. CHURCH. 1997. Facets for node packing. *Eur. J. Oper. Res.* 101:598–608.

MURRAY, A.T., M. GOYCOOLEA, AND A. WEINTRAUB. 2004. Incorporating average and maximum area restrictions in harvest scheduling models. *Can. J. For. Res.* 34:456–464.

MURRAY, A.T., AND A. WEINTRAUB. 2002. Scale and unit specification influences in harvest scheduling with maximum area restrictions. *For. Sci.* 48(4):779–788.

NELSON, J., AND J.D. BRODIE. 1990. Comparison of a random search algorithm and mixed integer programming for solving area-based forest plans. *Can. J. For. Res.* 20:934–942.

NEMHAUSER, G.L., AND L.A. WOLSEY. 1988. *Integer and combinatorial optimization.* John Wiley and Sons, New York, NY.

O'HARE, A., B.H. FAALAND, AND B.B. BARE. 1989. Spatially constrained timber harvest scheduling. *Can. J. For. Res.* 19:715–724.

OHMAN, K. 2000. Creating continuous areas of old forest in long-term forest planning. *Can. J. For. Res.* 30:1817–1823.

RICHARDS, E.W., AND E.A. GUNN. 2000. A model and tabu search method to optimize stand harvest and road construction schedules. *For. Sci.* 46:188–203.

REBAIN, S., AND M.E. MCDILL. 2003. A mixed-integer formulation of the minimum patch size problem. *For. Sci.* 49(4):608–618.

THOMPSON, E.F., B.G. HALTERMAN, T.S. LYON, AND R.L. MILLER. 1973. Integrating timber and wildlife management planning. *For. Chron.* 47:247–250.

TÓTH, S.F. 2005. Modeling timber and non-timber tradeoffs in spatially explicit forest planning. PhD thesis. The Pennsylvania State University, Departments of Operations Research and Forest Resources.

VIELMA, J.P., A.T. MURRAY, D. RYAN, AND A. WEINTRAUB. 2007. Improving computational capabilities for addressing volume constraints in forest harvest scheduling problems. *Eur. J. Oper. Res.* 176:1246–1264.

WALTERS, K.R. 1996. Defining adjacency and proximity of forest stands for harvest blocking. In *Proc. from GIS 1996.* Vancouver, BC, Canada. Available online at www.remsoft.com/docs/library/gis96–1.pdf; last accessed Feb. 28, 2008.

WARE, G.O., AND J.L. CLUTTER. 1971. A mathematical programming system for the management of industrial forests, *For. Sci.* 17:428–445.

WEI, Y., AND H. HOGANSON. 2006. Spatial information for scheduling core area production in forest planning. *Can. J. For. Res.* 36:23–33.

WEINTRAUB, A., F. BARAHONA, AND R. EPSTEIN. 1994. A column generation algorithm for solving general forest planning problems with adjacency constraints. *For. Sci.* 40:142–161.

WEINTRAUB, A., R. EPSTEIN, P. CHEVALIER, AND J. GABARRÓ. 1999. A system for short term harvesting. *Eur. J. Oper. Res.* 119:427–439.

Appendix I: Enumerating Sets in a Forest Graph

To successfully implement any of the integer programming formulations described in this article, it is critical that an efficient algorithm for enumerating sets in a graph be used. For the Path formulation, for example, it is necessary to enumerate all minimally infeasible clusters to define constraints 3. For the Clique-Cluster Packing formulation, it is necessary to enumerate all feasible clusters to define the variables and all maximal cliques to define constraints 32c. Finally, for the strengthened Bucket formulation, it is necessary to enumerate all maximal cliques to define the clique variables W and constraints 17b, 18b, and 23b.

In Algorithm 1, we describe an algorithm for enumerating all minimally infeasible clusters that can be defined from a forest. This algorithm works recursively, starting from clusters composed of a single stand, and growing them one stand at a time until the maximum area is met. Note that this algorithm is, in its original form, very similar to the one used by Rebain and McDill 2003.

More specifically, the algorithm starts by defining all feasible clusters composed of a single stand and putting them in a set called $S[1]$ (see lines 02–04). Then, a recursion is initiated (line 05). At the beginning of the kth iteration, the set $S[k-1]$ will contain all feasible clusters composed of exactly $(k-1)$ stands. From these, the set $S[k]$ is built by adding, to each cluster in $S[k-1]$, all stands that are contiguous to it and that do not make it grow beyond the required limit (see lines 09–19). When adding a stand results in an infeasible cluster, we test whether this cluster is minimally infeasible, and if so, store it (lines 15–17). The recursion ends when we can no longer grow any clusters and remain feasible.

Algorithm I

```
01. begin
02.   for v ∈ V do
03.     S[1]: = S[1] ∪ {v};
04.   end for.
05.   for k = 1 to ∞ do
06.     if S[k] is empty then
07.       terminate;
08.     end if.
09.     for C ∈ S[k] do
10.       for v ∈ N(C) do
11.         D:= C ∪ {v};
12.         if D is feasible and D ∉ S[k + 1] then
13.           S[k + 1]:= S[k + 1] ∪ {D};
14.         end if.
15.         else D is minimally infeasible and D ∉ Λ+ then
16.           Λ+:= Λ+ ∪ {D};
17.         end else.
18.       end for.
19.     end for.
20.   end for.
21. end.
```

For this algorithm to run fast, special attention must be paid in steps 12 and 15 to check whether $D \notin S[k+1]$ or whether $D \notin \Lambda^+$. If simple enumeration and comparison are used, then the algorithm may run prohibitively slow. One way of speeding up this running is use of hash tables or binary trees (see Cormen et al. 1990) to check for repetitions. Most modern programming languages (such as C++ and Python) include special data structures (such as "sets" in C++) that automatically check for repetitions. In our computational tests we found that using the "set" class to define the sets $S[k]$, rather than brute-force comparison, for handling repetitions allowed us to reduce the running time of this algorithm from many hours to a few seconds in most instances.

Note that it is easy to modify Algorithm 1 to generate all possible cliques. In fact, it is simply a matter of deleting lines 12–17 and replacing these with the following instructions:

```
12b. if D is a clique then
13b.   Mark C as "nonmaximal";
14b.   if D ∉ S[k + 1] then
15b.     S[k + 1]:= S[k + 1] ∪ {D};
16b.   end if.
17b. end if.
```

After completing a run, each of the sets $S[k]$ will contain cliques of size k. Note that not all of these will be maximal. However, if a clique is not maximal, it will have been marked in step 13b; thus, they can easily be identified.

Appendix II: Comparing Strength of the Path and Cluster Packing Formulations

In this section, we prove the following theorem and corollary.

THEOREM 1. *Let x be a non-negative vector in $R^{|\Omega| \times T}$. Define $y_{s,t} = \sum_{C \in \Omega(s)} x_{C,t}$ for each stand s and each $t = 1, \ldots, T$. If x satisfies constraints 10, 11b, 12, 13, and 14b (that is, if x is in the LP relaxation of the Edge Cluster Packing formulation), then y satisfies constraints 2, 3, 4, 5, and 6b (that is, y is in the LP relaxation of the Path formulation). Further, we have that*

$$\sum_{t=1}^{T} \sum_{C \in \Omega} p_{C,t}\, x_{C,t} = \sum_{t=1}^{T} \sum_{s=1}^{S} p_{s,t}\, y_{s,t}.$$

COROLLARY. The Cluster Packing formulation is tighter than the Path formulation. That is, the value obtained by solving the LP relaxation of the Edge Cluster Packing formulation can never be strictly greater than that obtained from the Path formulation.

PROOF OF THE COROLLARY. Given any solution x valid for the LP relaxation of the Edge Cluster Packing formulation, by defining $y_{s,t} = \sum_{C \in \Omega(s)} x_{C,t}$, we obtain a solution y valid for the LP relaxation of the Path formulation that has the same objective function value. Thus, the optimal value of the LP relaxation to the Edge Cluster Packing formulation must have a smaller or equal value than that of its Path counterpart, as it is optimizing over a subset of possible objective values.

PROOF OF THE THEOREM. To provide a compact proof we need to use the language of graph theory (Diestel 2006). We will work in the graph of stands with vertex set $V = 1, \ldots, S$ and edge set E given by all pairs of adjacent stands.

For every set of stands U, let $E(U)$ denote the set of all edges in E having both end points in U. Similarly, for every set of edges $F \subseteq E$, let $V(F)$ denote the set of all nodes in V that are an end point to some edge in F.

Let $L \subseteq E$ define a *tree* in G, if L is connected in G, and L contains no cycles. Note that a tree L always satisfies $|L| = |V(L)| - 1$. A well-known result is as follows: If a set $U \subseteq V$ is connected in G, then there exists a tree $L \subseteq E(U)$, such that $V(L) = U$.

Recall that Ω represents the set of all feasible clusters, and Λ represents the set of all infeasible clusters. Further, recall that for a set $U \subseteq V$ we have that $\Omega(U)$ represents the set of all feasible clusters intersecting U. That is, $\Omega(U) = \{C \in \Omega : C \cap U \neq \emptyset\}$.

Let y be a non-negative vector in $R^{|V|}$, and let x be a non-negative vector in $R^{|\Omega|}$.

LEMMA 1. *Assume that for all $v \in V$, the vectors x and y satisfy the following identity: $y_v = \sum_{C \in \Omega(s)} x_C$. For each $v \in V$ consider a scalar α_v and for each $C \in \Omega$, assume that $\alpha_c = \sum_{v \in C} \alpha_v$. Consider a set $U \subseteq V$. Then,*

$$\sum_{C \in \Omega(U)} \alpha_C x_C = \sum_{v \in U} \alpha_v y_v.$$

PROOF OF LEMMA 1.

$$\sum_{v \in U} \alpha_v y_v = \sum_{v \in U} \left(\alpha_v \left(\sum_{C \in \Omega(v)} x_C \right) \right)$$

$$= \sum_{C \in \Omega(U)} \left(x_C \left(\sum_{v \in C} \alpha_v \right) \right) = \sum_{C \in \Omega(U)} \alpha_C x_C. \qquad \blacksquare$$

LEMMA 2. *Assume that for all $v \in V$, the vectors x and y satisfy the following identity*

$$y_v = \sum_{C \in \Omega(v)} x_C$$

$$\textit{If} \quad \sum_{C \in \Omega(\{uv\})} x_C \leq 1 \quad \forall \{u, v\} \in E$$

$$\textit{then} \quad \sum_{v \in U} y_v \leq |U| - 1 \quad \forall U \in \Lambda.$$

PROOF OF LEMMA 2. Consider an infeasible cluster $U \in \Lambda$:

$$\sum_{v \in U} y_v = \sum_{v \in U} \left(\sum_{\{C : v \in V\}} x_C \right) = \sum_{C \in \Omega} |C \cap U| x_C.$$

In contrast, given that U is connected, there exists a tree $L \subseteq E(U)$, such that $V(L) = U$. Adding constraints 11b over the edges in L, one obtains

$$\sum_{\{uv\} \in L} \left(\sum_{C \in \Omega(\{uv\})} x_C \right) \leq |L|,$$

which is equivalent to

$$\sum_{C \in \Omega} x_C |\{u, v\} \in L : u \in C \text{ or } v \in C\}| \leq |L| = |U| - 1.$$

Thus, it suffices to prove that for all feasible clusters $C \in \Omega$ we have

$$|\{u, v\} \in L : u \in C \text{ or } v \in C\}| \geq |C \cap U|.$$

However, observe that $\{\{u, v\} \in L : u \in C \text{ or } v \in C\}\}$ can be partitioned into a disjoint family of trees $L_1, \ldots, L_k$ having the following properties:

- There is no edge in L joining L_i and L_j for all $i \neq j$.
- Defining $C_i = V(L_i) \cap C$, then, $C \cap U$ can be partitioned into the sets C_i.
- For each of the trees L_i, there exists at least one edge with an end point in C_i and the other end point in $U - C_i$ (This is because U is connected, and because $U - C_i$ is nonempty). This implies that $|V(L_i)| \geq |C_i| + 1$.

Putting everything together, it is easy to see that

$$|\{u, v\} \in L : u \in C \text{ or } v \in C\}|$$

$$= \sum_{i=1}^{k} |L_i| = \sum_{i=1}^{k} (|V(L_i)| - 1)$$

$$\geq \sum_{i=1}^{k} |C_i| = |C \cap U|. \qquad \blacksquare$$

PROOF. From Lemma 1 it is easy to see that because 10, 12, and 13 hold, then 2, 4, and 5 hold. From Lemma 2 it is easy to see that because 11b holds, then 3 holds.