AN ABSTRACT OF THE THESIS OF

Juan M. Torres-Rojo for the degree of Doctor of Philosophy in Forest Management presented on November 8, 1989.

Title: The Use of Relaxation to Solve Harvest Scheduling Problems with Flow, Wildlife Habitat and Adjacency Constraints.

Abstract approved: Redacted for Privacy

J. Douglas Brodie

Lagrangean relaxation is presented as a solution technique to solve flow constrained area-based harvest scheduling problems. The best set of multipliers in the Lagrangean approach is obtained through the subgradient method. Guidelines to set some parameters to compute the step size in the subgradient algorithm are provided. An additional procedure to improve the multipliers obtained through the subgradient algorithm is provided.

The area-based harvest scheduling problem with adjacency constraints is approached by reducing the number of these constraints required to specify the adjacency relations among harvest units. A heuristic procedure is proposed to perform this reduction. Such a procedure is based on computing one adjacency constraint per harvest unit. Additional reductions are possible by eliminating the

harvest units whose adjacency relations are described by surrounding areas.

By using surrogate relaxation the set of adjacency constraints is reduced to one constraint. Combining Lagrangean and surrogate relaxation the area-based harvest scheduling problem with adjacency constraints can be further reduced, so the relaxed problem becomes easier to solve than the original problem. The relaxation approach is used to solve the habitat dispersion problem. Simulated examples show that simultaneously optimizing flow, wildlife and adjacency constraints within an area-based approach will be costlier than previous continuous models have led us to believe.

The Use of Relaxation to Solve Harvest Scheduling Problems with Flow, Wildlife Habitat, and Adjacency Constraints

by

Juan M. Torres Rojo

A THESIS

submitted to

Oregon State University

in partial fulfillment of

the requirements for

the degree of

Doctor of Philosophy

Completed November 8, 1989

Commencement June, 1990

APPROVED

Redacted for Privacy

Professor of Forest Resources, in charge of major

Redacted for Privacy

Head of Departement of Forest Resources

Redacted for Privacy

Dean of Graduate School

Date thesis is presented November 8, 1989

Juan Manuel Torres-Rojo

T O

Carmen, Ericka, Martin,

Rodolfo y Soledad

ACKNOWLEDGMENTS

I want to express my gratitude to Dr. J. Douglas Brodie, my major professor for his help, encouragement and advice in accomplishing this work. To Dr. John Sessions who always supported this research and provided important advice during its development. I also would like to thank Dr. Jeffrey L. Arthur and Dr. Richard Adams for serving in my committee.

Finally I want to thank the "Instituto Nacional de Investigaciones Forestales Agrícolas y Pecuarias" for the financial support that made possible my graduate studies at O.S.U.

TABLE OF CONTENTS

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF FIGURES

LIST OF TABLES

PREFACE

This dissertation is composed of three manuscripts. Chapter 2, "The use of Lagrangean relaxation to solve the flow constrained harvest scheduling problem", has been submitted to Forest Science. Chapter 3, " Adjacency constraints in harvest scheduling: an aggregation heuristic" is under review process in the Canadian Journal of Forest Research. Chapter 4, "The use of relaxation to solve the habitat dispersion problem" has been submitted to Forest Science. All three manuscripts have been advised and criticized by Dr. J. Douglas Brodie and Dr. J. Sessions. Both have assisted since early stages of the research with discussion of theory and programming, as well as interpretation of results. The remaining errors in this dissertation are my total responsibility.

The Use of Relaxation to Solve Harvest Scheduling Problems with Flow, Wildlife Habitat, and Adjacency Constraints

INTRODUCTION

During the last few decades forest planning has become a complex decision process where silvicultural, transportation, protection and financial activities must be simultaneously considered within the harvest scheduling decisions. In this way, the economically efficient allocation of production factors and the correct distribution of products derived from the forest is ensured. The harvesting decisions have been constantly improved because of technological changes that increase the efficiency of forest activities or because of changes in the preferences of society that change the valuation of diverse goods and services derived from the forest. Hence modern forest planning is directed toward multiple-use. In the long run not only the activities related to market goods are considered, but also other goods and services that are becoming increasingly more valuable to society. Recent concerns have focused on the direct applicability of forest plans for specific small harvest units on the ground with efficient allocation to sustain timber, wildlife and

scenic values over time and space.

These modern requirements in forest planning are leading to replacement of traditional strata-based harvest scheduling plans with modern area-based plans. Through this strategy, forest plans have direct applicability in the field, the control of each harvest unit is facilitated so that production factors are more efficiently allocated and outputs are better monitored. Additionally it provides a richness of information to efficiently direct the management of non-market goods and services derived from the forest.

Several researchers have pointed out the potential of area-based models in forest planning. Nevertheless, to date little empirical work has been done to improve the solution techniques or problem formulations of the area-based harvest scheduling problems. One of the main disadvantages of current formulations is their solution difficulty given their discrete nature and dimensionality. Although sophisticated software packages and the aid of more computer power have helped to overcome some of these difficulties current algorithms used to solve the area-based harvest scheduling problem make huge computational demands, that make even moderate scale applications prohibitively expensive to implement at the field level.

This research is focused on the use of alternative methods to solve the area-based harvest scheduling problem

that offer "good" solutions without large computational requirements. The basic assumption of this work is that some manager imposed constraints can be slightly violated in a harvest schedule. These violations do not harm the "best" expected output since the discrete nature of the problem requires slacks in some constraints in order to reach integer solutions. Hence if those slacks are associated with the "flexible" manger imposed constraints an agreement between the smallest violation of a constraint and the quality of the integer solution achieved can be met.

The first manuscript, Chapter 2, describes the use of Lagrangean relaxation to solve the flow constrained area-based harvest scheduling problem. This solution strategy was originally proposed by Hoganson and Rose (1984) with some problems to obtain good estimators for the Lagrange multipliers. The manuscript emphasizes the use of the subgradient method and proposes an additional heuristic to further improve the estimations obtained from the subgradient algorithm.

The major advantage of the area-based approach is its facility to consider spatial requirements. These requirements are modeled through adjacency constraints. Conventionally these constraints are written as relationships between pairs of adjacent harvest units. However, this strategy vastly increases the number of rows in any formulation. The second manuscript, Chapter 3,

addresses this problem by introducing a heuristic to reduce the number of adjacency constraints required to specify the adjacency requirements in any formulation.

The third manuscript, Chapter 4, addresses the problem of solving harvest scheduling problems with adjacency constraints and wildlife habitat requirements. Further implementation of the relaxation method is used to solve the habitat dispersion constraints. Introduction of surrogate constraints ensures that the adjacency constraints are met exactly. The methods developed handle problem sizes that are unsolvable by integer programming. Optimal or minimally infeasible "good" solutions are found, even in cases where the specification is infeasible.

SOLUTION TO THE AREA-BASED HARVEST SCHEDULING PROBLEM THROUGH LAGRANGEAN RELAXATION

by

Juan M. Torres-Rojo

J. Douglas Brodie

and

John Sessions

ABSTRACT

The use of Lagrangean relaxation as a solution technique for the flow constrained area-based harvest scheduling problem is presented. The best set of multipliers in the Lagrangean function is searched through the subgradient algorithm. An analysis of the basic structure of this algorithm is provided. The analysis is used to set guidelines for some parameters of the subgradient algorithm in the solution of the relaxed area-based harvest scheduling problem. A procedure to further improve the multipliers provided by the subgradient method is proposed. Such a procedure is used when the solutions obtained from the subgradient algorithm do not meet the desired harvest levels or violate additional requirements. Results showed

good convergence of the subgradient algorithm. Integer solutions within 2% of the optimal integer solution can be obtained in shorter time than a solution by linear programming. The proposed improvement algorithm yields better solutions only for large problems or when the optimization includes more than 7 periods.

INTRODUCTION

The area-based approach to the timber harvest scheduling problem has become increasingly important in the last few years. This is due to its capacity to allocate forest lands to several uses subsuming the spatial distribution of the management units. The approach assigns silvicultural prescriptions to specific analysis areas or blocks, so that the biological, legal or spatial management requirements can be met to achieve multiple-use objectives during the planning interval. Likewise, through this approach it is possible to enhance both the efficient allocation of production factors among the harvest units, and the practical applicability of the management plans.

The area-based approach requires integer solutions because the harvesting decisions are dichotomous decisions, harvest or not to harvest a given unit. However, optimal solution methods for Integer Programming (IP) or Mixed Integer Programming (MIP) make enormous computational demands to obtain an optimal solution, and in some cases it is difficult to obtain even a feasible solution. Hence IP and MIP solution techniques have rarely been implemented for solving large harvest scheduling problems.

One of the most useful ideas in the solution of IP problems is the observation that "many hard problems can be viewed as easy problems complicated by a relatively small

set of side constraints" (Fisher, 1981). This set of "complicating" constraints can be dualized by using Lagrange multipliers so that the resulting Lagrangean problem is easier to solve than the original one. Additionally, the solution to the Lagrangean problem might (under certain circumstances) provide an optimal solution to the original problem or at least an upper bound (for maximization problems) of that optimal solution. Such an approach is called Lagrangean relaxation (Geoffrion, 1974).

The idea of "relaxing" the constraints of an IP problem and incorporating them into the objective function was used by Lorie and Savage (1955) in their solution procedure to discrete capital budgeting problems and later by Dreyfus (1957) to perform state variable reduction in Dynamic Programming (DP). However, the first theoretical concepts of the technique to solve discrete resource allocation problems were established by Everett (1963). The potential of Lagrangean relaxation in the solution of IP and MIP was not fully realized until Held and Karp (1970) used these principles to derive a powerful algorithm for the solution of the traveling salesman problem. Ever since, the theory of Lagrangean relaxation has been continuously refined and successfully applied to areas of IP such as scheduling, location, matching, the traveling salesman problem and network problems. Excellent presentations of the theoretical basis of the Lagrangean approach and reviews on the topic for the solution of special problems

can be found in Shapiro, (1971); Geoffrion, (1974); Fisher et al., (1975); Shapiro, (1979); Bertsekas, (1982); Fisher, (1985); Tseng and Bertsekas, (1987); and Ibaraki and Katoh, (1988).

Lagrangean relaxation has been present in the forestry literature although not precisely defined in such a way. Paredes and Brodie (1986) used the principles of Lagrangean relaxation developed by Dreyfus (1957) to develop an efficient algorithm for the solution of the Dynamic Programming (DP) formulation of the stand level optimization problem. In forest level optimization, Berck and Bible (1984) used Lagrangean relaxation to establish optimality conditions for a Model II Linear Programming (LP) formulation of the harvest scheduling problem (Johnson and Scheurman, 1977). They found an optimal solution to the original problem through complementary slackness conditions by forcing some dual variables to take known optimal values. Hoganson and Rose (1984) proposed a simulation approach to the harvest scheduling problem. Their approach was not defined as Lagrangean relaxation of an integer problem because their construction was proposed as a solution to the Model I LP harvest scheduling problem. However, as will be shown, such a construction corresponds to the Lagrangean relaxation of an integer version of the Model I harvest scheduling problem (Johnson and Scheurman, 1977). In fact their construction indeed yields integer

solutions. Hence, their approach can be considered as the first attempt to solve the IP harvest scheduling problem through Lagrangean relaxation.

Hoganson and Rose (1984), pointed out that the crucial step in their simulation approach was the process of guessing the multipliers. Eldred (1987), evaluated the Lagrangean relaxation technique suggested by Hoganson and Rose. He also found that the main problem of the technique is the difficulty in finding a good set of multipliers. He proposed another set of heuristics to search multipliers besides the ones proposed by Hoganson and Rose. However the search still had some problems.

The multiplier search is the most important and delicate step in any Lagrangean relaxation approach and has become a central topic of research in IP. According to Fisher (1981) there exist three strategies for the multiplier search, namely: a) the subgradient method, b) the column generation method and c) heuristics. Among these strategies the subgradient method has been preferred because it is simple and yields good approximations. However, some heuristics for specific problems have proved to be very successful. Held et al., (1974); Fisher et al., (1975); Fisher, (1981); Bertsekas, (1982); Fisher, (1985); and Ibaraki and Katoh, (1988); provide excellent summaries of some successful approaches to the multiplier search for specific problems.

We will present the use of the subgradient method to search the multipliers of the Lagrangean relaxation of the flow constrained area-based harvest scheduling problem. We will first show the IP harvest scheduling problem and a proof that the construction by Hoganson and Rose (1984) corresponds to the dual of the Lagrangean relaxation of the integer programming (IP) harvest scheduling problem. Then we describe briefly the theoretical basis of the Lagrangean approach and the subgradient algorithm and provide some guidelines to set the parameters of this algorithm for the harvest scheduling problem. A procedure will be described to further improve the solutions obtained through the subgradient algorithm. Finally, we present some comparative results and discuss briefly some extensions of the procedure to the solution of area-based harvest scheduling problems with additional constraints.

THE AREA-BASED HARVEST SCHEDULING PROBLEM

The flow constrained area-based harvest scheduling problem is an Integer Programming problem that can be stated as:

$$Z_P = \max_{x_{ij}} \sum_{i=1}^{m} \sum_{j=1}^{n_i} c_{ij}\, x_{ij} \qquad (P)$$

s. t.

$$(1) \quad \sum_{i=1}^{m} \sum_{j=1}^{n_i} [V_{ijt} - V_{ij(t+1)}] x_{ij} = 0 \quad \forall\ t = 0, 1, .., T$$

$$(2) \quad \sum_{j=1}^{n_i} x_{ij} = 1 \quad \forall\ i = 1, 2, .., m$$

$$(3) \quad x_{ij} \in \{0, 1\} \quad \forall\ i = 1, 2, .., m \quad \forall\ j = 1, 2, .., n_i$$

where:

$x_{ij} = \begin{cases} 1 \text{ if the treatment "j" in analysis area "i" is selected. The option of not to harvest in any period is considered a possible treatment for each analysis area.} \\ 0 \text{ if the treatment "j" in analysis area "i" is not selected.} \end{cases}$

c_{ij} : Total present net value derived from allocating treatment "j" in analysis area "i" during "T" periods. Period "T+1" is considered the first period. Observe that each analysis area "i" has n_i treatments and the value of c_{ij} accounts for the acreage variation in different analysis areas.

V_{ijt}: Volume harvested in analysis area "i" under treatment "j" in period "t".

The set of constraints (1) represents the strict even-flow constraints. Constraint set (2) is the set of area restrictions, and set (3) represents the integer constraints of the problem. Observe that if we relax the integer restrictions (3) (i. e. a linear programming relaxation of P), each decision variable "x_{ij}" will represent the fraction of the i-th analysis area managed under treatment "j". In that case, such formulation can be accommodated in the context of a Model I Linear Programming (LP) formulation of the harvest scheduling problem (Johnson and Scheurman, 1977). In addition, it can be easily observed that the set of area constraints (2) in the LP relaxation of (P) imposes natural bounds in the decision variables, i.e. each non-negative x_{ij} can not be greater than 1 ($0 \leq x_{ij} \leq 1$).

Substituting "d_{ijt}" for the values "$V_{ijt} - V_{ij(t+1)}$" in constraint set (1) and relaxing the constraint set (3), the dual of problem P (D1) can be stated as:

$$Z_{D1} = \min_{a_i} \sum_{i=1}^{m} a_i \qquad \text{(D1)}$$

s. t.

(1) $\sum_{t=1}^{T} d_{ijt}\, u_t + a_i \geq c_{ij}$ $\quad \forall\ i = 1, 2, .., m$; $\forall\ j = 1, 2, .., n_i$

(2) u_t unrestricted $\quad \forall\ t = 1, 2, .., T$

(3) a_i unrestricted $\quad \forall\ i = 1, 2, .., m$

where u_t ($t = 1, 2, .., T$) is the Lagrange multiplier associated with the " t-th" even flow constraint in problem

(P) and " a_i " (i = 1, 2,.., m) is the multiplier associated with the "i-th" area constraint. Hoganson and Rose (1984) -- working with a Model I LP formulation -- obtained the same dual problem as (D1). This is a logical result given that to obtain (D1) we relaxed the integer constraints in problem (P), so (D1) is the dual of an LP problem (Model I formulation). Hoganson and Rose (1984) determined that by guessing a value for each one of the u_t (t=1, 2, ..., T), problem (D1) could be rewritten as:

$$Z_{D2} = \min \sum_{i=1}^{m} a_i \qquad \text{(D2)}$$

s. t.

$$a_i \geq c_{ij} - \sum_{t=1}^{T} d_{ijt}\, u_t \qquad \forall\ i = 1, 2, .., m;\ \forall\ j = 1, 2, .., n_i$$

$$a_i \text{ unrestricted} \qquad \forall\ i = 1, 2, .., m$$

As they suggested, such a problem can be easily solved just by finding for each a_i:

$$\max_{1 \leq j \leq n_i} \{ c_{ij} - \sum_{t=1}^{T} d_{ijt}\, u_t \} \qquad \forall\ i = 1, 2, .., m \qquad (1)$$

This reduced problem consists of finding for each analysis area the treatment with the largest "weighted cost coefficient", where the "weights" are represented by the expression:

$$\sum_{t=1}^{T} d_{ijt}\, u_t$$

If this solution provides a feasible solution to problem (P), then the primal problem is solved and additionally the

"best" guess for u_t has been found. Their procedure consists of guessing different u_t values, then checking the feasibility of the primal solution associated with that guess. If a permissible deviation of feasibility was not reached then another set of multipliers is guessed to reinitialize the iterative procedure. Otherwise the search is terminated.

Assume that for a given guess of u_t:

$$\max_{1\le j\le n_i} \{ c_{ij} - \sum_{t=1}^{T} d_{ijt}\, u_t \} = c_i^* - \sum_{t=1}^{T} d_{it}^*\, u_t = a_i^* \qquad (2)$$

Substituting the right hand side of (2) in problem (D2) yields a solution for problem (D2) (Z_{D2}) given a guess for u_t; such solution is:

$$Z_{D2} = \sum_{i=1}^{m} (c_i^* - \sum_{t=1}^{T} d_{it}^*\, u_t) \qquad (3)$$

which is the solution of the dual problem solved by Hoganson and Rose (1984) at each iteration. We can verify that this procedure yields only integer solutions since in problem D2 for each a_i only one constraint is active (if there are no multiple optima), which implies that only one primal variable (one treatment) for each analysis area will be selected.

Now let us go back to problem (P). By dualizing the set of constraints (1) in (P) the Lagrangean relaxation of this problem yields:

$$Z_D(u) = \max_{x_{ij}} \sum_{i=1}^{m} \left(\sum_{j=1}^{n_i} c_{ij} - \sum_{j=1}^{n_i} \sum_{t=1}^{T} d_{ijt}\, u_t \right) x_{ij} \qquad \text{(LR)}$$

s. t.

$$(1) \qquad \sum_{j=1}^{n_i} x_{ij} = 1 \qquad \forall\ i = 1, 2, .., m$$

$$(2) \qquad x_{ij} \in \{0, 1\} \qquad \forall\ i = 1, 2, .., m$$

$$\forall\ j = 1, 2, .., n_i$$

Such a problem can be divided into "m" subproblems in which the k-th subproblem, LR-k, is:

$$Z_{D-k}(u) = \max_{1 \le j \le n_k} \left(c_{kj} - \sum_{t=1}^{T} d_{kjt}\, u_t \right) x_{kj} \qquad \text{(LR-k)}$$

$$x_{kj} \in \{0, 1\} \qquad \forall\ j = 1, 2, .., n_k$$

For a given guess of u_t the solution to problem LR-k will yield only one x_{kj} optimum (x^*_{kj}) which obviously will take its upper bound (recall that the non-harvesting option is considered also a treatment). Let us assume that such a solution is:

$$Z_{LR-k}(u) = \left(c^*_k - \sum_{t=1}^{T} d^*_{kt}\, u_t \right) x^*_{kj} \qquad (4)$$

substituting (4) in problem LR and considering that $x^*_{kj}=1$ (for all k = 1, 2, .., m) the solution to LR for a given guess of u_t yields :

$$Z_D(u) = \sum_{i=1}^{m} \left(c^*_i - \sum_{t=1}^{T} d^*_{it}\, u_t \right) \qquad (5)$$

which is exactly the same result as the one obtained in (3) with the construction outlined by Hoganson and Rose (1984). Therefore, their simulation approach to solve the Model I LP harvest scheduling problem corresponds to the Lagrangean

relaxation solution of the area-based harvest scheduling problem when dualizing the even-flow constraints.

Let **c** be the N x 1 vector of cost coefficients "c_{ij}", where N is the total number of variables in the problem. Let **D** be the T x N matrix of coefficients "d_{ijt}" , let **x** be the N x 1 vector of primal variables x_{ij}, and let **A** be the m x N matrix of area constraints. The harvest scheduling problem (P) can now be rewritten as problem P1 (figure 1). Such a problem has three Lagrangean relaxations. The first one is obtained by dualizing constraint set (1) in P1, yielding problem LR (figure 1). As discussed above the resulting Lagrangean problem is easy to solve and provides integer solutions. Hence it is a good candidate to be used in a Lagrangean relaxation approach. The second relaxataion can be obtained by dualizing constraint set (2) in problem P1. By defining μ as the vector of dual variables associated with the area constraints such a relaxation yields:

$$Z_D(\mu) = \max_{x} \mathbf{c'x} + \mu'(\mathbf{Ax} - \mathbf{1})$$

subject to constraint sets (1) and (3). The resulting Lagrangean problem can be seen as a 0-1 multi-constrained knapsack problem, which is more difficult to solve than the previous one. The structure of this relaxation requires the computation of as many multipliers as there are harvest units, in contrast with the previous relaxation which only requires the estimation of one multiplier per period considered in the optimization. In addition, it is very

likely that the best solution we can get through Lagrangean relaxation will violate the area constraints. Such violations in harvest scheduling problems are more critical than violations to manager-imposed strict flow constraints which are usually infeasible in an integer problem.

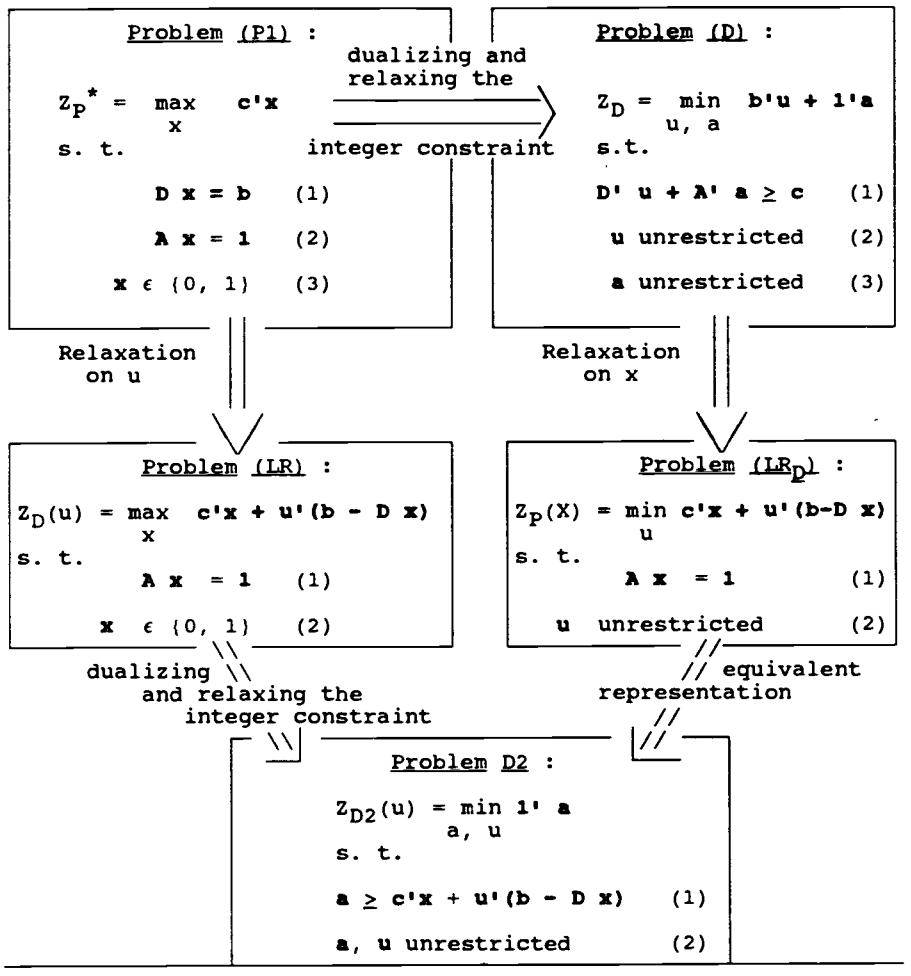


Figure 1. Relations among the integer primal problem (P1), the dual of the primal problem with the integer constraints relaxed (D), the Lagrangean relaxation on **u** of the primal problem (LR), the Lagrangean relaxation on **x** of the dual problem (LR_D), and the Hoganson and Rose (1984) specification for the Model I formulation of the harvest scheduling problem (L_{D2}).

The third relaxation is obtained by dualizing both the flow and the area constraints. This relaxation yields:

$$Z_D(u,\mu) = \max \mathbf{c}' \mathbf{x} + \mathbf{u}'(\mathbf{b} - \mathbf{D}\mathbf{x}) + \mu'(\mathbf{A}\mathbf{x} - \mathbf{1})$$

subject to the integer restrictions. Although this Lagrangean problem is simply stated, it is more difficult to guess a set of multipliers that yields feasible primal solutions. In this case the best solution for the Lagrangean problem will be equivalent to the solution obtained for the dual of the LP relaxation of problem (P1), which is not necessarily primal feasible. Theoretically such a solution provides bounds less tight (poor integer solutions) for the true integer solution than a relaxation without all the constraints dualized (Gavish and Pirkul, 1985).

Considering the three Lagrangean relaxations of the area-based harvest scheduling problem, we have shown that only by relaxing the flow constraints (or any other set of linking constraints) we will be able to obtain integer solutions and additionally a Lagrangean problem that is easy to solve. Such conditions can not be met by relaxing the area constraints or all the constraints at the same time.

THE SUBGRADIENT METHOD

It is well known that the solution to the Lagrangean problem provides an upper bound on the optimal primal value (for maximization problems). For instance, if in problem LR the vector **u** is nonnegative and the vector (**b -Dx**) is also nonnegative then the Lagrangean problem (LR) is an upper bound of the primal solution Z_P^*, since we are just adding a nonnegative term into the objective function of problem (P1) (figure 1). The same is true if the sign of the scalar $(\mathbf{b} - \mathbf{Dx})_i$ equals the sign of u_i. In practice given the way the multipliers are adjusted, $Z_D(u) \geq Z_P^*$ for most of the guesses. It is possible though to have a dual solution whose associated primal solution has a larger objective function value. Intuitively we can observe that the best choice for the vector **u** in problem (LR) would be the solution of its dual problem, i. e.

$$Z_D(u^*) = \min_u \; Z_D(u) \tag{6}$$

Observe that such a solution is the Lagrangean relaxation of the dual of problem (P1), letting **x** be the vector of dual variables and constraining the dual problem such that **Ax = 1** (see figure 1). Thus, if we relax the integer constraints (3) in problem (P1) we have the following relationships for the Lagrangean relaxations of primal and dual problems:

$$Z_D(u^*) = \min_u \; Z_D(u) = Z_P(x^*) = \max_x \; Z_P(x) \tag{7}$$

Most of the algorithms for determining **u** in the solution of the Lagrangean problem $Z_D(u)$ have as their objective to find an optimal or close to optimal solution to $Z_P(x)$. Ideally when we have a solution that meets the relationships in (7) as closely as possible we have the best solution Z_P for the original problem. In general, because of the duality gap (a duality gap is the difference between Z_P^* and its associated dual solution in problem D), caused by the integer constraint (3) in (P1), it is not possible to find a vector **u** for which $Z_D(u) = Z_P^*$. If such a vector existed (in fact it exists when there is no duality gap), it would correspond to the optimal solution of problem (P1) . Bertsekas (1982), has shown that the "ratio of the duality gap over the optimal primal value goes to zero as the number of variables goes to infinity". Hence as the problem has more integer variables, the best choice for $Z_D(u)$ becomes a better estimate for Z_P^*.

Assume a harvest scheduling problem with just one flow constraint dualized. In terms of problem (P1) such a problem would correspond to a problem with just two periods. For any guess of the vector **u** a solution **x** for $Z_D(u)$ is obtained. Hence the value of $Z_D(u)$ depends on the guesses of **u**. The function $Z_D(u)$ is piecewise linear on u (see figure 2). Each line segment of this function corresponds to the primal solution associated with that specific value of u. In addition, $Z_D(u)$ has a nice behavior for linear problems. It is a convex function

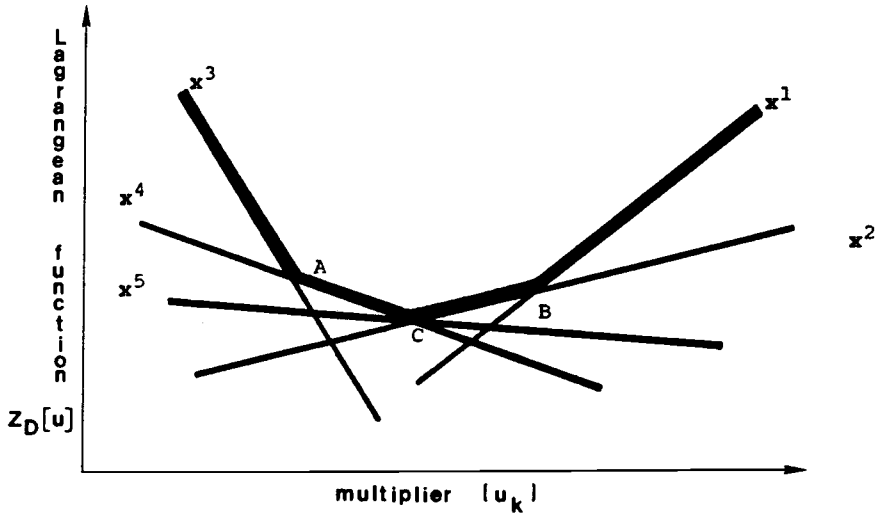


Figure 2. The Lagrangean function $Z_D(u)$

differentiable everywhere except at points where one guess of **u** yields multiple primal solutions. These points can be observed in figure 2 and correspond to the intersection of two or more line segments (e.g. points A, B, and C in figure 2). The subgradient method exploits such behavior of the Lagrangean function making an adaptation of the gradient search method at points where the function $Z_D(u)$ is nondifferentiable. For differentiable points the subgradient method estimates the search direction as the "gradient search" by taking the gradient of the Lagrangean function. Such gradient is given by the derivative of $Z_D(u)$ with respect to **u**:

$$\frac{\partial Z_D(u)}{\partial \mathbf{u}} = \mathbf{b} - \mathbf{D}\,\mathbf{x} \qquad (8)$$

As the reader can verify this derivative corresponds to the vector of infeasibilities in the harvest flow constraints given the current guess of **u**. At nondifferentiable points the subgradient method chooses arbitrarily an alternative primal solution **x**, computes the infeasibility vector (**b** - **D** **x**) and uses that vector as if it were the true gradient of $Z_D(u)$. In this way the sequence of values of the multipliers to search the minimum of $Z_D(u)$ is given by the formula:

$$\mathbf{u}^{k+1} = \mathbf{u}^k + t_k\,(\mathbf{D}\,\mathbf{x}^k - \mathbf{b}) \qquad (9)$$

where "t_k" is a positive step size and $\mathbf{x}^k$ is the current solution to $Z_D(u)$ associated to the guess $\mathbf{u}^k$. In the subgradient method the fundamental relationship on which computation of step size depends is:

$$Z_D(u) \longrightarrow Z_D \quad \text{if} \quad t_k \longrightarrow 0 \quad \text{and} \quad \sum_{i=1}^{\infty} t_i \longrightarrow \infty$$

In practical terms this result states that "t" has to converge to zero but not necessarily quickly. Many step sizes have been proposed (Shor, (1964); Polyak (1969); Held and Karp (1971); Held et al., (1974); Allen et al., (1987)). However, the step size most commonly used is:

$$t_k = \frac{\lambda_k\,[Z_D(u^k) - \hat{Z}]}{\| \mathbf{b} - \mathbf{D}\,\mathbf{x} \|^2} \qquad (10)$$

This step size was proposed initially by Polyak (1969) and its convergence properties have been discussed by Polyak (1969) and Held et al., (1974). In (10) λ_k is a scalar satisfying the constraints $0 \leq \lambda_k \leq 2$. The usual practice is setting $\lambda_0 = 2$ and halving λ_k whenever $Z_D(u)$ has failed to decrease $Z_D(u^k)$ in a specific number of iterations (number of cycles). Such number of iterations is often halved as the value of λ is halved (Held et al, 1974). The value $Z\hat{}$ is usually set to zero and it is updated every time the Lagrangean problem yields a better feasible primal solution. Sometimes when the structure of the problem does not produce a primal feasible solution as a byproduct of the search then $Z\hat{}$ is set to an upper or lower bound of the expected Z_P^*. Figure 3 shows the flow chart with the basic steps to perform a search for the best multipliers of a Lagrangean relaxation using the subgradient algorithm.

In general the subgradient algorithm performs very well for most of the problems, although good results from special applications have required some modifications to its basic structure. In the next section we describe some modifications that proved to be effective to improve the results obtained from the subgradient algorithm.

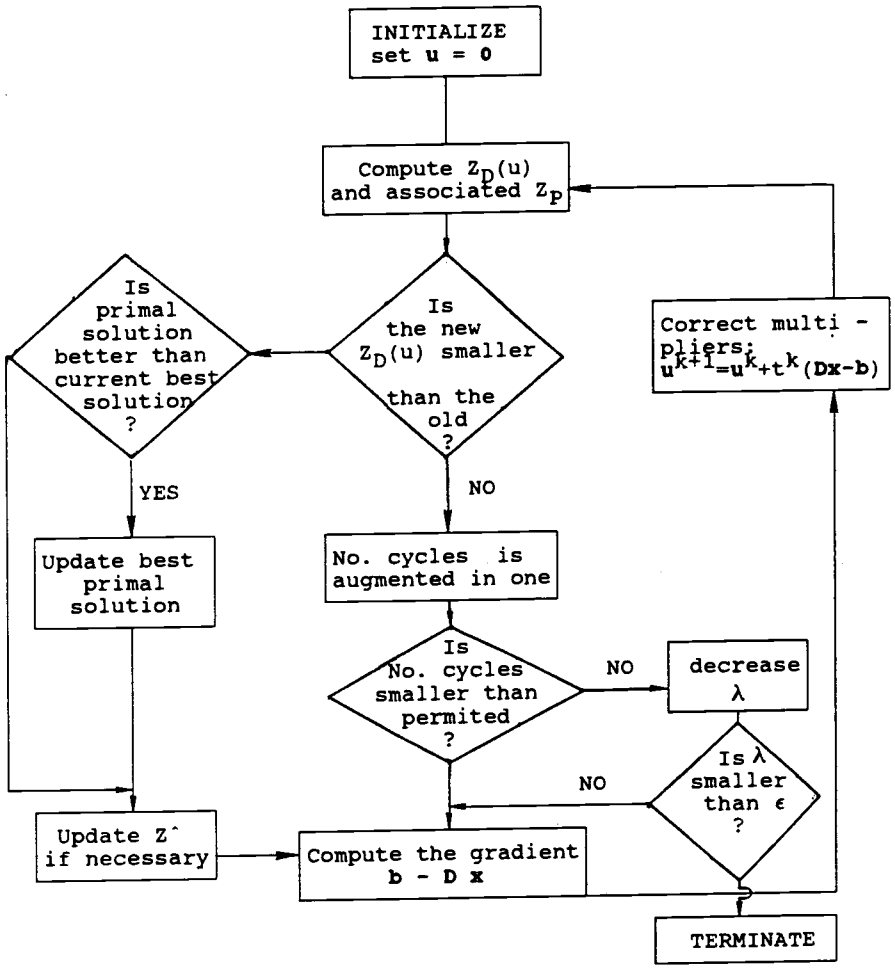


Figure 3. A flow chart of the subgradient algorithm.

IMPROVEMENTS ON THE SUBGRADIENT METHOD APPLIED TO THE HARVEST SCHEDULING PROBLEM.

The problem structure of any harvest scheduling problem is the same, however the size varies with each application. We adopted as a measure of size of the problem the expression:

$$\text{size} = \sqrt{\text{No. of periods} * \text{total number of integer variables}}$$

This expression is a linear measure of the size of the harvest flow matrix that is searched for the dual solution. Since the whole matrix is searched at each iteration the expression reflects the number of repeated computations per iteration. In the rest of the paper we will refer to problems with size less than 40 as small problems, those with size between 40-70 as medium, and those with size larger than 70 as large.

When the subgradient algorithm is being implemented many primal solutions are provided. In the case of problem (P1) (figure 1) all of these solutions are primal infeasible. If they were feasible solutions, then by duality theory they would correspond to the optimal primal solution. Given this inconvenience, we need a criterion to distinguish one primal infeasible solution from another primal infeasible solution considered better. One might be tempted to say that a "better" primal infeasible solution is that solution with the largest objective function value.

However this is not true, otherwise we would always select the solution corresponding to the problem restricted only to the area constraints, i.e. the primal solution associated with the Lagrangean problem with $\mathbf{u} = \mathbf{0}$.

A better selection criterion consists of choosing the primal solution with the smallest sum of absolute infeasibilities i. e. the smallest $\|\mathbf{b} - \mathbf{D}\,\mathbf{x}\|$. This criterion is based on the relationship between the primal and Lagrangean problem and the definition of a primal optimal solution. Recall that an optimal primal solution for problem (P1) (figure 1) corresponds to the solution with the largest objective function value and where all constraints are satisfied. Likewise, recall that in the last section we stated that the solution of the Lagrangean problem is an upper bound of the primal optimal solution. Using these statements we can say that the primal solution associated with a Lagrangean problem with minimum value, which in addition has the smallest infeasibilities, is a better primal solution than a solution with larger objective function value and larger sum of absolute infeasibilities. Figure 4 shows the relationship among the Lagrangean function $Z_D(u)$, its associated primal solution and the sum of absolute infeasibilities in the primal problem during the search for the minimum of the Lagrangean function. The figure was constructed using just the points of improvement in the search i.e., those points for which the sum of absolute infeasibilities was smaller than the

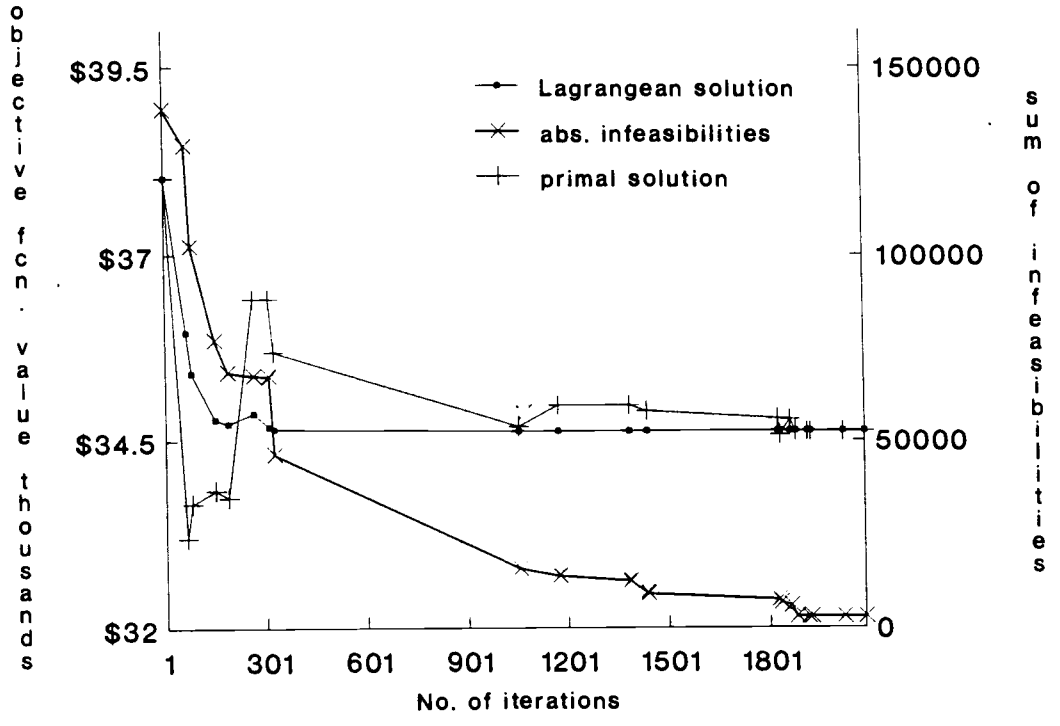


Figure 4. General trend of the search and its relation with the sum of absolute infeasibilities.

previous smallest sum of absolute infeasibilities. Observe that the smaller the value of $Z_D(u)$ in the final convergence of the algorithm, the smaller the sum of absolute infeasibilities. Also observe, that the primal solutions associated with these Lagrangean problems tend to converge to the value of the Lagrangean function as the search continues but they do not have a homogeneous pattern.

In general the solutions with smaller values for $Z_D(u)$ are associated with primal solutions closer to the true optimum, which sometimes are larger and sometimes smaller

than the optimum. Figure 4 suggests that most of the infeasible solutions obtained after early searching are superoptimal. However this is not true, figure 5 shows the same search after iteration 710 every 3 iterations with an extended scale. Observe that most of the infeasible solutions obtained during the search are suboptimal. Hence the theoretical argument that the best primal solution is increasing as the search of the minimum of the Lagrangean function continues is met. These results suggest that the value of the primal objective function is not a good indicator of the goodness of an infeasible primal solution. However, the sum of absolute infeasibilities behaves consistently with the kind of solutions we want to achieve throughout the search. Hence in the rest of this paper we will consider as the "best" solution the primal solution with the smallest sum of absolute infeasibilities.

Many authors have emphasized that the efficiency of the subgradient method in finding the minimum of the Lagrangean function $Z_D(u)$ is dependent on an adequate match between the step size selection and the problem structure. The structure of a problem can be such that the step size has a minimum effect in the search. For instance, Held and Karp (1971) used a constant step size t = 1 in the solution of the traveling salesman problem. This step size apparently performs better than the variable step size considered in equation (10). Other problem structures require a more precise computation of the step size. Such is the case for

the non-integer traveling salesman problem, the maximum flow problem and the multicommodity flow problem analyzed by Held et al.,(1974). In an attempt to find an appropriate step size for our specific problem or at least to find any relationship among some parameters of the subgradient algorithm and problem size we analyzed several parameters of the subgradient algorithm. The parameters involved in the analysis were: the reduction factor of λ, the choice of the subgradient at nondifferentiable points, the selection of $Z^{\wedge}$, and the number of iterations permitted before reducing λ.

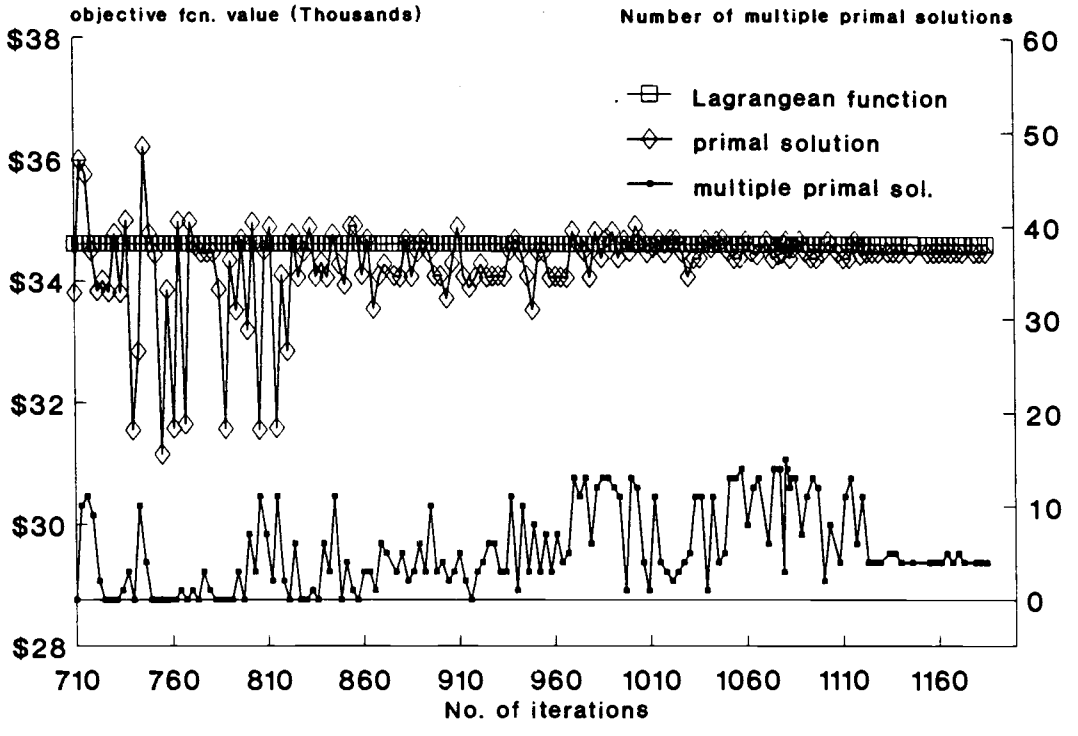


Figure 5. Convergence in late stages of the search.

The reduction factor of lambda.

The pattern of convergence is highly related to the reduction factor in lambda. As we mentioned before, conventionally lambda is halved after NI number of iterations have failed to yield a lower $Z_D(u)$ (we used NI = 20 in the comparisons). For almost all harvest scheduling problems we tested (small problems are excluded), if lambda is halved after NI iterations the pattern of convergence is less smooth than if lambda is reduced only 25 percent after NI iterations. If the convergence pattern is not smooth, then it is very likely that the search deviates and converges to a poor solution. Such solutions are characterized by a large sum of absolute infeasibilities, large values in the minimum Lagrangean function $Z_D(u)$ and generally superoptimal primal solutions. Figure 6 shows the effect of decreasing the reduction factor of lambda by several percentages. As can be observed the smaller the reduction of lambda after NI iterations the better the final solution. This behavior can be attributed to the fact that with smaller reductions of lambda the spectrum of values that the multipliers can take is larger and the search is more likely to hit the correct ones. This feature might suggest that by reducing lambda by small amounts the algorithm yields better convergence. However this procedure has a negative side effect. As we decrease the reduction factor of lambda the

number of iterations required increases exponentially and obviously such an increment is accompanied by a proportional increment in solution time. For this reason it is helpful to decrease the reduction factor of lambda but such a reduction needs to be accompanied by another feature that permits a smooth convergence with, lesser number of iterations.

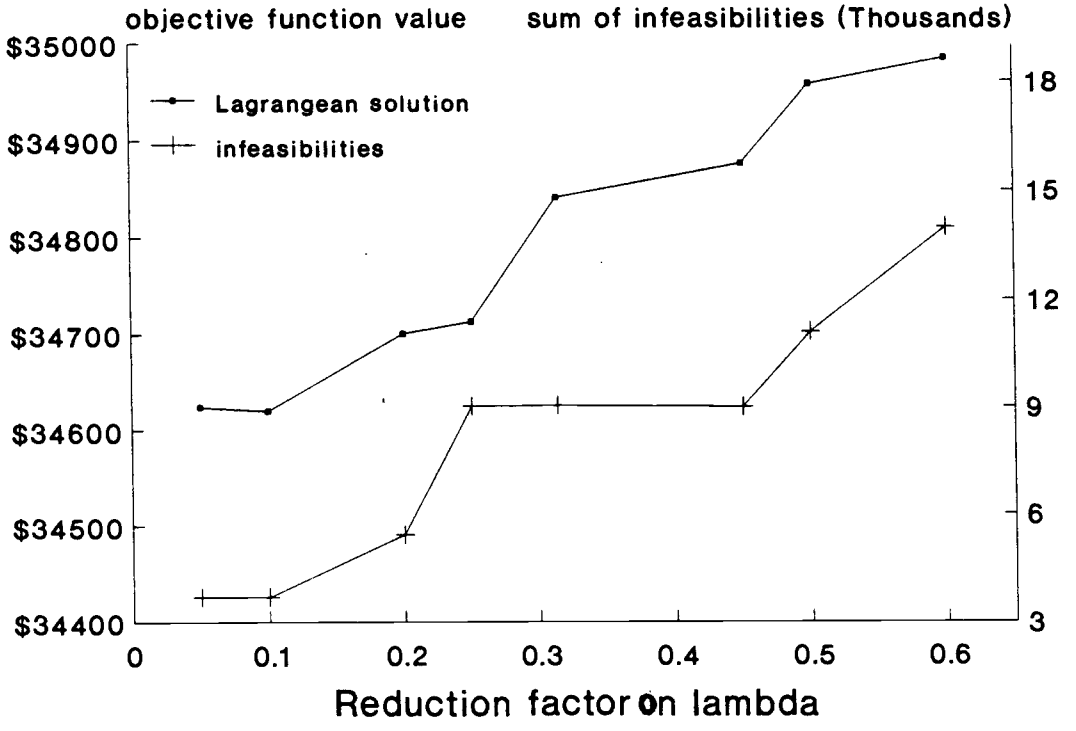


Figure 6. Effect of the reduction factor on lambda in the final solution.

The choice of the subgradient at nondifferentiable points

For a wide range of problem sizes, the search presents a trend similar to the one shown in figure 4. As can be observed the largest improvements in the search occur during the first iterations. So after a relatively good solution of the Lagrangean function $Z_D(u)$ has been found, the convergence becomes very slow. Consider figure 5. Observe how as the search proceeds small changes in the Lagrangean function (barely perceptible in the figure) are associated with large changes in the primal objective function. This observation suggests that as the algorithm converges to the "best" solution and there are small changes in the multipliers, the number of primal solutions associated with a single dual solution increases.

Consider figure 2. Observe that at nondifferentiable points there are more than one primal solution (intersecting lines) associated with a value of u_k and $Z_D(u)$. Moreover, at points where $Z_D(u)$ reaches its minimum the number of multiple primal solutions increases. This is a hypothetical example, however, we counted the number of multiple primal solutions for our example problems and the trend is general. Figure 5 shows the trend of the number of multiple primal solutions associated with the Lagrangean

function given a multiplier guess[1/] and its relation with the convergence of the algorithm. Observe that as the algorithm converges to the "best" solution the number of associated primal solutions increases, as opposed to the first stages of the search where there are no more than one associated primal solution per dual guess. As we mentioned in the last section, the subgradient algorithm selects any of the alternative primal solutions, computes the subgradient and takes it as the true gradient. However, if we happen to select a very bad solution, the new direction such a solution will provide could deviate the search.

For this reason we propose to select the primal solution with the largest sum of absolute infeasibilities (i.e. the "worst" primal solution) as the subgradient at nondifferentiable points of $Z_D(u)$. One might be tempted to

---

1/ This number corresponds to the counting of primal variables (treatments) whose associated dual constraints were binding, given a bound defined by another primal variable, i.e. if for 3 treatments of a given stand their dual constraints were binding, only one defined the solution and the other two entered in the counting. This means that the number of primal solutions further increases if we consider all the combinations among these counted primal variables. Any dual constraint was considered binding if its difference with the maximum was less than or equal to 10^{-6}.

say that the "best" associated primal solutions (solutions with the smallest sum of absolute infeasibilities) will yield a better and faster convergence. However, for large problems where there exist multiple primal solutions per dual guess we want a slow convergence. Hence if at nondifferentiable points of the Lagrangean function we take the "best" primal solution as the subgradient, the search might yield abrupt "improvements" (i.e. it reaches a local minimum of Z_D) at those stages and might lead to a poor final convergence.

We tested three ways of selecting the subgradient at nondifferentiable points for several problem sizes. To homogenize the comparisons, the size of the problems depended only on the number of periods selected in the planning horizon, i.e. the harvest scheduling problems were solved for the same forest with different periods. Results are shown in figure 7. As can be observed for small problems there is no difference in the strategy selected. This is because there are not many multiple primal solutions per dual guess. However when those multiple solutions increase in large problems we always obtained a better solutions by applying the strategy of selecting the "worst" associated primal solution as the solution to define the subgradient of the search. Likewise as expected, the strategy of selecting any associated primal solution as the subgradient of the search was always

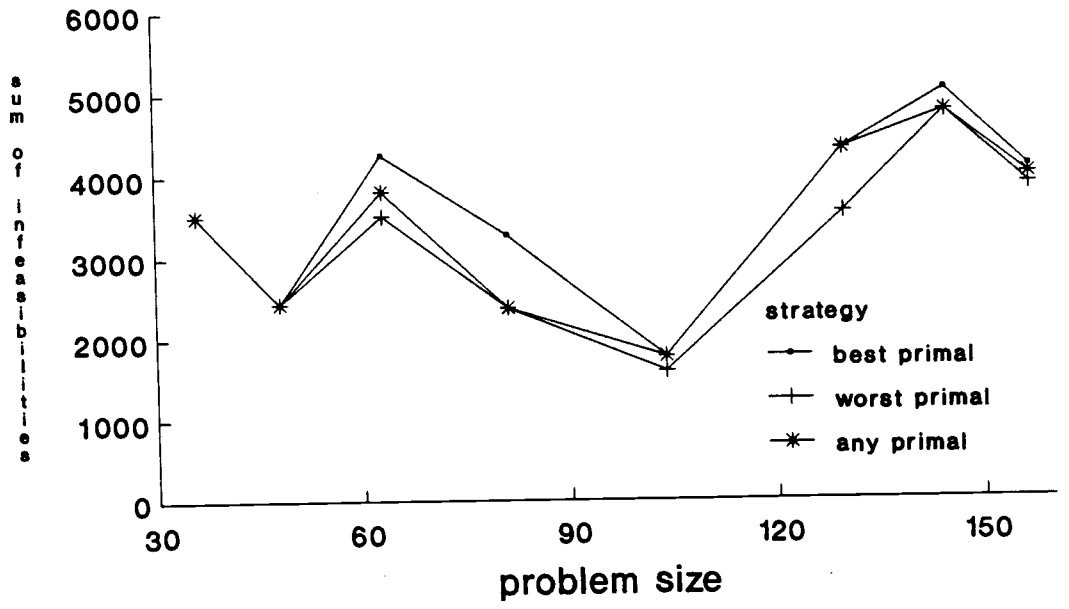


Figure 7. Effect of different strategies to select the subgradient at nondifferentiable points.

at least as good as the strategy of selecting the "best" solution as the subgradient, and for some problems was as good as the strategy of selecting the "worst" solution. In general, the strategy of selecting the "worst" primal solution to define the subgradient of the search at nondifferentiable points of $Z_D(u)$ yielded better solutions for large problems than the other strategies tested. There is a negative side to applying this strategy. The time required to finish the search is longer than the time required by the conventional strategy . Hence the proposed strategy is valuable to obtain better solutions under the knowledge that it slows down the search. Likewise, we should use it for medium-to-large problems

since for small problems there are not many associated primal solutions per multiplier guess.

Selection of $Z\hat{}$

For most of the applications of the subgradient algorithm, the search provides a feasible solution as it goes on. Such solutions are used to update $Z\hat{}$ in the computation of the step size (equation 10). In other cases such feasible solutions are not available and the guess of $Z\hat{}$ depends on previous knowledge of the problem i.e. the optimal primal solution Z_P^*. For problem (P1) (figure 1) an estimate of the value of Z_P^* is not available. Moreover, when the multiplier search is performed, no feasible primal solutions are generated. Given these conditions we analyzed the following strategies for the guess of $Z\hat{}$ necessary to compute the step size:

a) The "best" primal infeasible solution.

b) The current smallest value of $Z_D(u)$ reduced by a percentage.

For almost all cases we tested, the first strategy had convergence problems. Such problems were due to the fact that the value of the step size "t" (equation 10) has to be positive, and this choice of $Z\hat{}$ often produced negative step sizes. These negative step sizes were computed at the beginning of the search, when many superoptimal infeasible solutions with small infeasibilities are generated. Such values were ignored and the search seemed to deviate at

those points since in all tests the number of iterations was dramatically reduced and poor convergence was reached.

The second strategy is based on the theoretical result stated in the last section which indicates that the Lagrangean function $Z_D(u)$ is an upper bound of the true value of Z_P^*. Then reducing $Z_D(u)$ by a fraction should provide a good estimate of Z_P^*. Following this strategy the value of $Z\hat{}$ is forced to be always smaller than $Z_D(u)$, so there is no possibility of computing a negative step size in equation (10). The "current smallest value of $Z_D(u)$ can be reduced by several percentages. We analyzed the effect of changing this percentage for some example problems of different size. Figure 8 shows these results. In general, as we increased the percentage of reduction in the minimum $Z_D(u)$ to estimate $Z\hat{}$ a better convergence was obtained. For small size problems this difference was not as evident as for large problems where the improvement in the convergence was exceptional. The increment in the percentage of reduction of $Z_D(u)$ was always accompanied by an increment in the number of iterations required until the end of the search. This feature indicates that as we increased the percentage of reduction in the minimum $Z_D(u)$ the number of times that an improved solution was found was larger than with low percentages of reduction, i.e. the convergence was smoother by using large percentages of reduction.

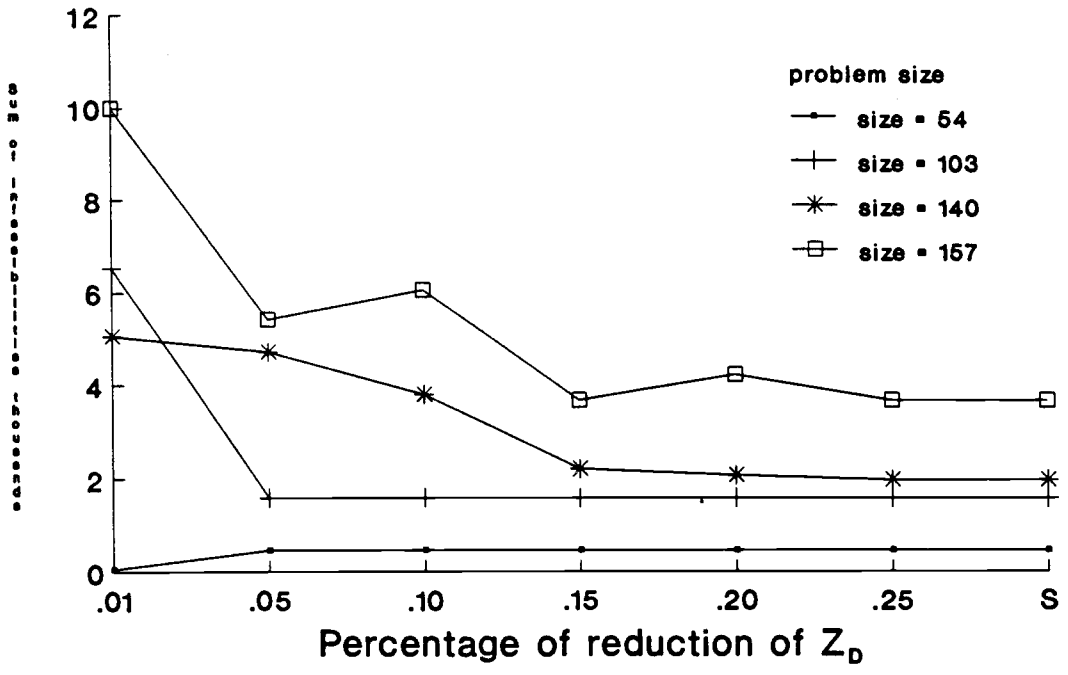


Figure 8. Effect of several percentages of reduction of $Z_D(u)$ to estimate $Z^{\wedge}$.

Larger percentages of reduction of $Z_D(u)$ to estimate $Z^{\wedge}$ make larger step sizes, which as explained above, helps to make smooth the convergence, so better solutions are obtained.

From figure 8 we can observe that a 20% reduction in $Z_D(u)$ to estimate $Z^{\wedge}$ is the lowest percentage that yielded the best convergence for large problems. Likewise, for medium size problems 5 % of reduction always yielded good convergence. Since the increment in the reduction percentage of $Z_D(u)$ to estimate $Z^{\wedge}$ increases the number of iterations and given that once $Z_D(u)$ starts to converge we need small step sizes, we tested the strategy of starting

the search with an estimate of $Z^{\wedge}$ given by 20% reduction in $Z_D(u)$ and once $Z_D(u)$ started to converge, i.e. its value did not change by more than 10^{-3} after 10 iterations, then we gradually reduced this percentage until a minimum of 5% was reached. Figure 8 shows these results under the abscissa marked with an "S". As can be observed, in all cases the proposed strategy yielded results at least as good as the best found with the strategy of constant percentage of reduction. Additionally this strategy reduced by an average of 36% the number of iterations required to end the search.

Number of iterations that the algorithm has failed in reducing the Lagrangean function

A record on the times and parameters of the best solution for some example problems showed that the improvements to the best solution are associated with the first 20 iterations every change in lambda and occasionally for big problems the first 40 are important. As the search continues, just the first five iterations after lambda has been changed have some positive effect in the search. In other words, the number of iterations that the algorithm has failed to reduce $Z_D(u)$ (cycle length of lambda) is usually less than 40 and variable throughout the search.

We performed a sensitivity analysis on the cycle length of lambda in order to define its appropriate magnitude

according to problem size. We tested 5, 10, 15, 20, 30 and 50 cycles. Results showed that even for small problems the cycle length of lambda has to be greater than 5. For small problems no difference in the final solution was detected by using cycle lengths between 10 and 50. However, the number of iterations increased linearly as we increased the cycle length and this increment was accompained by an increment in solution time. For large problems the cycle length seems to have some effect. The solutions were better as we increased the cycle length. Beyond 30 iterations no difference in the final solutions was detected other than the increment in the number of iterations and the associated longer time to end the search. Once we detected that the bounds of the cycle length are around 10 and 30 we tested the strategy of reducing the cycle length as the search continues starting with a maximum of 30 until a minimum of 10 is reached. The reduction strategies tested were:

a) Held and Karp (1971) strategy of reducing the cycle length in the same proportion as lambda is decreased until reaching a minimum of 5. By using $\lambda = 0.7$ approximately in 5 iterations we reach that minimum of 5.

b) Reducing the cycle length every 5 reductions in lambda by the same reduction factor in lambda until a cycle length of 10 is reached.

c) Reducing the cycle length every 5 reductions in

lambda after $Z_D(u)$ stabilizes i.e. it does not change by more than 0.001 of its absolute value.

As we expected the first strategy always converged to poor solutions when applied to big problems. When applied to small problems the strategy did not yield better solutions but reduced the time dramatically. For instance by using a $\lambda = 0.7$ a problem with 50 stands and 5 periods (308 integer variables) was solved in less than 18 seconds (using a Compaq 386/25) and the solution converged to the "best" integer solution of the problem. The second strategy was very variable. It was tested just with big problems given that the results on the first strategy proved that there is no effect with small problems other than the increase in the number of iterations. This strategy yielded better results for large problems than the first strategy, although not all final solutions obtained with this strategy reached the best known solution. The last strategy performed better than the former strategies. For all the test problems it yielded solutions similar to the best known solutions. As expected, the number of iterations was larger than the number of iterations required by the former strategies, although it was smaller than the strategy of having constant cycle length for lambda. By applying the third strategy what we do is to start reducing the cycle length when we require small perturbations in the multipliers. Such small perturbations are more related to the change in lambda and the value of

$Z^{\wedge}$ than to the cycle length. Hence if we reduce the cycle length at this stage of the search, it is less likely to skip any solution that would improve the search. Thus with this strategy we ensure enough iterations in the cycle of lambda when needed and we reduce them when they are not needed.

From this analysis we make the following recommendations to implement the subgradient algorithm to the problem of finding the minimum of the Lagrangean function of the area-based harvest scheduling problem:

a) Use the sum of absolute infeasibilities as a measure of the goodness of an infeasible primal solution.

b) Use a reduction factor of lambda preferably smaller than the conventional 0.5. For big problems the reduction factor should not be greater than 0.25. If a minimum value of lambda is being used as stopping rule for the search, small problems require a lambda around 10^{-5} to yield good solutions, while large problems require approximations of the order of 10^{-7}.

c) For medium-to-large size problems use the primal solution with the largest infeasibilities as the subgradient of the search at nondifferentiable points of $Z_D(u)$. This strategy might be used for small problems with more than 5 planning periods.

d) Use as an estimate of $Z^{\wedge}$ in the computation of the step size a value computed by a 15-20% reduction of the

current minimum value of $Z_D(u)$ in the search. Such percentage can be reduced after $Z_D(u)$ has minimum changes. We defined such minimum changes of $Z_D(u)$ as occurring when its value does not change by more than 0.001.

e) Use a cycle length of lambda greater than or equal to 30 for big problems and a minimum of 10 for small problems. For big problems the cycle length can be reduced slowly until reaching a minimum of 10 once $Z_D(u)$ has minimum changes.

These recommendations consider the associated effects. For instance, decreasing the reduction factor of lambda beyond some percentages has the same effect as if we increase the cycle length. Hence these recommendations include those strategies that yield good solutions with few iterations.

IMPROVEMENTS ON THE SOLUTION YIELDED BY THE SUBGRADIENT ALGORITHM

So far we have seen how by just varying some parameters in the subgradient algorithm it is possible to improve the estimation of the multipliers. However for some problems the solution obtained through the subgradient method is not within the permissible bounds of variations of harvest flow (10%), and therefore such estimates must be improved. Problems of this type are often characterized by a lack of an integer solution within the specified bounds, or they may be highly infeasible problems such as harvest scheduling problems with a large number of periods and few stands. Sometimes even for problems well formulated and with feasible solutions within the required bounds, the following procedure improves the solution obtained through the subgradient algorithm. The procedure we propose is based on the characteristics of the Lagrangean function dependent only on one dual variable. Consider the Lagrangean function:

$$Z_D(u_k) = \max \mathbf{c}' \mathbf{x} + \mathbf{u}'(\mathbf{b} - \mathbf{D}\mathbf{x}) \qquad (11)$$

where only u_k is variable and all other multipliers remain constant. This function behaves as the function depicted in figure 2, i.e. it is convex and defined by different primal solutions associated with the specific value of u_k. By taking the first derivative of $Z_D(u_k)$ with respect to u_k we obtain the direction of adjustment of u_k to reach a

better solution (smaller) for $Z_D(u_k)$. Such derivative is given by the scalar:

$$(\mathbf{b} - \mathbf{D}\,\mathbf{x}^r)_k \tag{12}$$

which corresponds to the infeasibility in the k-th constraint given the "r-th" guess on u_k. Assume we have reached the best solution through the subgradient method at the r-th iteration. Then we know the best vector $\mathbf{u}^r$ (where "r" represents the iteration for which the best solution was obtained) and the minimum value of the Lagrangean function $Z_D(u_k^r)$. Given that the direction of improvement of the function $Z_D(u_k)$ is given by $(\mathbf{b} - \mathbf{D}\,\mathbf{x}^r)_k$ we could change u_k at the "r+1" iteration such that:

$$u_k^{r+1} = u_k^r - (\mathbf{b} - \mathbf{D}\,\mathbf{x}^r)_k \tag{13}$$

would yield a better solution. Unfortunately this is not as straightforward as it looks, since although the direction of adjustment is correct, generally the new value of the multiplier "u_k^{r+1}" yields a solution $Z_D^{r+1}(u_k)$ larger than the current value at iteration "r" and with the "k-th" infeasibility $(\mathbf{b} - \mathbf{D}\,\mathbf{x}^{r+1})_k$ of contrary sign. Consider figure 2. Assume that the subgradient method yielded a solution at point A. After applying the adjustment in (13) the solution moves to point B. What the subgradient algorithm does is to weight the subgradient such that the big change caused by $(\mathbf{b} - \mathbf{Dx})$ is reduced by a step size. In our procedure we will use the points A and B as an interval of uncertainty to proceed with the search of lower

values of $Z_D(u_k)$ presumably to reach a hypothetical minimum of $Z_D(u_k)$. The search is performed by applying any search algorithm such as dichotomous search, Golden search or Fibonacci search (Bazaraa and Shetty, 1979). This search often provides smaller values for $Z_D(u_k)$ and sometimes those smaller values are associated with "better" primal solutions. The proposed procedure can be summarized in the following steps:

STEP 1. Initialize variables with the best solution obtained from the subgradient search.

STEP 2. Sort all infeasibilities in decreasing order by their absolute value. Choose the infeasibility with the largest value. The constraint associated with that infeasibility becomes constraint "k".

STEP 3. Compute the interval of uncertainty given by u_k^r and u_k^{r+1}.

STEP 4. Use bisection to determine lower values of $Z_D(u_k)$. For each guess of $u_k{}^r$ check if the associated primal solution is better. If so, update best **u** and GO TO STEP 2. Otherwise, continue the search until a permissible ϵ is reached.

STEP 5. Select the constraint with the next lowest infeasibility from the sort list in STEP 1. If there are no more constraints GO TO STEP 6. Otherwise, the selected constraint becomes constraint "k", GO TO STEP 3.

STEP 6. Terminate.

The idea of this procedure is to make small perturbations in the neighborhood of the "best" multipliers in order to find better solutions; which is especially useful when the subgradient search yields solutions with one or two large infeasibilities. It is important to mention that big changes in the primal solution might occur with changes of the order of 10^{-6} in one of the multipliers. Hence the search requires a lot of precision. The search of the primal solution is accompanied with the search for all associated primal solutions given one dual guess. For this algorithm we will always select the primal solution with the smallest sum of absolute infeasibilities.

COMPUTATIONAL RESULTS

In order to compare the performance of Lagrangean relaxation we simulated 5 forests. Example forest 1 is composed by 20 stands. Half of the area of this forest is within age class 40 and the other half is within age class 80. Example forest 2 is composed by 50 stands and its age class distribution resembles a normal distribution with a minimum age class of 10 years and a maximum of 100. Example forest 3 is composed by 70 stands. It is an overmature forest where all its age classes are above 80 years. The distribution is as follows: 50% of the area is within the 80 year age class, 25% is within the 90 year age class, 15% is within the 100 year age class and 10% is within the 110 year age class. Example forest 4 is composed of 100 stands and its age class distribution is similar to example forest 5. Example forest 5 has 136 stands and its age class distribution looks like the one depicted in figure 9. This distribution and the area of each harvest unit correspond to the Green River subbasin in the Siuslaw National Forest (Barker, 1989). The growth and yield estimations for example forests 2 and 4 were made through the growth model for loblolly pine developed by Cao et al., (1982). The rest of the yield estimates were made by using the equations of volume for Douglas fir defined in Brodie et al., (1979). Price and cost information corresponds to the reports in Fight et al., (1984) and

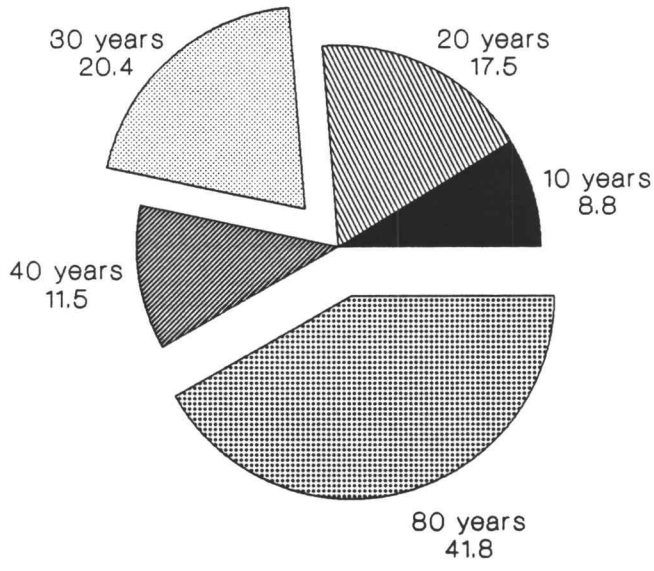


Figure 9. Proportion of age classes in example forest 5.

local information. We used four different planning horizons: 3, 5, 7, and 10 decades. The formulation corresponds to the formulation of problem P1 (figure 1) with strict even-flow constraints using all possible combinations of harvest for each stand and considering a minimum harvest age of 40 years.

The harvest scheduling problems were solved by three solution techniques: linear programming (LP), integer programming (IP) and Lagrangean relaxation (LAR). For the integer formulation we defined a maximum fractional reduction and a maximum fractional increase permitted in the harvest flows (harvest flow fluctuations). These harvest flow fluctuations were bounded as close as possible in order to maintain a fair comparison between the LAR and

IP solutions. Obviously LAR solutions provided a likely bound for these permissible harvest flow fluctuations.

The variables we compared were: differences in maximum, average, and minimum deviation in harvest flows within the Lagrangean relaxation, solution time, and objective function value. The maximum deviation is the difference between the maximum and minimum harvest as percentage of the minimum harvest. The average deviation corresponds to the sum of absolute infeasibilities as percentage of the total harvest volume. The minimum deviation is computed as the minimum infeasibility as percentage of the minimum harvest flow that yielded such infeasibility. Table 1 shows the comparative results. Linear and integer programming solutions were obtained by using the computer package MILP88. In this table the LAR solutions correspond just to the results obtained from the subgradient search. We divide our discussion of these results into three main topics: the solution time, the quality of the solution and the efficiency of our search algorithm to improve the solution.

Solution time

Given the highly combinatorial nature of the area-based harvest scheduling problem, solution time of IP increases exponentially with the number of integer variables in the problem. In our example problems, solution time from IP was considerably larger than solution time from LAR. Just

Table 1. Comparative results from linear programming, integer programming and Lagrangean relaxation for 20 example forests.

Problem No.	SIZE	No. of periods	No. of stands	No. of vars.	INTEGER SOLUTION			LP SOLUTION		LAGRANGEAN RELAXATION SOLUTION				
					Obj. fcn. value ($)	Max dev. flow (%)	Time (hrs)	Obj. fcn. value ($)	Time (hrs)	Obj. fcn. value ($)	deviation of flows (%) max.	avg.	min	Time (hrs)
1	15	3	20	80	1951	3.68	0.013	1954	0.0008	1950	- 7.50	4.83	+ 3.47	0.0049
2	23	3	50	173	3569	4.50	2.812	3567	0.0054	3547	+ 8.29	3.37	- 1.52	0.0080
3	29	3	70	280	40842	4.92	5.175	40951	0.0098	40842	+ 4.92	3.31	- 0.57	0.0126
4	32	3	100	328	5269	1.14	29.846	5277	0.0172	5269	+ 1.14	0.89	- 0.07	0.0128
5	36	3	136	436	33338	3.72	34.386	33480	0.0272	33389	- 8.93	5.21	+ 2.34	0.0238
6	26	5	20	140	1772	5.02	6.483	1771	0.0016	1758	+ 13.92	7.99	+ 2.61	0.0066
7	39	5	50	308	3704	6.82	7.944	3698	0.0166	3696	- 6.32	4.03	- 0.01	0.0133
8	49	5	70	475	a	a	a	36547	0.0312	36994	- 16.08	8.97	+ 1.08	0.0219
9	49	5	100	589	5787	5.37	34.256	5759	0.0341	5758	+ 2.32	1.54	- 0.46	0.0194
10	63	5	136	787	***	***	***	34363	0.1091	34491	- 10.01	5.22	+ 0.12	0.0371
11	44	7	20	280	1728	14.83	10.145	1787	0.0055	1708	+ 14.62	9.59	+ 2.29	0.0124
12	65	7	50	607	3846	8.74	50.368	3836	0.0328	3830	+ 7.9	3.43	- 0.24	0.0255
13	78	7	70	980	b	b	b	34701	0.0592	34231	- 43.63	9.47	+ 0.38	0.0397
14	90	7	100	1157	***	***	***	6135	0.1219	6165	+ 10.62	4.42	- 0.19	0.0561
15	104	7	136	1539	***	***	***	34726	0.2342	34659	+ 4.17	1.74	- 0.08	0.1011
16	63	10	20	480	1794	19.8	39.274	1702	0.0122	1909	+ 16.29	8.23	+ 4.19	0.0318
17	101	10	50	1122	***	***	***	3833	0.0777	3851	+ 9.20	2.76	+ 0.03	0.0825
18	94	10	70	1680	***	***	***	31372	0.1425	32956	+ 25.20	9.56	- 0.18	0.0369
19	148	10	100	2181	***	***	***	6164	0.5608	6277	+ 12.13	3.23	- 0.14	0.1957
20	157	10	136	2638	***	***	***	34619	1.0374	34557	- 6.47	3.06	+ 0.55	0.1888

*** No attempt to solve was made.

a) Infeasible with maximum deviation of flows of 15%

b) Infeasible with maximum deviation of flows of 45%

to give a rough idea about the difference in solution time between the LAR and IP solutions, an average of the time differences for all the IP problems for which we obtained a solution (see table 1) showed that IP solutions took about 1000 times longer than the time required to obtain a LAR solution. Obviously if we had considered the large problems we did not solve through IP this average would increase. LP solution time is very variable as opposed to LAR solution time which is related to our size measure. In general, solution time was shorter for LAR solutions than for LP solutions except for problems with few stands. For those problems LP solutions were always obtained faster than LAR solutions. The reason for this exception is the lack of a convergence criterion in the subgradient algorithm. For small problems the subgradient algorithm finds the best solution faster than LP. However, the termination criterion of the subgradient algorithm is set by a minimum value that lambda can take. Hence although the subgradient algorithm converges rapidly for these problems it takes additional time to perform the rest of the iterations until it reaches the minimum lambda that will stop the search.

The selection of an adequate "convergence criterion" is an important factor that determines the efficiency of the subgradient algorithm. Such criterion can be a minimum value of lambda or a minimum step size. However, any of these criteria is highly problem dependent. The

determination of a stopping criterion for the subgradient algorithm is beyond the scope of this study. However we believe that the answer is probably in the pattern that the primal solutions take after the "best" solution has been reached. Such a pattern looks cyclic for small problems and like a pattern in "chaos" for big problems (see figure 5).

Quality of solutions

LAR solutions were always very close to the optimal integer solution. As table 1 shows, for two problems the LAR solutions were exactly the same as the "optimal integer solution". What is noteworthy is that for problems with three periods, the maximum difference in the objective function value obtained by LAR and IP solutions was smaller than 0.15% and the maximum flow fluctuation was always less than 10%.

Because of the way some IP problems were formulated (wide bounds in the harvest flow fluctuations) the LAR solutions yielded tighter bounds for the harvest flows than the IP solutions. Problems 7, 9 ,and 12 are examples of this kind of solution. For these problems the IP solutions exploited the larger flexibility in the flows and yielded a larger objective function value. Obviously those solutions are superoptimal and more infeasible than LAR solutions. On the contrary observe the IP solutions for problems 1 and 2. These solutions yielded tighter bounds

for the harvest flows and larger objective function value than the LAR solutions, which simply supports our strategy of choosing the LAR solution with smallest infeasibilities as the best solution. Obviously for these problems the LAR approach was not able to find the optimal integer solution. All the problems solved for an optimal IP solution had a difference in the objective function value between LAR solutions and the IP solutions no greater than 2% and on average the harvest flow deviation never was greater than 8%. Following duality theory we would expect that LAR solutions would yield an objective function value between the one yielded by IP and the one yielded by LP, which obviously would be more infeasible than the IP solution. However, solutions were variable but always close to the optimal integer solution. An example of this expected result is the solution obtained for problem 5.

Table 1 shows clearly the close relationship between the objective function value obtained with LP and IP. The solutions are so close that for all 3-period problems the LP solution yielded only 2 stands with fractional values for the treatments. Obviously this is a product of the way the LP is formulated since each problem has NP + NS binding constraints (NP represents the number of periods and NS the number of stands). Hence each LP can have a maximum of NP stands with fractional values (with two treatments per stand). In most of our example problems this maximum

number of stands with fractional values for the treatments was always NP - 1, except for those IP infeasible problems (e.g. problems 8, 13, and 18) where the number decreased but the number of fractional treatments per stand increased.

Theoretically as we increase the number of integer variables the IP solution gets closer to the LP solution (Bertsekas 1982). Observe the results for example forest 1. Considering just a 3 period problem the difference between the LP and IP objective function values is just 0.15% while for the five period problem such difference is reduced to 0.05%. However when considering the 7 period problem the difference is greater and even the fluctuations in the harvest flows increase. This behavior is a result of the "natural integer infeasibility" of the problem, i.e. we are considering too many periods for a few stands. Consider problems 7 and 8. They have the same number of periods and problem 8 has more stands and a larger number of treatments per stand than problem 7. However, the IP solution for problem 8 is more infeasible than problem 7. This difference might indicate that the latter statement about the relationship between number of periods and number of stands is wrong. But this difference is a product of a second source of "natural integer infeasibility" of the problem, namely the original age class distribution of stands in the forest. The distribution is so diverse in example forest 2 that there is no problem in meeting the

flow constraints. However, it is so restricted for example forest 3 that a larger deviation in the flows is required in an integer solution to meet those constraints. The natural integer infeasibility of a problem is not important in LP since it can take fractions of treatments per stand. But this is not possible if integer solutions are required. As can be observed in table 1 this natural integer infeasibility of the problem causes the objective function from integer solutions to deviate more from the LP objective function (larger duality gap). The LAR approach always offers a solution for such integer infeasible problems, which is generally superoptimal (considering the LP solution) and infeasible. For instance in problem 8 the integer solution reached by LAR probably corresponds to the optimal solution, since the maximum infeasibility is 16%, and we did not obtain a feasible solution by IP within a 15% deviation. In addition as can be observed there is not much difference between the LAR and LP objective function values for this problem; which suggests that the integer solution obtained by LAR might be close to the best integer solution. Integer infeasible problems cause the subgradient algorithm to converge rapidly, which as we saw is not good for the search. Sometimes the convergence yields the best attainable integer solution, but sometimes the search is deviated. For this kind of problem we can apply the proposed search algorithm to improve the solutions.

Search procedure

Our search procedure was applied to problems 8, 11, 13, 14, 16, 18, and 19 (see table 1). Results on these searches are shown on table 2.

Table 2. Solutions of the search algorithm for selected problems.

Problem No.	Solution after search algorithm				
	Obj. fcn. value ($)	deviation of flows (%) max.	avg.	min	Time (hrs)
8	36994	- 16.08	8.97	+ 1.08	0.011
11	1708	+ 14.62	9.59	+ 2.29	0.013
13	34531	- 43.63	6.47	+ 0.38	0.029
14	6124	- 5.74	2.98	- 0.19	0.083
16	1769	+ 13.42	7.74	+ 1.34	0.047
18	32456	+ 25.20	9.56	- 0.18	0.026
19	6214	+ 4.72	2.37	- 0.14	0.109

As can be observed the procedure does not always find a better solution. For the mentioned set of problems only the solutions to problems 8, 11, and 18 were not improved. The solution of problem 19 was dramatically improved by reducing the maximum deviation of harvest flows up to 4.7%. In this case the diversity of age classes and the large number of treatments available in example forest 4 helped

in reaching a better solution. Presumably such a better integer solution does not exist for problems 8, 11, and 18, or if it exists it can not be found in the dual space. The time of our search algorithm depends on the problem size and as can be observed the procedure is very slow. To obtain an improved solution of problem 19 it took the equivalent of 0.6 times the time taken by the subgradient search to find a "good" solution. The time could be reduced if we avoid the first guesses in the search procedure, since we expect that only small changes in the multipliers will have an effect in finding a better solution.

CONCLUSIONS

Lagrangean relaxation can be applied to other sets of constraints in an area-based harvest scheduling problem. However, the nature of the approach and the solutions obtained limit its use to constraints that meet the following conditions:

a) It is not required that the constraints are satisfied strictly. For instance, when we include adjacency restrictions we require that those constraints are satisfied strictly and it is very likely that LAR will not yield a solution that guarantees that all adjacency constraints are met. Hence LAR should be applied when

small deviations from the requirement vector are not harmful in the solution. Such constraints might be wildlife habitat constraints, sediment production constraints or any other harvest constraint.

b) Constraints should be dualized so the resulting Lagrangean problem is easier to solve.

c) The number of constraints dualized affects the rate of convergence. Many authors have emphasized that the success of algorithms to solve the Lagrangean problem depends on the set of constraints we dualize. We believe that this depends not only on how wisely we dualize the constraints but also on the number of constraints dualized. When we dualize many constraints the convergence is better. This is simply because there are more dual variables that define one single primal solution. Hence there are fewer nondifferentiable points in the Lagrangean function and it is less likely that the search is deviated. However, by dualizing too many constraints the convergence is faster which might be harmful. Additionally the search for several dual variables complicates the computation of the Lagrangean function. Hence good results from Lagrangean relaxation will depend on the type and quantity of constraints dualized.

d) Constraints with large technical coefficients should be scaled. Better solutions are obtained if there are no large disproportions among the infeasibilities of

the relaxed constaints.

Lagrangean relaxation is a very useful technique to obtain "good" solutions of area-based harvest scheduling problems. As we have seen even optimal solutions can be obtained through the this procedure without devoting much time and resources in obtaining integer solutions. Additionally the technique is comparatively more efficient as the size of the problem increases and always provides a "good" infeasible solution when there is no feasible integer solution within the specified bounds. The main problem in the technique is the multiplier guess. However, by using all the recommendations stated in the fourth section, the subgradient algorithm always yields very good solutions whether a feasible integer solution exists within the bounds of the harvest flow deviations or not. For some problems there might be poor convergence. For such problems our improvement procedure indeed yields better solutions only when the problem is large. Since the search is slow our procedure should be used only in cases were a poor convergence is reached through the subgradient search.

LITERATURE CITED

ALLEN, E., R. HELGASON, J. KENNINGTON, and B. SHETTY. 1987.

A generalization of Polyak's convergence result for subgradient optimization. Math. Programming 37(3):309-317.

BARKER, B.R. 1989.

Utilizing scheduling and network analysis program (SNAP) as a forest plan implementation tool on the Siuslaw National Forest. MS thesis, College of Forestry. O. S. U. Corvallis Oregon. 76 p.

BAZARAA, M.S. and C.M. SHETTY. 1979.

Nonlinear programming: Theory and algorithms. John Wiley & Sons. 560 p.

BERCK, P. and T. BIBLE. 1984.

Solving and interpreting large-scale harvest scheduling problems by duality and decomposition. For. Sci. 30(2):173-182.

BERTSEKAS, D.P. 1982.

Constrained optimization and Lagrange multipliers methods. Academic Press. London. 395 p.

BRADLEY, G.H. 1971.

Transformation of integer programs to knapsack problems. Discrete Mathematics 1:29-45.

BRODIE, J.D., H.C. BLACK, E.J. DIMOCK II, C. KAO, and J.A. ROCHELLE. 1979.

Animal damage to coniferous plantations in Oregon and Washington:part II an economic evaluation. Forest Research Laboratory, O. S. U. Corvallis OR. Research Bull No. 26. 22p.

CAO, Q.V., H.E. BURKHART, and R.C. LEMIN. 1982.

Diameter distributions and yields of thinned loblolly pine plantations. School of For. and Wildlife Res., V.P.I. and State Univ. Publ. FWS-82,62p.

DREYFUS, S.E. 1957.

Computational aspects of Dynamic Programming. Opns. Res. 11(4):399-417.

ELDRED, P. 1987.

Evaluation of a shadow price search heuristic as an alternative to linear programming or binary search for timber harvest scheduling. Unpublished MS Thesis, College of Forestry. Oregon State University. 64 p.

EVERETT III, H. 1963.

Generalized Lagrange multipliers method for solving problems of optimum allocation of resources. Opr. Res. 11(3):399-347.

FIGHT, R.D., C.B. LEDOUX and T.L. ORTMAN. 1984.

Logging costs for management planning for young-growth coast Douglas-fir. USDA For. Serv. Pac. Northwest For. and Range Exp. Stn. Gen. Tech. Rep. PNW-176. 10p.

FISHER, M.L. 1981.

The Lagrangian relaxation method for solving integer programming problems. Mgmt. Sci. 27(1):1-18.

FISHER, M.L. 1985.

An applications oriented guide to Lagrangian relaxation. Interfaces 15(2):10-21.

FISHER, M.L., W.D. NORTHUP and J.F. SHAPIRO. 1975.

Using duality to solve discrete optimization problems. Math. Programming study No. 3. pp 56-94.

FISHER, M.L., JAIKUMAR, R. and VAN WASSENHOVE L. 1980.

A multiplier adjustment method for the generalized assignment problem. Dec. Sci. Working paper. Univ. of Pennsylvania.

GAVISH, B. and H. PIRKUL. 1985.

Zero-one integer programs with few constraints:Lower bounding theory. European Journal of Opr. Res. 21(2):213-224.

GEOFFRION, A.M. 1974.

Lagrangean relaxation for integer programming. (Ed. M. L. Balinski), North-Holland/Amer. Elsevier. Mathematical Programming Study No. 2. pp 82 -114.

GLOVER, F. 1975.

New results on equivalent integer programming formulations. Mathamtical Programming 8(1):84-90.

GOFFIN, J.L. 1977.

On convergence rates of subgradient optimization methods. Math. Programming study No. 13. pp 329-347.

HELD, M. and R.M. KARP. 1970.

The traveling salesman problem and minimum spanning trees. Opr. Res. 18(4):1138-1162.

HELD, M. and R.M. KARP. 1971.

The traveling salesman problem and minimum spanning trees: part II. Math. Programming 1(1):6-25.

HELD, M., P. WOLFE, and H.P. CROWDER. 1974.

Validation of subgradient optimization. Math. Programming, 6(1):62-88.

HOGANSON, H.M. and D.W. ROSE. 1984.

A simulation approach for optimal timber management scheduling. For. Sci. 30(2):220-238.

IBARAKI, T. and N. KATOH. 1988.

Resource allocation problems. Algorithmic approaches. Found. of Computing Series. MIT Press. Cambridge, Massachusetts. 229 p.

JOHNSON, K.N. and H.L. SCHEURMAN. 1977.

Techniques for prescribing optimal timber harvest and investment under different objectives: Discussion and synthesis. For. Sci. Monograph No. 18. 31 p.

LORIE, J.N. and L.J. SVAGE. 1955.

Three problems in rationing capital. J. of Business. 28(4):229-239.

MEYER, G.G.L. 1987.

Convergence of relaxation algorithms by averaging. Math Programming 40(2):205-212.

PAREDES VELOSO, G.L. and J.D. BRODIE. 1986.

Efficient specification and solution of the even-aged rotation and thinning problem. For. Sci. 33(1):14-29.

POLYAK, B.T. 1969.

Minimization of unsmooth functionals. USSR Computational Mathematics and Mathematical Physics, 9(3):14-29

SHAPIRO, J.F. 1971.

Generalized Lagrange multipliers in integer programming. Opr. Res. 19(1):68-76.

SHAPIRO, J.F. 1979.

A survey for Lagrangean techniques for discrete optimization. Ann. Discrete Math. Vol. 5. pp. 113-138.

SHOR, N.Z. 1964.

On the structure of algorithms for the numerical solution of optimal planning and design problems. Dissertation Cybernetics Institute, Academy of Sciences USSR.

TSENG, P. and D.P. BERTSEKAS. 1987.

Relaxation methods for linear programs. Math. Opr. Res. 12(3):569-596.

ADJACENCY CONSTRAINTS IN HARVEST SCHEDULING: AN AGGREGATION HEURISTIC

by

Juan M. Torres Rojo

and

J. Douglas Brodie

ABSTRACT

A heuristic for adjacency constraint aggregation is proposed. The heuristic is composed of two procedures. Procedure one consists of identifying harvesting areas for which it is not necessary to write adjacency constraints. Procedure two consists of writing one adjacency constraint for each one of the harvesting areas not identified in procedure one. 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 conjecture". The aggregated constraints present fewer variables per constraint than the aggregator described by Meneghin et al., (1988) 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 type 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 Meneghin et al., (1988).

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) can not be harvested in the same period.

Adjacency constraints add a large number of rows to any integer programming (IP) or mixed integer programming (MIP) formulation of 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 to fit those large complex formulations into standard linear programming (LP) or IP solution packages. More important potential applications of the aggregated constraints consist of improving the efficiency of

heuristics to solve special IP problems such as the multiconstrained knapsack problem, or when the solution to the IP 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 intensively studied because of its simplicity and the aim of deriving aggregators with manageable coefficients able to improve the computational efficiency of current solution algorithms (Fishburn and Kochenberg, 1985). There exists some reasonable concerns about the improvement in computational efficiency by using aggregated constraints (Onyekwelu, 1983). However, there is no doubt about its usefulness in the solution of highly structured IP problems (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 not only increase the number of variables but 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 Kochenberg (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 6 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 an LP or IP 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 which can be easily implemented in 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 Conjecture" (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 type 2 constraints before constructing the aggregated constraint.

In the next section we present our heuristic for adjacency constraints aggregation and the rules to compute the aggregators. Then 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).

A HEURISTIC FOR CONSTRAINT AGGREGATION

Our heuristic is based on recognizing the following characteristics of the adjacency constraints.

1) 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.

2) 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 further reduced.

Based on these observations, our heuristic consists of two steps. The first step, called 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, called 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 figure 10. 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 figure 2. Recalling that each X in figure 11 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, figure 11, $NR_1=3$ and $NC_1=3$. If for the i-th row $NR_i=NC_i$ it implies that area i can be identified i.e. its adjacency relations 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.

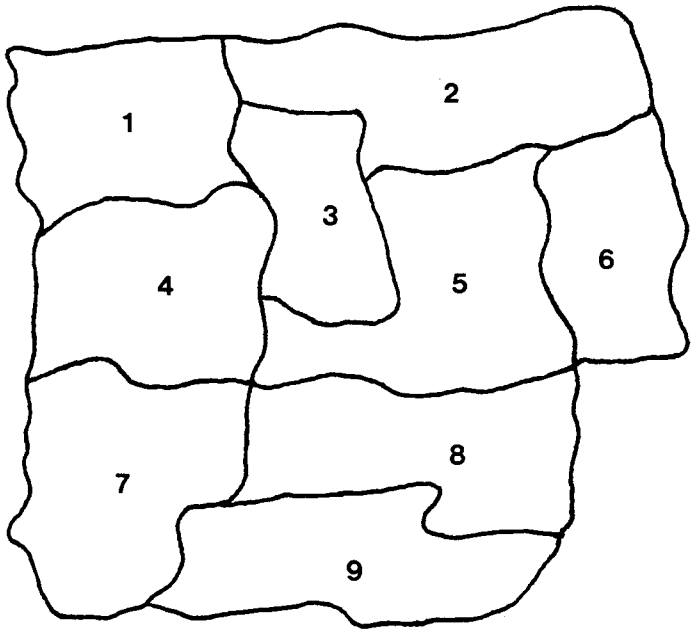


Figure 10. Example forest with 9 harvesting areas.

AREA	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	

Figure 11. Adjacency matrix for pattern in figure 10.

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 figure 11, $NR_1 = NC_1 = 3$, so row 1 might be eliminated and the rows 2, 3, and 4 will still keep the adjacency relations of area 1 and these areas (namely 2, 3, and 4). If row 1 is eliminated (i.e. area 1 has been identified), $NR_2=4$ and $NC_2=3$, so row 2 can not be eliminated and since $NR_3=NR_4=4$ and $NC_3=NC_4=3$ neither row 3 nor row 4 can be eliminated. However $NR_5=NC_5=5$. Therefore, row 5 can be eliminated. Following the same pattern row 6 can not be eliminated but row 7 can. Thus, rows 8 and 9 can not be eliminated.

Note that in order to obtain the relationships $NC_2=NC_3=NC_4=3$ 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 figure 11 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 figure 10. An adjacency constraint for area 9 that describes all the adjacency relationships between area 9 and its surrounding areas is :

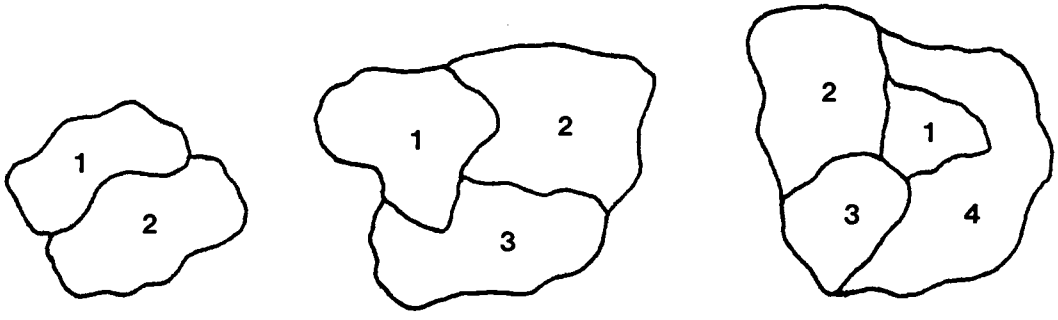
$$X_7 + X_8 + X_9 \leq 1 \qquad (1)$$

$$X_i \in \{0, 1\}$$

where:

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

which is called a triplet. Figure 12 shows 3 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 figure 12 pattern (a).



(a) (b) (c)

Figure 12. Three different patterns where is possible to write "type 1 constraints"

To write an adjacency constraint for area 1 under pattern (a) we have a pair:

$$X_1 + X_2 \leq 1$$

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) (figure 12). For cases where type 1 inequalities do not work, our procedure consists of penalizing the coefficients of the areas adjacent to the area for which we are writing the constraint. Consider again the pattern depicted in figure 10. If we want to write an adjacency constraint for area 1, the constraint:

$$X_1 + X_2 + X_3 + X_4 \leq 1 \qquad (2)$$

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:

$$5\ X_1 + 3\ X_2 + 4\ X_3 + 2\ X_4 \leq 5 \qquad (3)$$

keeps the desired adjacency relations and considers all the areas adjacent to area 1. Constraint 3 was partially constructed following the idea of penalizing each one of the harvesting areas that violate other constraints. The penalization procedure is simple and better understood through an example. Hence next we will define some notation and the rules of the procedure, 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 non-colorable.

The rules of procedure 2 for penalizing variables in the aggregated adjacency constraint are:

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 in the aggregated constraint adds a penalty to the reference area if:

a) It is the last variable to enter (i.e. the fourth or less if the reference area has less than four adjacent areas). Areas that are colorable are not considered in the counting.

b) It is not the last one 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).

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 non-colorable 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 one 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". Then such 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 the entering variable say 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 ———> C ———> 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 area(s) enters the constraint without penalty and the reference area is penalized just once.

7. The set of harvesting areas (variables) in each constraint has to be different in each aggregated constraint. Otherwise the constraint is considered redundant and is dropped. Likewise, the set of harvesting

areas in a given constraint can not be a subset of another set of harvesting areas in any other constraint.

EXAMPLE

Consider that we want to write aggregated adjacency constraints for the pattern depicted in figure 10. 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 last section we know that we do not need to write adjacency constraints for areas 1, 5, and 7. Then we 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 figure 10. 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:

$$X_7 + X_9 \leq 1 \qquad (4)$$

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 because it has an adjacent area in constraint (4) (rule 4) namely area 7 (X_7). Likewise, following rule 4, X_7 is penalized because of its adjacency relation with area 8 and 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, once because X_8 enters (rule 3a) and the second time because of the adjacency relations between areas 7 and 8. Figure 13 shows this penalization procedure which yields the following aggregated constraint:

$$2\ X_7 + 2\ X_8 + 3\ X_9 \leq 3 \qquad (5)$$

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. doubles, triplets and quadruplets (Meneghin et al., 1988), these constraints always have smaller coefficients than our aggregators.

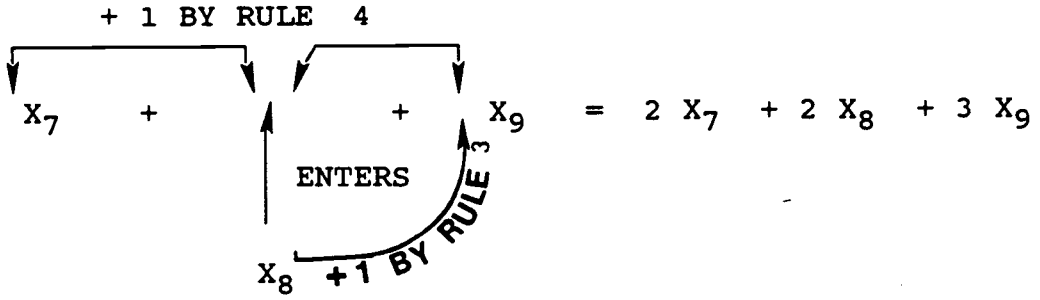


Figure 13. Transitivity effect to compute the penalties.

Suppose that we want to write an adjacency constraint for area 1 (from procedure 1 we know that we do not need 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 above, the second aggregation level yields:

$$3\ X_1 + 2\ X_2 + 2\ X_3 \leq 3 \qquad (6)$$

Note that in this case the second penalty to the reference area X_1 followed rule 3b not rule 3a as the last example. The third aggregation level has a transitivity effect which will be described in two steps. First following rule 4 we enter X_4 with a penalty because of its adjacency relation with X_3 and 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:

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

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 the adjacent area 2, adding the corresponding penalty to the reference area (rule 5). Thus the final aggregator for area 1 is:

$$6\ X_1 + 3\ X_2 + 4\ X_3 + 2X_4 \leq 6 \qquad (7)$$

Notice that this aggregator violates the infeasibility 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:

$$2\ X_1 + 2\ X_3 + 3\ X_4 \leq 3 \qquad (8)$$

For the third aggregation level we have the transitive effect which is described in two steps. In the first step X_5 enters in 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:

$$2\ X_1 + 3\ X_3 + 4\ X_4 + 2\ X_5 \leq 4 \qquad (9)$$

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 in 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:

$$3\ X_1 + 4\ X_3 + 5\ X_4 + 2\ X_5 \leq 5 \qquad (10)$$

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

$$3\ X_1 + 4\ X_3 + 6\ X_4 + 2\ X_5 + X_7 \leq 6 \qquad (11)$$

Note that this aggregated constraint violates the requirement that X_3 and X_5 can not 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:

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

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:

$$X_5 + 2\ X_7 + 4\ X_8 + 2\ X_9 \leq 4 \qquad (12)$$

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

$$2\ X_1 + 2\ X_2 + 3\ X_3 \leq 3 \qquad (13)$$

For the third aggregation level we enter X_4. Variable X_4 enters in 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:

$$4\ X_1 + 3\ X_2 + 5\ X_3 + 2\ X_4 \leq 5 \qquad (14)$$

The fourth level of aggregation has a transitivity effect which will be 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 because X_5 has adjacent areas in constraint (14) (rule 4). 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:

$$4\ X_1 + 4\ X_2 + 7\ X_3 + 3\ X_4 + 3\ X_5 \leq 7 \qquad (15)$$

In the second step we consider the transitivity effect. 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:

$$6\ X_1 + 5\ X_2 + 9\ X_3 + 4\ X_4 + 3\ X_5 \leq 9 \qquad (16)$$

As the reader can verify, the violation incurred in

constraint (11) 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 3 shows the aggregated constraints for each one of the areas in figure 10. 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.

Table 3. Aggregated constraints for each area in figure 10 considering 4 non-colorable areas per constraint.

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

* Constraints needed to explain all the adjacency relations.

Thus for describing the adjacency relations of the pattern depicted in figure 10 we just need the six constraints marked with an asterisk in table 3, 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 4 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 violate 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 constraint; 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 figure 10 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, 3, and 4 consider area 1, then the constraint for area 1 is not necessary since its adjacency relations with its surrounding areas is implicit in the constraints for areas 2, 3, and 4. In other words, the number of areas adjacent to area 1 equals the number of areas adjacent to area 1 left (non-identified) 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 if $NR_i = NC_i$ 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 non-identified.

In section 2 we described the rules of procedure 2 to compute aggregated adjacency constraints when the reference area has up to 4 adjacent areas (recall that colorable areas are not considered in the counting). However, it is very likely that the reference area has more than 4 (non-colorable) adjacent areas. We can apply the same rules when the reference area has more than 4 (non-colorable) 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 figure 10, 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 figure 10. 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 which span the same period of time, where some events can occur simultaneously but others can not (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 areas (non-colorable) 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 minimum 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 (non-colorables) 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 non-colorable areas, i.e. those areas adjacent not only to the reference area but also adjacent to at least one more area already included in the constraint. 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 non-colorable areas we should include in the aggregated constraint, we will use two examples. First, consider a pattern of harvesting areas similar to a checkerboard (figure 14). This pattern has a chromatic number of two[1/],

1	2	3
4	5	6
7	8	9

Figure 14. Checkerboard pattern of an example forest.

which means that only two colors are sufficient to color each area and no two adjacent areas are the same color.

1/ Only the infinite checkerboard has a chromatic number of 2 since technically the area around the checkerboard has to have another 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. Translating this idea into colors means that each one of the areas adjacent to the reference area in each constraint can have the same color.

Following the rules of procedure 2, we can construct the aggregated adjacency constraints for each one of the areas in this example. These aggregated adjacency constraints are shown in table 4. In all cases we did not stop adding areas to the aggregated constraint once 2 areas were already in the aggregated constraint (recall that the chromatic number of this example is two) 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 non-colorable areas in the constraint (i.e. two areas not only adjacent to the reference area but also adjacent to each other) and two non-colorable areas are more than the minimum required for a pattern with chromatic number of two.

A second example might help to clarify this idea. For our example problem in figure 10 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

Table 4. Aggregated constraints for each area in figure 14 considering 1 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	$X_6 + X_8 + 2X_9 \leq 2$

but also adjacent to any other area already in the constraint (i.e. a non-colorable area) we stop entering non-colorable areas and we just enter areas adjacent

exclusively to the reference area (colorables). When we enter the first non-colorable area in the constraint we automatically have two non-colorable areas in the aggregated constraint and for the pattern with chromatic number of three this is the minimum required.

Following the rules of procedure 2 we computed the adjacency constraints for each area in figure 10 stopping the addition of non-colorable areas once the first non-colorable area is added to the constraint. Table 5 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 redundancy and a violation of rule 7 (be redundant).

Table 5. Aggregated constraints for each area in figure 10 considering 2 non-colorable areas per constraint.

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

Likewise, observe that for area 2 we could write a constraint similar to 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; for area 3, we did not include the adjacent areas 2 and 5 and in this way many other adjacent areas were not included in other constraints. The reason to avoid these areas in the pattern with chromatic number of three is because we just need two non-colorable areas in each aggregated constraint, regardless if the reference area has more than two non-colorable adjacent areas. In the constraint for area 7 we included a fourth area, namely area 4, given that this area is adjacent only to the reference area 7 but is 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. The principles of graph theory

assert the simplicity of proving the sufficiency of five colors for coloring any map (Saaty and Kainen, 1977).

However there is no analytical proof[2/] of the so called "Four color conjecture". This conjecture 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 the "Four color conjecture" we can state that at least 3 areas adjacent (recall that only non-colorable areas are considered in the counting) to the reference area would be necessary to identify the adjacency relations (aggregated constraints) of each reference area.

The four color conjecture provides an upper bound for the minimum number of colors sufficient to color a map, and when the pattern is complicated it is likely that such an upper bound actually corresponds to the chromatic number. Observe what could happen if we use the chromatic number to define the minimum number of non-colorable areas in an aggregated constraint. In the case of a chromatic number of two (figure 14) if we do not include any of the constraints (see table 4) all the adjacency relations are kept. Thus, using procedure 1 we can eliminate constraints

---

2/ Appel and Haken (1976) used simulation to test 1936 special cases to prove the four color conjecture. However, this proof has not been completely accepted.

1, 3, 5, 7 and, 9 and the adjacency relations are kept. However, consider table 5. In this case the aggregated adjacency constraints were computed considering just two non-colorable 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 can not eliminate arbitrarily any constraint in this case. Consider again the pattern in figure 10. 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 non-colorable areas in the constraint, then we end up with the constraints shown in table 5. Note that the rules of procedure 2 assume the pattern is five colorable. In this case, as the reader can verify in table 3, we can eliminate arbitrarily any constraint and the adjacency relations are kept.

By increasing the number of non-colorable areas in the aggregated constraint beyond the chromatic number we have a better "identification" of the adjacency relations that 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 conjecture 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 4 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 non-colorable areas per constraint are sufficient in each aggregated constraint (when the reference area has more than 3 non-colorable areas). However, if we apply procedure 1 there is a possibility of violating some adjacency relationships. Hence we recommend to use four non-colorable areas when possible to write the aggregated adjacency constraints.

In order to identify any difference in the number of aggregated constraints computed using different numbers of non-colorable 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 twelve adjacent areas to only one area. We used 4, 5, and 6 as coloring numbers, i.e. 3, 4, and 5 as number of non-colorable 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 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 (i.e. avoid eliminating some constraints eliminated in procedure 1) 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 (the rules described for procedure 2 correspond to a coloring number of 5) 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 four there is a possibility of violating adjacency relations for complicated patterns, while by using larger coloring numbers we reduce that possibility, although we increase the number of variables per constraint and the size of the coefficients.

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:

$$\sum_{j=1}^{m} X_{ij} = 1 \quad (i=1, 2, \ldots, n) \quad m, n \geq 2$$

$$X_{ij} \in \{0, 1\}$$

The common aggregator for this system of equations is given by:

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

where the $a_1, \ldots, a_n$ is a superincreasing sequence of "positive integers" that satisfy:

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

However this aggregator yields values for a_i that increase rapidly (Fishburn and Kochenberg, 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 figure 10. All the conventional adjacency constraints that include area 1 are:

$$X_1 + X_2 \leq 1 \tag{18}$$

$$X_1 + X_3 \leq 1 \tag{19}$$

$$X_1 + X_4 \leq 1 \tag{20}$$

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

$$2\,X_1 + X_2 + X_4 \leq 2 \tag{21}$$

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:

$$\begin{aligned} X_2 + X_3 &\leq 1 \\ X_3 + X_4 &\leq 1 \end{aligned} \tag{22}$$

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 non-colorable areas.

COMPARISONS

In order to compare the performance of our algorithm we formulated adjacency constraints for 5 examples by using two other algorithms. The first algorithm was the conventional formulation which 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 figure 14 the constraints that involve area 1 are:

$$X_1 + X_2 \leq 1$$

$$X_1 + X_4 \leq 1$$

Thus for the pattern in figure 14 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 3 basic steps. 1) Identify all type 1 constraints. This step is basically to identify all the doubles, triplets or quadruplets. 2) 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. 3) 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 sort of 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 figure 14 and following this procedure to formulate adjacency constraints we need just 4 constraints to describe all the adjacency relations. Those constraints correspond to inequalities 2, 4, 6, and 8 in table 4.

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 above described and our heuristic. Then, the number of constraints used in each case was counted. Table 6 shows these results.

Table 6. Number of adjacency constraints required for 3 different algorithms in some example problems.

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

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 4 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) 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 a lesser number of variables per constraint than Meneghin's constraints, although the coefficients obtained with our

heuristic were often larger compared with Meneghin's algorithm. We did not write any computer code to compute the adjacency constraints 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 the conventional procedure but it is if compared to Meneghin's, 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 4 non-colorable areas in each constraint (when the reference area has at least four non-colorable 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 LP or IP 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 when we 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 a way to reduce adjacency constraints in area-based formulations of harvest scheduling problems. It can be applied to heuristics or optimal solution algorithms to reduce storage limitations or to reduce solution time for highly structured IP problems (Kannan, 1983), which are the main problems in area-based formulations of the harvest scheduling problem.

LITERATURE CITED

APPEL, K.I. and W. HAKEN. 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 G.A. KOCHENBERG. 1985.

Aggregating assignment constraints. Nav. Res. Log. Quart. 32(4):653-663.

GROSS, T.E. and D.P. DYKSTRA. 1988.

Harvest scheduling with nonadjacency constraints. In Proceedings, Society of American Foresters National Convention, October 16-19 1988. Wash. D. C. pp 310-315.

KANNAN, R. 1983.

Polynomial time aggregation of integer programming problems. J. Ass. of Computing Machinery. 30(1):133-145.

MENEGHIN, B.J., M.W. KIRBY, and J.G. JONES. 1988.

An algorithm for writing adjacency constraints efficiently in linear programming models. The 1988 Symp. on Systems analysis in Forest Resources. March 29-April 1 1988. Gen. Tech. Rep. RM-161 Rocky Mtn For. & Ran. Exp. Stn., Ft. Collins Colorado.

NELSON, J., J.D. BRODIE and J. SESSIONS. 1988.

Integrating short term spatially feasible harvest plans with long term harvest schedules using Monte-Carlo integer programming and linear programming. In: The 1988 Symp. Syst. Analysis on For. Res. (B. Kent and L. Davis Ed.) U.S.D.A. For. Serv. Gen. Tech. Rep. RM-161. pp. 224-229.

ONEYKWELU, D.C. 1983.

Computational viability of a constraint aggregation scheme for integer linear programming problems. Opr. Res. 31(3):795-801.

PADBERG, M.W. 1972.

Equivalent knapsack-type formulations of bounded integer linear programs: an alternative approach. Nav. Res. Log. Quart. 19(4):699-708.

RINGEL, G. 1959.

Farbungsprobleme auf Flachen und Graphen. Mathematische Monographien Numer 2. Veb Deutscher Verlag der Wissenschaften. Berlin, 132 p.

SAATY, T.L. and P.C. KAINEN. 1977.

The four-color problem. McGraw-Hill Inc. Great Britain. 217 p.

SESSIONS, J. 1988.

User's guide for Scheduling and Network Analysis Program (SNAP). Draft U.S.D.A. For. Serv., Division of Timber Mgmt., Portland, Oregon. 256 p.

WOOD, D.C. 1969.

A technique for colouring a graph applicable to large scale timetabling problems. Computer Journal 12:317-319.

THE USE OF RELAXATION TO SOLVE THE HABITAT DISPERSION PROBLEM

by

JUAN M. TORRES-ROJO

J. DOUGLAS BRODIE

and

JOHN SESSIONS

ABSTRACT

A composite relaxation approach is used to solve area-based harvest scheduling problems with flow, wildlife habitat and adjacency constraints (the habitat dispersion problem). The solution strategy consists of dualizing the harvest flow and wildlife constraints into the objective function and forming a surrogate constraint with the adjacency restrictions. The original problem is vastly reduced and the resulting relaxed problem is easy to solve. The approach was tested with several example problems. It quickly yielded "very good" integer solutions for harvest scheduling problems with any combination of flow, wildlife habitat or adjacency constraints. Results showed that in the long run wildlife habitat and adjacency constraints individually combined with flow constraints have little

effect in reducing the harvest levels lower than the long term sustained yield. However all three constraint sets in the same problem can reduce harvest levels from 25-35 percent below the long term sustained yield.

INTRODUCTION

Wildlife habitat integration with timber management has become one of the main concerns in modern multiple use management. This concern has grown in importance not only because of the ecological role of wildlife within the forest but also to more accurately evaluate the potential economic benefits and costs generated from multiple use management. Traditionally wildlife habitat requirements within a geographic area have been modeled by assigning predeterminated percentages of cover to certain age classes (Thompson et al., 1973; Mealey et al., 1982). Thus forage requirements for big game animals are modeled by assigning a percentage of the area covered by recently regenerated areas. Additional requirements for hiding can be modeled by assigning a minimum coverage of middle-aged timber and thermal requirements by ensuring a coverage of mature timber or old growth. Through this strategy conventional strata-based harvest schedules provide adequate quantities of any required age class or forest vegetation. However, the distribution of harvest units in most cases does not satisfy the wildlife spatial needs of cover and edges (Mealey et al., 1982). In order to provide these needs the harvest units have to follow special harvest patterns which guarantee that the diversity of ages and species is adequately scattered for wildlife use. The harvest scheduling problem that considers spatial concerns for

wildlife habitat managment is known as the "habitat dispersion problem" (Mealey et al., 1982; Weintraub et al., 1988).

Two approaches for considering spatial concerns in wildlife habitat management have been proposed. The oldest one permits the incorporation of spatially feasible harvest alternatives in linear programming formulations, since it recognizes spatial objectives for groups of harvest units (Mealey et al., 1982). In this approach, the spatial characteristics of each harvest choice are embedded in each decision variable. Hence net returns for each harvest choice are difficult to measure and solutions are limited because of the high risk of underspecification of formulations given the large number of variables included. The second approach has been through the inclusion of adjacency constraints (Weintraub et al., 1988). These constraints impose a minimum age difference between adjacent regenerated harvest units. The age difference has been called the "exclusion period" and is often set to the length of the planning period, although it can be variable (Gross and Dykstra, 1988). Through this approach foraging characteristics and cover and thermal requirements of species can be modeled by defining target age class distributions. Additional requirements can be included by specifying size and shape of each harvest unit, driving their spatial distribution by adjacency constraints.

Spatial concerns modeled through adjacency restrictions have been used not only for management of wildlife resources but also to manage recreation areas, lakes, riparian zones for the protection of fisheries, and to regulate sediment production (Bare et al., 1984). Several techniques to solve harvest scheduling problems with adjacency restrictions have been proposed. Optimal solutions to this problem can be obtained by integer or mixed integer programming (Kirby et al., 1986). However, given the large number of harvest units to be considered in a forest plan and the exaggerated number of adjacency constraints required to model spatial requirements, optimal solutions have been obtained for very small problems only (Bare et al., 1984; Nelson, 1988). Most of the proposed approaches search for approximate solutions. Within this category three strategies can be distinguished. The first strategy considers the adjacency constraints implicitly. The strategy consists of a sequential approach where for each planning period adjacency feasible solutions that meet harvest flow requirements are generated and the solution that yields the highest objective function value is chosen as the best solution. Non-adjacency feasible solutions are created by random mechanisms (Sessions and Sessions, 1988; Nelson et al., 1989; O'Hara et al., 1989) heuristics (Hokans, 1983) or dynamic programming (Bare et al., 1984). This approach provides "good" adjacency feasible solutions although its ability to solve problems with additional

constraints besides the flow and adjacency requirements is limited.

The second strategy considers explicitly the adjacency constraints. The constraints are included in a Model I formulation of the harvest scheduling problem. Solutions to this approach have been proposed through heuristics attached to linear programming solutions (Weintraub et al., 1982) or the use of duality accompanied by some heuristics (Weintraub et al., 1988). This approach permits the inclusion of additional constraints besides flow and adjacency. Hence it is more suitable for the habitat dispersion problem.

The third strategy defines the adjacency requirements as the exact age differences between adjacent harvest units (Roise, 1989). Thus, the exclusion period between adjacent harvest units is strictly determined. Although additional constraints can be included with this strategy the approach is limited since it requires a nonlinear formulation of the problem. Hence its use is restricted even for small size problems.

We present a solution approach to the habitat dispersion problem that can be generalized to include several additional constraints. Our approach considers wildlife habitat requirements as percentages of area covered by specific age classes and the spatial requirements are modeled through adjacency constraints. The solution

procedure is based on composite relaxation where the harvest flow and wildlife constraints are dualized in the objective function and a surrogate constraint is computed with the adjacency requirements. In this way the size of the problem is reduced considerably and "good" solutions are obtained within a short time. The presentation is organized as follows. The next section describes briefly composite relaxation and some theoretical principles used in the solution approach. The mathematical model for the habitat dispersion problem is introduced in the third section and the solution technique is described in the fourth section. The fifth section describes some computational considerations of the model. Finally the last section presents comparative results of the approach and discusses some effects of considering wildlife habitat constraints in the setting of long term harvest levels.

THEORETICAL BACKGROUND

Consider the following integer problem:

$$Z_{IP} = \max \mathbf{cx} \qquad \text{(IP)}$$

$$\text{s. t.}$$

$$\mathbf{A}\,\mathbf{x} \leq \mathbf{b}$$

$$\mathbf{x} \in \{0, 1\}$$

where $\mathbf{x}$ is the vector of decision variables, $\mathbf{A}$ is the matrix of technical coefficients and $\mathbf{b}$ and $\mathbf{c}$ are vectors of

suitable size. Such a problem can have several types of relaxation. A common relaxation is the linear programming (LP) relaxation, which consists of "relaxing" the integer restrictions of the problem. Another common relaxation is the Lagrangean relaxation, which is obtained by dualizing a set of constraints considered active. By letting μ be the set of multipliers associated with the constraints in the above problem IP, we can transform it to the Lagrangean relaxation problem below:

$$Z_{LR}(\mu) = \max \mathbf{cx} + \mu'(\mathbf{b} - \mathbf{Ax}) \qquad \text{(LR)}$$

$$\text{s. t.}$$

$$\mathbf{x} \in \{0, 1\}$$

$$\mu \geq \mathbf{0}$$

Another type of relaxation is known as "surrogate relaxation". This relaxation consists of finding a set of multipliers "π" associated with a constraint set, such that the product of the multiplier set times the constraint set yields a "surrogate constraint". An example of surrogate relaxation for problem IP is:

$$Z_{SR}(\pi) = \max \mathbf{cx} \qquad \text{(SR)}$$

$$\text{s. t.}$$

$$\pi'\mathbf{A}\,\mathbf{x} \leq \pi'\mathbf{b}$$

$$\mathbf{x} \in \{0, 1\}$$

$$\pi \geq \mathbf{0}$$

Finally, combining Lagrangean and surrogate relaxation we obtain the "composite relaxation". An example of this type of relaxation for problem IP is:

$$Z_{CR}(\mu,\pi) = \max \mathbf{cx} + \mu'(\mathbf{b} - \mathbf{Ax}) \qquad \text{(CR)}$$

$$\text{s. t.}$$

$$\pi'\mathbf{A}\,\mathbf{x} \leq \pi'\mathbf{b}$$

$$\mathbf{x} \in \{0, 1\}$$

$$\pi, \mu \geq \mathbf{0}$$

The solution approach through relaxation consists of finding sets of multipliers π and μ that minimize the "relaxed function". Thus for the Lagrangean relaxation we search for a set of multipliers μ^* that minimizes the Lagrangean function:

$$Z_{LR}(\mu^*) = \min \{ Z_{LR}(\mu) \}$$

for the surrogate relaxation we search for a set of multipliers π^* that minimize the surrogate function:

$$Z_{SR}(\pi^*) = \min \{ Z_{SR}(\pi) \}$$

and for the composite relaxation we search for a set of multipliers that minimizes the composite Lagrangean-surrogate function:

$$Z_{LR}(\mu^*, \pi^*) = \min \{ Z_{CR}(\mu, \pi) \}$$

The solution yielded by the "best" set of multipliers in any relaxation approach will provide an upper bound of the optimal solution of the original problem and in absence of a duality gap (duality gap is the difference between the optimal primal integer solution and the associated dual solution) the solution indeed corresponds to the optimal solution. The relationships between the bounds provided by different relaxations are well known for general integer

programming problems. They can be summarized as follows (Gavish and Pirkul, 1985):

$$Z_{LP} \geq Z_{LR}(\mu^*) \geq Z_{SR}(\pi^*) \geq Z_{CR}(\mu^*, \pi^*) \geq Z_{IP}$$

where Z_{LP} denotes the LP relaxation of problem IP. This relationship indicates that increasingly better solutions are obtained by using Lagrangean, surrogate and composite[1/] relaxation.

The critical issue in the efficiency of Lagrangean, surrogate or composite relaxation to solve integer problems is the derivation of a good set of multipliers. Subgradient optimization methods and various multiplier adjustment procedures have been proved to be effective for obtaining good multipliers for Lagrangean relaxation. Torres et al.,(1989) showed some adaptations to the subgradient method to find the minimum of the Lagrangean function of a simple area-based harvest scheduling problem.

Surrogate relaxation was suggested by Greenberg and Pierskalla (1970), and Glover (1975) as a means to close the duality gap in a Lagrangean relaxation approach. However, the difficulty in computing the multipliers limited its use. Many procedures have been proposed to search surrogate functions (Glover, 1975; Karwan and

---

1/ There exists a decomposition approach for relaxed problems which theoretically yields the best bounds (Guignard and Kim, 1987).

Rardin, 1981; Dyer, 1980; Gavish and Pirkul, 1985a; Sarin et al., 1987). Most of these procedures are based on the form of the surrogate function which is convex on the multipliers (for maximization problems). The most important applications of surrogate relaxation include heuristics for the solution of multiconstrained knapsack problems and some network applications.

Theoretically composite relaxation yields solutions closer to the true integer optimum than the other relaxations. However, although it has a great potential as solution technique in integer programming it has not been widely used for two main reasons: a) the difficulty of computing surrogate constraints and, b) because the relaxation approach has been mostly used as a basis for refinement algorithms such as branch and bound and not as a technique to obtain optimal solutions.

In the case of the habitat dispersion problem an approximate and good solution may be acceptable since flow requirements are hard to meet in any integer formulation and in very few situations the initial age class distribution of a forest yields feasible solutions for target wildlife requirements. Since composite relaxation yields solutions with minimum deviations from the flow and wildlife requirements and closer to the true optimum, it might be considered a potential tool for the solution of area-based harvest scheduling problems.

THE HARVEST SCHEDULING PROBLEM WITH HABITAT DISPERSION REQUIREMENTS

One of the first attempts to model wildlife requirements was made by Thompson et al., (1973). They formulated the harvest scheduling problem such that the allowable cut for certain age classes was restricted to maintain a desired age distribution. Mealey et al., (1982) presented a more complete example of how wildlife requirements can be modeled in an LP formulation by assigning lower bounds to the area covered by specified age classes. Weintraub et al., (1988) presented the same formulation for habitat requirements but they included the dispersion part of the problem in the form of adjacency constraints.

Many authors have calculated wildlife habitat requirements as percentages of a geographic area cover by specific age classes. These requirements are available for several species considering their food, cover, water, reproduction and space needs (Thomas, 1979; Mealey and Horn, 1981). Such percentages are flexible and depend on the site capabilities and the target species which are being managed. Hence modeling these requirements just by specifying lower bounds on the area covered by a given age class would provide a biased model since the excess of certain age classes might enrich the habitat of some species but could harm the habitat of others. For this reason we formulated the problem with upper and lower

bounds on the permissible area covered by age classes that define a wildlife habitat. Those bounds correspond to the proportions of the geographic area studied, which can be a whole forest, a basin or a subbasin. Additionally by including upper and lower bounds the number of dual variables increases. Hence the convergence of the solution algorithm is faster. The dispersion part of the problem is formulated through adjacency constraints. Since these constraints are handled explicitly in the solution procedure, the length of the exclusion period can be variable. Likewise, given that the problem is formulated as an integer programming problem the area of each harvest unit might be specified within the limits of the wildlife habitat requirements and even special forms of harvest units or special harvest conditions (for indicator species like the spotted owl) might be included in the model. The integer programming formulation of the wildlife habitat dispersion problem can be stated as:

$$Z_P = \max_{x_{ij}} \sum_{i=1}^{m} \sum_{j=1}^{n_i} c_{ij}\, x_{ij} \qquad (P)$$

s. t.

$$(1) \quad \sum_{i=1}^{m} \sum_{j=1}^{n_i} [V_{ijt} - V_{ij(t+1)}]\, x_{ij} = 0 \quad \forall\, t = 1, 2, .., T$$

$$(2) \quad \sum_{j=1}^{n_i} x_{ij} = 1 \quad \forall\, i = 1, 2, .., m$$

$$(3) \quad \sum_{i=1}^{m} \sum_{j=1}^{n_i} w_{ijkt}\, x_{ij} \leq u_k \quad \forall\, k = 1, 2, 3 \quad \forall\, t = 1, 2, .., T$$

$$(4) \qquad \sum_{i=1}^{m} \sum_{j=1}^{n_i} w_{ijkt}\, x_{ij} \geq l_k \qquad \forall\, k = 1, 2, 3 \quad \forall\, t = 1, 2, .., T$$

$$(5) \qquad \sum_{i \epsilon \Phi_r} \sum_{j \epsilon \Gamma_s} x_{ij} \leq 1 \qquad \forall\, r = 1, 2, .., m \quad \forall\, s = 1, 2, .., n_i$$

$$(6) \qquad x_{ij} \in \{0, 1\} \qquad \forall\, i = 1, 2, .., m \quad \forall\, j = 1, 2, .., n_i$$

where:

$x_{ij} =$ { 1 if the treatment "j" in harvest unit "i" is selected. The option of not to harvest in any period is considered a treatment for all harvest units.
0 if treatment "j" in harvest unit "i" is not selected.

c_{ij} : Total present net value derived from allocating treatment "j" in harvest unit "i" during "T" periods. Period "T+1" is considered the first period. Each harvest unit "i" has n_i treatments and the value of c_{ij} accounts for the acreage variation in different harvest units.

V_{ijt}: Volume harvested in harvest unit "i" under treatment "j" in period "t".

w_{ijkt}:Acres of the k-th wildlife habitat provided by harvest unit "i" under treatment "j" in period "t". The k-th wildlife habitat can be forage habitat, hiding habitat or thermal habitat.

u_k : Maximum number of acres of the k-th wildlife habitat required in the geographic area.

l_k : Minimum number of acres of the k-th wildlife habitat required in the geographic area.

Φ_i : Set of harvest units adjacent to harvest unit "i".

Γ_j : Set of treatments that permit final harvest in the same period as treatment "j".

Constraint set (1) corresponds to the strict even-flow constraints. Constraint set (2) corresponds to the area constraints. Constraint set (3) represents the upper bounds on the wildlife habitat requirements and constraint set (4) corresponds to the lower bounds of these requirements. Constraint set (5) represents the set of adjacency constraints. This set of constraints can be included under any of the procedures to write adjacency constraints (Meneghin et al., 1988; Torres and Brodie, 1989; Yoshimoto and Brodie, 1989). The last set of constraints represents the integer restrictions of the problem. Observe that the objective function value in this formulation corresponds to the maximization of the present net value of treatments during the specified planning horizon. Grouping the coefficients of problem P within suitable size sets of matrices and vectors, such a problem can be rewritten in matrix notation as:

$$Z_{P1} = \max \mathbf{c}\,\mathbf{x} \qquad \text{(P1)}$$

s. t.

(1) $\mathbf{D}\,\mathbf{x} = \mathbf{0}$

(2) $\mathbf{A}\,\mathbf{x} = \mathbf{1}$

(3) $\mathbf{W}\,\mathbf{x} \leq \mathbf{u}$

(4) $\mathbf{W}\,\mathbf{x} \geq \mathbf{1}$

(5) $\mathbf{Y}\,\mathbf{x} \leq \mathbf{b}$

(6) $\mathbf{x} \in \{0, 1\}$

where each set of constraints corresponds to the set of constraints with the same number as in problem P. Let the vectors π, μ_1, μ_2 be the set of dual variables associated with the set of constraints 1, 3 and 4 respectively. The Lagrangean relaxation of problem P1 obtained by dualizing constraint sets 1, 3, and 4 yields the following Lagrangean problem:

$$Z_D(\pi,\mu_1,\mu_2) = \mathbf{cx} - \pi'\mathbf{Dx} + \mu_1'(\mathbf{u} - \mathbf{Wx}) - \mu_2'(\mathbf{1} - \mathbf{W}) \qquad \text{(P2)}$$

s. t.

(1) $\mathbf{A}\,\mathbf{x} = \mathbf{1}$

(2) $\mathbf{Y}\,\mathbf{x} \leq \mathbf{b}$

(3) $\mathbf{x} \in \{0, 1\}$

(4) $\pi \gtrless \mathbf{0}$

(5) $\mu_1, \mu_2 \geq \mathbf{0}$

Letting σ be the vector of dual variables associated with the adjacency constraints, problem P1 can be further reduced by applying a surrogate relaxation to the adjacency constraints. The resulting relaxed problem yields:

$$Z_D(\pi,\mu_1,\mu_2,\sigma) = \mathbf{cx} - \boldsymbol{\pi}'\mathbf{Dx} + \boldsymbol{\mu}_1'(\mathbf{u} - \mathbf{Wx}) - \boldsymbol{\mu}_2'(\mathbf{1} - \mathbf{Wx}) \qquad \text{(P3)}$$

s. t.

$$\text{(1)} \qquad \mathbf{A}\,\mathbf{x} = \mathbf{1}$$

$$\text{(2)} \qquad \boldsymbol{\sigma}'\mathbf{Y}\,\mathbf{x} \leq \boldsymbol{\sigma}'\mathbf{b}$$

$$\text{(3)} \qquad \mathbf{x} \in \{0,\ 1\}$$

$$\text{(4)} \qquad \boldsymbol{\pi} \gtrless \mathbf{0}$$

$$\text{(5)} \qquad \boldsymbol{\mu}_1,\ \boldsymbol{\mu}_2,\ \boldsymbol{\sigma} \geq \mathbf{0}$$

Problem P3 is a mixture between Lagrangean and surrogate relaxation. Strictly speaking it does not correspond to a composite relaxation of problem P1. However it will be considered in this way since it groups Lagrangean and surrogate relaxation. Additionally it presents the same properties as composite relaxation (Karwan and Rardin, 1980). Note that the relaxed problem (P3) is not as easy to solve as if we had also applied a Lagrangean relaxation to the adjacency constraints in problem P2 or if we had computed a single surrogate constraint from all the constraints in problem P1 (surrogate relaxation). However, the resulting relaxed problem is small and relatively easy to solve since it is only complicated by the "surrogate constraint". Moreover, theoretically it will yield better solutions than a simple Lagrangean relaxation or a single surrogate relaxation applied to problem P1. Observe also that if the optimal surrogate multipliers were known, the surrogate constraint might be dualized in the objective function. This process, similar to a Lagrangean decomposition (Guignard and Kim, 1987) would yield a better

solution than the composite relaxation and additionally a smaller relaxed problem. Composite relaxation applied in this way to the habitat dispersion problem offers the following advantages:

i) The relaxed problem is small and relatively easy to solve. Hence solutions can be obtained within a short time.

ii) The solutions obtained through composite relaxation will satisfy the adjacency requirements and area constraints. This would be more difficult to obtain if we had applied a Lagrangean relaxation to the adjacency constraints.

iii) The solutions will meet harvest flow and wildlife habitat requirements with minimum infeasibilities. Integer formulations of these problems usually yield infeasible solutions not only due to the integer restrictions but also to the absence of an initial age class distribution that provides feasible solutions for the wildlife requirements. The relaxation approach will yield a solution that minimizes those infeasibilities.

Therefore, composite relaxation applied to the habitat dispersion problem will yield feasible solutions for the area and adjacency constraints. Additionally, this relaxation will yield a solution very close to the true optimum in a short time and with minimum infeasibilities in the manager imposed harvest flow and wildlife habitat constraints.

SOLUTION PROCEDURE

The solution procedure through relaxation consists of finding sets of multipliers π, μ_1, μ_2, and σ that minimize the relaxed problem P3. The search procedure we propose is an iterative process. The first step consists of computing the surrogate multipliers. Once the surrogate multipliers are known, the sets of Lagrange multipliers in problem P3 are adjusted at successive iterations, such that at each iteration lower valued solutions for this relaxed problem are obtained. Note that once the surrogate multipliers are computed only the Lagrange multipliers are adjusted at each iteration. The iterative process ends when the permissible maximum deviation in harvest flows or wildlife requirements are met, or when the search procedure has reached ending conditions. For explanatory purposes we will divide the solution procedure in 3 parts: i) the computation of surrogate multipliers for adjacency constraints. ii) The solution of the relaxed problem and iii) the adjustment procedure of Lagrange multipliers in the relaxed problem.

Computation of multipliers for the surrogate relaxation

The most simple and successful algorithm to compute multipliers for a surrogate relaxation has been the one proposed by Gavish and Pirkul (1985). This algorithm was designed for the multiconstrained knapsack problem. It requires all the technical coefficients in the constraints

as well as the right-hand-side vector to be nonnegative. These requirements are met by the adjacency constraints. Hence it was the algorithm we selected. The algorithm is based on searching the minimum of the surrogate function in a two-constraint problem. Assume we perform a surrogate relaxation to a two-constraint problem. The relaxed problem becomes:

$$Z_1(\mu,\pi) = \max \mathbf{cx} \qquad (1)$$

$$\text{s. t.}$$

$$(\mu, \pi)\mathbf{A}\,\mathbf{x} \leq (\mu, \pi)\mathbf{b}$$

$$\mathbf{x} \in \{0, 1\}$$

As the reader can verify the surrogate function $Z_1(1,\pi)$ is a convex function of π. Hence the minimum of such a function can be easily found by any searching algorithm (bisection, golden search or Fibonacci search). The search procedure yields a pair of multipliers $(1, \pi^+)$ close to the optimal pair $(1, \pi^*)$. If such a pair yields a surrogate constraint that does not violate any of the original constraints then an "ϵ-neighborhood set of optimal multipliers" has been found and the pair of multipliers $(1, \pi^+)$ is used to compute the surrogate constraint. The procedure does not yield the optimal set of "surrogate multipliers" . However, good sets of multipliers can be obtained. The algorithm can be extended to multiple constraints by following Procedures 1 and 2 in Gavish and Pirkul (1985). These procedures have been grouped in the

following Procedure 1. Problem "SR" could be considered as the example problem.

Procedure 1

Step 1 Determine the constraint "i^*" which corresponds to the one-constraint problem with lowest objective function value. Let constraints "i^*" be the current surrogate constraint and assign a value of "1" to the surrogate multiplier associated with it.

Step 2 Determine the amount by which each constraint is violated. If no one is violated STOP, current surrogate constraint is the final surrogate constraint. Otherwise identify constraint "i'" as the most violated and let π be the multiplier associated with that constraint.

Step 3 Form a two-constraint problem with constraints "i^*" and "i'". Let π_L and π_U be two multipliers such that the solution to the two-constraint problem $Z_{SR}(1, \pi)$ satisfies constraint i^* at $\pi = \pi_L$ and does not satisfy it at $\pi = \pi_U$.

Step 4 If $\pi_U - \pi_L \leq \epsilon$ and if the surrogate objective function has not declined or a maximum number of iterations has been reached then STOP with the current constraint i^* as the surrogate constraint.

Step 5 Let $\pi = \pi_L + (\pi_U - \pi_L)/2$. Solve $Z_{SR}(1,\pi)$. If in the solution:

i) both constraints are satisfied GO TO STEP 6.

ii) only the first constraint is satisfied. Let $\pi_L=\pi$. GO TO STEP 4.

iii) the first constraint is not satisfied. The let $\pi_U = \pi$. GO TO STEP 4.

Step 6. Replace current surrogate constraint i^* with $\pi^\circ \mathbf{A} + \pi a_i$, where π° is the multiplier vector with the current surrogate constraint. Update $Z_{SR}(\pi)$. GO TO STEP 2.

The procedure is very effective especially in the case of adjacency constraints where each constraint has very few positive technical coefficients.

Solution of the relaxed problem

If a guess for the multipliers associated with the harvest flow and wildlife constraints is available, and the multipliers of the surrogate relaxation are known, the relaxed problem is formed just by the area constraints and the surrogate constraint (problem P3). Observe that the dualized constraints could be interpreted as weights on the original cost coefficients. So the "corrected cost coefficients" of the form:

$$\tilde{\mathbf{c}} = \mathbf{c} - \pi'\mathbf{D} - \mu_1'\mathbf{W} + \mu_2'\mathbf{W}$$

constitute the objective function coefficients of the relaxed problem (P3). Note that if problem (P3) did not have area constraints and the surrogate multipliers were known, problem P3 could be categorized as a "knapsack problem". The knapsack problem is a single one-constraint

integer problem and several algorithms are available for its solution. We will review a procedure to solve the relaxed problem without area constraints (the knapsack problem) and then adapt the solution procedure to consider these constraints.

The algorithm we selected makes use of the "bang-for-buck" ratios. These ratios are defined as the "ratios of the objective function coefficients to the coefficients of the resource constraint" (Nauss, 1976). Hence in a one-constraint problem the greater the ratio, the higher the chance that the corresponding variable will be equal to one in the problem solution. This principle has been used in many successful algorithms for solving the single-constraint knapsack problem (Salkin and Kluyver, 1975; Nauss, 1976; Balas and Zemel, 1980) and was first applied to the multiconstraint knapsack problem by Gavish and Pirkul (1985). They constructed a surrogate constraint with the constraints of the multiconstraint knapsack problem. We use a similar procedure for the adjacency constraints. Once the surrogate constraint is formed the resulting knapsack problem is solved following Procedure GP in Gavish and Pirkul (1985). This algorithm is duplicated in the following Procedure 2. The notation corresponds to our relaxed problem without area constraints.

Procedure 2

Step 1. Calculate the bang-for-buck ratios which in the relaxed problem P3 correspond to $\tilde{c}_{ij}/\sigma' \mathbf{Y}_{ij}$. Where $\tilde{c}_{ij}$ represents the corrected objective function coefficient and $\mathbf{Y}_{ij}$ is the column in the adjacency constraints corresponding to the variable x_{ij}.

Step 2. Sort the variables according to decreasing order of bang-for buck ratios.

Step 3. Fix variables equal to one according to the order determined in step 2. If fixing a variable equal to one causes violation of one of the constraints fix that variable equal to zero and continue. Denote the feasible solution determined in this step as $\tilde{x}$.

Step 4. For each variable fixed to one in step 3 ($\tilde{x}$) fix that variable equal to zero and GO TO STEP 3 to determine additional feasible solutions. Stop when no more additional solutions can be found and choose the solution with highest objective function value.

The procedure is very effective for multiconstrained knapsack problems. Pirkul (1987) reported solutions within 1 percent of the optimal solution for several problems. Despite its efficiency this algorithm can not be fully applied to our relaxed problem (P3), since we have to consider the area constraints.

Given that the area constraints indicate that only one treatment per stand can be included in the solution, then

an intuitive modification to Procedure 2 would consist of constraining the size of the ordering in step 2 (Procedure 2) to be equal to the number of harvest units. Obviously the treatment of each harvest unit selected in the ordering, would be the one with largest bang-for-buck ratio. In most of the cases the ordering violates the adjacency constraints. One way to deal with this problem is to substitute the treatment that violates the adjacency constraints with the treatment that considers no harvest at all, which obviously is always feasible. However, this procedure skips many options and in most cases poor solutions are reached. We opted for selecting the treatment with the next largest bang-for-buck ratio (i.e. the second largest) and resort the ordering. If this treatments causes any violation of adjacency constraints then we substitute the treatment that includes no harvest at all. Alternatively we could have repeated the latter process of resorting the variables with largest bang-for-buck ratios before resorting the non-harvest choice. But, this procedure is very slow and does not improve substantially the solutions obtained. The following Procedure 3 considers the adaptations to Procedure 2 to include the area constraints.

Procedure 3

Step 1. Calculate the bang-for-buck ratios $\tilde{c}_{ij}/\sigma' Y_{ij}$.

Step 2. For each harvest unit choose the treatments with the largest and the second largest bang-for-buck ratios. Sort the treatments with largest ratios in decreasing order. Store the the value of the second largest ratio for each harvest unit.

Step 3. Fix variables equal to one according to the order determined in step 2. If fixing a variable equal to one causes violation of one of the constraints fix that variable equal to zero, recover the variable with the second largest ratio for that harvest unit, resort its ratio in the ordering defined in step 2 and continue. If the variable with the second largest ratio for a harvest unit violates any adjacency constraint then fix that variable to zero and fix the variable associated with the no harvest option for that harvest unit to one. Denote the feasible solution determined in this step as $\tilde{x}$.

Step 4. For each variable fixed to one in step 4 ($\tilde{x}$) fix that variable equal to zero and GO TO STEP 3 to determine additional feasible solutions. Just include variables defined in the first $\tilde{x}$ solution. Stop when no more additional solutions can be found and choose the solution with largest objective function value.

Procedure 3 provides "excellent" solutions to problems

with adjacency constraints only. Test problems provided optimal solutions in more than 65% of the cases and the nonoptimal solutions were always within 1% of the optimal objective function value.

Adjustment of Lagrange multipliers in the relaxed problem

So far we know how to solve the relaxed problem given a guess of Lagrange multipliers π, μ_1, μ_2 and the "best" set of surrogate multipliers. However, we need a mechanism to correct the multipliers π, μ_1, and μ_2 such that at successive iterations, the new solutions will minimize the relaxed problem (P3). The mechanism we adopted is the subgradient method. Following the subgradient method, the sequence of values for the multipliers is given by:

$$\begin{aligned} \pi^{k+1} &= \pi^k - t_k(\mathbf{0} - \mathbf{Dx}) \\ \mu_1^{k+1} &= \max\{0,\ \mu_1^k - t_k(\mathbf{u} - \mathbf{Wx})\} \\ \mu_2^{k+1} &= \max\{0,\ \mu_2^k - t_k(\mathbf{l} - \mathbf{Wx})\} \end{aligned} \tag{2}$$

where k represents the iteration number and t_k is the step size. The step size is computed as:

$$t_k = \frac{\lambda_k\,[Z_D(\pi,\mu_1,\mu_2,\sigma) - \hat{Z}]}{\|\mathbf{D}\,\mathbf{x}\|^2 + \|\mathbf{u}-\mathbf{Dx}\|^2 + \|\mathbf{l}-\mathbf{Wx}\|^2} \tag{3}$$

where $Z_D(\pi,\mu_1,\mu_2,\sigma)$ is the objective function value of the relaxed problem (P3) and $\hat{Z}$ and λ are set as in Torres et al., (1989). The following Procedure 4 summarizes the updating procedure for the multipliers associated with harvest flow and wildlife requirements.

Procedure 4

Step 1. Compute the infeasibilities in harvest flow and wildlife habitat constraints of the primal problem associated with the solution from the relaxed problem.

Step 2. Compute the step size using (3) and recommendations in Torres et al., (1989).

Step 3. Update multipliers using equations in (2).

The objective function coefficients of the relaxed problem are corrected using the new multiplier set to initiate another iteration of the search. The whole searching procedure is simple and fast. The following Procedure 5 integrates the latter Procedures 1, 3, and 4. To summarize the searching algorithm we propose to find the best multipliers for the relaxed problem. Procedure 5 considers adjacency, wildlife habitat and flow constraints all together. But it can be adapted to consider any combination of the three sets of constraints.

Procedure 5

Step 1. Use Procedure 1 to compute the surrogate multipliers associated with the adjacency constraints of the problem. Use Procedure 3 to solve the relaxed problem. If the primal solution associated with the relaxed problem satisfies the maximum deviation of harvest flows and wildlife habitat constraints STOP.

The best solution through composite relaxation has been reached.

Step 2. Use subgradient optimization to solve the scheduling problem without adjacency constraints. The set of best multipliers π, μ_1, and μ_2 becomes the first guess. Correct the objective function coefficients "$\tilde{c}$" of the relaxed problem (P3).

Step 3. Use Procedure 3 to solve the relaxed problem (P3).

Step 4. Check the infeasibilities of the associated primal solution. If they are within the permissible deviations STOP. The best solution through relaxation has been reached. Otherwise if the sum of absolute infeasibilities is smaller than previous, store the solution as the "best solution".

Step 5. Use procedure 4 to correct the multipliers π, μ_1, and μ_2. If stopping rules on the step size have been reached, then recover best solution, STOP. Otherwise correct the objective function coefficients of the relaxed problem ($\tilde{c}$). GO TO STEP 3.

The stopping rules on the step size are given by a minimum value of or a minimum value of the step size. A maximum number of iterations could be used also as a stopping rule, however this selection is problem dependent so it is not recommended.

IMPROVING COMPUTATIONAL EFFICIENCY

Computational efficiency might be increased by using the following strategies in the searching procedure.

i) Procedure 1 step 1. The solution to the single-constraint problems could be obtained as if they were continuous "knapsack" problems. This can be accomplished by filling the knapsack completely with the variables of largest bang-for-buck ratios until no more room remains in the knapsack. At this point the variable that could not fit is placed in the knapsack at fractional level, such that the knapsack is filled. This strategy reduces the time taken by Procedure 1 and provides good results.

ii) Procedure 1 step 3. The interval of uncertainty for π can be obtained through Swann's bounding search (Kowalik and Osborne, 1968). To speed up the procedure we used a value of zero for the lower bound, and the value of the right hand side of each constraint as upper bound. If adjacency constraints are written as in Torres and Brodie (1989) or Yoshimoto and Brodie (1989) the latter strategy yields good bounds.

iii) Procedure 1 step 4. The maximum number of iterations is important to define a good set of surrogate multipliers, especially for large problems with many periods, which usually have hundreds of adjacency constraints. We ran different problems testing several values for the maximum

permissible number of iterations. The number depends mainly on the size of the problem but in general results indicated that after 2000 iterations there is no difference in the quality of the surrogate constraint obtained. Problems with many constraints generally reach the maximum permissible number of iterations. For these cases a good strategy to follow consists of dividing the adjacency constraints in groups. These groups can be of any size. Then we apply Procedure 1 to each group, such that we form a subgroup of surrogate constraints. The last step consists of applying Procedure 1 to the subgroup of surrogate constraints to obtain just one final surrogate constraint. Observe that if all harvest units have the same age or if all of them are older than the minimum harvest age, the adjacency constraints for successive periods are repeated with different variables but for the same harvest units. In this case the grouping can be performed by periods. This way of grouping is very efficient and yields "excellent" surrogate constraints since groups are independent from each other.

iv) Procedure 3 step 4. This step is very important and must be efficient since it represents the "bottle neck" of this procedure. In this step a strategy similar to that of Balas and Zemel (1980) defining "core problems" could be adopted. This procedure consists of setting equal to 1 those variables that are most likely to be in the optimal

solution. Thus not many variables have to be fixed to zero and fewer iterations within Procedure 3 are required. By defining "core problems" we were able to reduce the computation time taken per iteration in small test problems to a level of 17% (3 periods and less than 60 harvest units). Note that this reduction in time will be greater as the size of the problem increases.

v) Sign of the multipliers associated with wildlife constraints. Preliminary results indicated that the lower bounds of the thermal habitat constraints and the upper bounds of the forage constraints were mostly violated. This indicates that the multipliers associated with these constraints could not drive the solution. Checks on final dual variables revealed that the multipliers associated with the upper bounds of the thermal habitat constraints and the ones associated with the lower bounds of the forage constraints had a value of zero. We might think that this is obvious and not important since those constraints are not binding and therefore its marginal value is zero. However for our solution approach it is very important since we want those constraints to push the wildlife requirements so we can obtain solutions with such requirements set between the specified lower and upper bounds. Otherwise it would be easier just to specify a lower bound for the wildlife requirements. We forced those constraints to drive the solution to the desired wildlife

requirements by allowing the multipliers associated with them to take negative values. When we tested this alternative the formulation was changed. The inequalities in problem (P1) were changed to strict equalities. Hence all the multipliers of the Lagrangean-surrogate function $Z_D(\pi,\mu_1,\mu_2,\sigma)$ in problem (P3) were permitted to take negative values.

By using this strategy we obtained the expected results. The solutions yielded wildlife requirements between the specified lower and upper bounds when possible and they maintained the harvest levels within the permissible deviations.

EXAMPLES

The algorithm described in Procedure 5 was coded in FORTRAN 77. The computer code recognizes any combination of wildlife habitat, adjacency or flow constraints to perform the search. A surrogate relaxation is performed only if there exists a set of adjacency constraints. Any other set of constraints is solved through Lagrangean relaxation. We tested the performance of the algorithm with 3 combinations of sets of constraints i) flow and wildlife habitat constraints, ii) flow and adjacency constraints and iii) flow, wildlife habitat, and adjacency constraints all together. We present some comparative results for each of these cases to show the efficacy of relaxation in solving this kind of problems.

Area-based scheduling problems with wildlife habitat constraints

For this kind of problem the relaxed problem resembles problem (P2) without constraint set (5). The search for the minimum of the Lagrangean problem is performed through the subgradient method following the recommendations in Torres et al., (1989) and allowing negative values for all the multipliers. We created three example problems with 50, 100 and 136 harvest units respectively. The age distribution of the 50 unit forest varies from 10-100 years

and resembles a normal distribution, while the last two forests have an age distribution with five age classes namely: 10 (8.8%), 20 (17.5%), 30 (20.4%), 40 (11.5%), and 80 (41.8%) years. The scheduling problems were formulated for 3, 5, and 8 planning periods. To set the wildlife habitat constraints we used the following habitat requirements:

Forage habitat: Stands 20 years old or younger, covering 15-25 percent of the total area.

Hiding habitat: Stands with age between 21 and 40 years, covering 30 - 50 percent of the total area.

Thermal habitat: Stands older than 70 years covering 20-50 percent of the total area.

Based on the theoretical result that integer solutions for problems with a large number of variables approach solutions obtained by linear programming (LP), and considering that the LP solutions are an upper bound of the true integer solutions, we compared the results from relaxation with solutions obtained from LP. Table 7 shows these comparisons for nine example problems. Note that although relaxation yields integer solutions and LP yields continuous solutions, the latter procedure yielded few fractional values. Hence both procedures are more comparable. Observe that the objective function value obtained from both procedures is very close. In all cases except problem 6, LP solutions yielded larger objective function value, which is basically due to the continuous

Table 7. Comparative results of relaxation and linear programming for area-based problems with wildlife habitat and harvest flow constraints.

Problem No.	No. of periods	No. of stands	No. of vars.	L. P. SOLUTION				LAGRANGEAN RELAXATION SOLUTION						AVERAGE VIOLATIONS IN WILDLIFE HABITAT CONSTRAINTS (Considering all periods in the optimization)					
				Obj. fcn. value ($) thousands	Harvest flow (MCF)	Time (hrs) *	No. NIN **	Obj. fcn. value ($) thousands	Time (hrs) *	deviation of flows (%) from period to period max.	avg.	min.	average harvest (MCF)	UPPER BOUND (%) Forage	Hiding	Thermal	LOWER BOUND (%) Forage	Hiding	Thermal
1	3	50	173	2980	2979	0.054	3	2883	0.012	- 6.27	1.30	+ 0.14	3070	63	0	0	0	24	36
2	3	100	328	4020	5430	0.124	6	4000	0.039	+ 8.04	5.13	- 3.60	5546	77	0	0	0	11	34
3	3	136	436	33096	17371	0.224	2	32976	0.057	- 5.91	3.80	+ 1.41	17112	61	0	0	0	4.2	33
4	5	50	308	2803	2287	0.643	6	2767	0.032	- 9.95	4.53	- 0.49	2317	38	0	0	0	19.4	0
5	5	100	489	4672	4880	0.892	8	4154	0.096	- 10.91	7.00	- 1.05	4462	38	0	0	0	4.6	20
6	5	136	787	32932	12825	1.456	7	33135	0.175	- 7.73	3.24	+ 0.18	13039	40.8	0	0	0	6.6	17.1
7	8	50	830	2635	1731	1.372	12	2477	0.199	+ 10.80	4.36	+ 0.29	1778	13	0	0	0	19.1	0
8	8	100	1585	4124	3240	2.863	10	3922	0.458	+ 9.91	4.72	+ 0.73	3180	8.9	0	0	0	12.8	19.4
9	8	136	2013	32890	10564	3.240	11	32669	0.907	- 10.21	6.98	+ 0.27	10981	25	0	0	0	9.2	2.9

* Times correspond to a Compaq (386/25)

** No. of harvest units with non-integer values in the LP solution.

nature of the LP approach. The average harvest flow is also very close in both solution techniques and generally higher in the relaxation approach. This is due to the fact that flow constraints for integer solutions have to be met within certain deviations from even-flow and for some problems high positive infeasibilities overestimated the average harvest flow. Note also that harvest flow deviations in the relaxation approach are very small and in average always less than 10%. LP solutions for problems 4 and 7 were obtained by allowing a deviation of 5% in harvest flows, since even-flow formulation yielded infeasible LP solutions. In all cases when we tried to obtain LP solutions with the specified bounds for wildlife requirements we obtained "infeasible" solutions. For problems with five or eight periods, we tried to modify just the first two periods to allow feasible solutions. However, the procedure still yielded infeasible solutions. The only way we obtained feasible solutions was by rearranging the wildlife habitat requirements bounds to be equal to the bounds determined by the relaxation solution. Observe that in order to obtain a feasible linear solution a forest would have to have an age distribution such that after the first period harvest the residual distribution would fit into the wildlife requirements bounds. This requirement is obviously difficult to meet in real problems. A likely approach to obtain feasible solutions would consist of modifying the wildlife requirement bounds

for the first periods, keeping the desired bounds in the last periods of the optimization. However, this approach raises the questions: How much the bounds should be expanded ?, How many periods would be allowed with the expanded bounds ?. These questions are difficult to answer but focus attention on correct areas to investigate for management of forests in transition.

Violations to wildlife constraints depend mostly on the original structure of the forest, the minimum harvest age, value of the upper and lower bounds of wildlife requirements, planning horizon and so on. However, the relaxation approach yields solutions that efficiently convert the forest to the desired structure with minimum infeasibilities in flows and wildlife requirements. Figure 15 shows the average percentage in which wildlife requirements were violated for a 136 harvest unit example forest under different planning horizons. Each period length is a separate optimization problem. Observe that the violations on upper bounds of forage constraints and lower bounds of thermal constraints (the other bounds were not violated) are smaller as we increase the number of periods. For problems with small planning intervals these violations occurred in the last periods of the optimization, driven to meet the harvest flow constraints. When the number of periods increases the harvest levels decrease. Hence in order to maintain the harvest level in the last periods of

the optimization it is not necessary to use larger amounts of old growth, so the wildlife requirements for the last periods can be satisfied. Thus for long planning horizons the relaxation approach reduces smoothly the violations of wildlife requirements in each period. Note that hiding constraints are constantly violated. This is because the specified requirements were set beyond the attainable proportions.

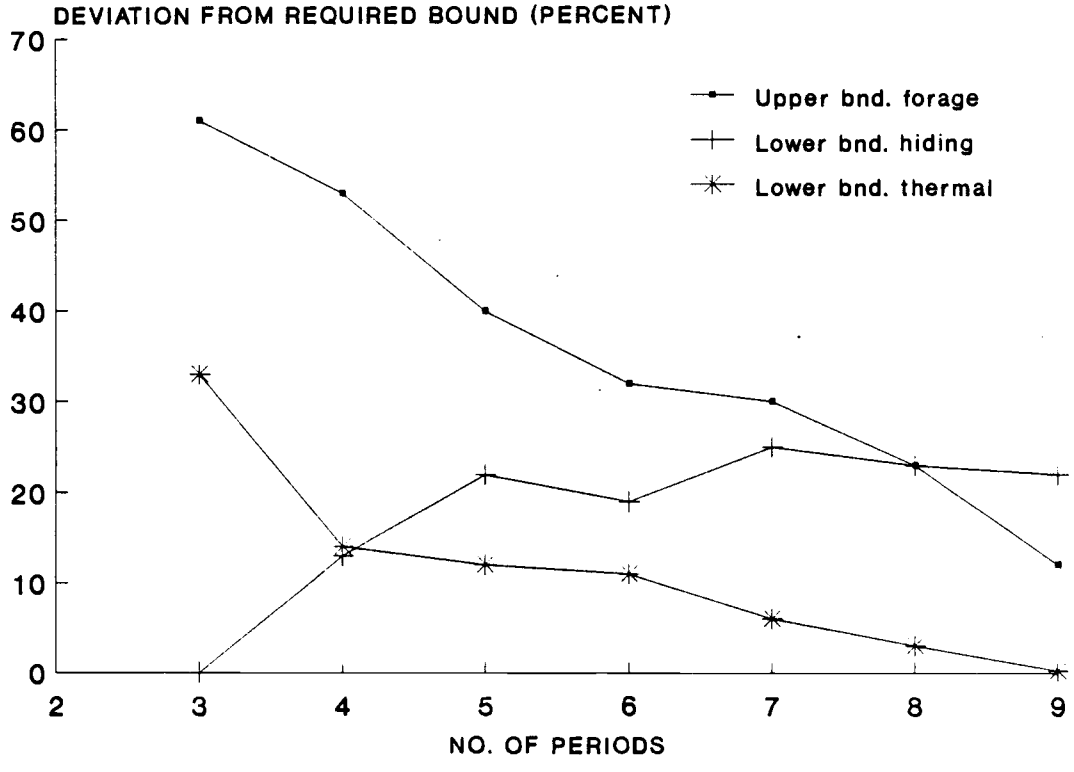


Figure 15. Violations in wildlife habitat constraints for several periods.

Relaxation is a solution technique that has a great potential application in solving this kind of highly infeasible problem. It yields a tight upper bound (i.e. a

solution very close to the true optimal solution) for the optimal solution (when it exists) or solutions with minimum violations of constraints for infeasible problems. Although one distinct disadvantage of relaxation to obtain bounds is that "only an optimal solution to the relaxed problem guarantees an upper bound on the optimal solution of the original problem" (Nemhauser and Wolsey, 1988), procedures to select the "best" solution to the relaxed problem such as the one proposed by Torres et al., (1989) guarantee solutions very close to the true optimum or the "best" solution for infeasible problems.

Another important feature to notice in the relaxation approach is its short solution time. Problems with wildlife constraints were solved in at most 30% of the time taken by LP. It is expected that as the size of the problem is increased the ratio of the solution time by relaxation to the LP solution time will become smaller.

Area-based scheduling problems with adjacency constraints

In this case the relaxed problem resembles problem P3 without the dualized wildlife habitat requirements. So the relaxed problem includes Lagrangean and surrogate relaxations. The solution strategy follows Procedure 5, without considering the corrections to multipliers associated with wildlife habitat constraints. We created three example forests with 15, 30 and 50 harvest units

respectively. All of them were overmature forests with ages older than 80 years. The harvest scheduling problems were formulated for 3, 5, and 8 periods. No reharvest was allowed in any harvest unit. The problems were solved by relaxation and the optimal integer solution was obtained by branch and bound using a commercial software package (MILP88). In order to have a fair comparison between relaxation and the branch and bound algorithm the deviations on harvest flow for the integer programming formulations were fixed according to the maximum deviations obtained from relaxation, i.e. if the maximum deviation in harvest flows from relaxation was 5%, the integer programming problem was formulated allowing 5% of deviations in the harvest flows from period to period. Table 8 shows the comparative results.

Observe that the relaxation approach becomes very efficient as the number of variables increases in the problem. Although most of the examples show this trend it is not as evident as examples 1, 2, and 3 (table 8). For all examples it is quite clear how for the same problem with a different number of periods the relaxation approach is increasingly more efficient as we increase the number of periods. Additionally these solutions are better as the size of the problem increases. Observe that problems with a larger number of harvest units have smaller deviations in harvest flows and minimum deviations from the objective function value. This is probably due to the fact that in

Table 8. Comparative results of relaxation and branch and bound for area-based problems with flow and adjacency constraints.

Problem No.	No. of periods	No. of stands	No. of vars.	COMPARISONS BETWEEN RELAXATION & B. B.*				
				Obj. fcn. deviation %	Time ratio **	dev. harvest flow (%) from period to period max.	avg.	min.
1	3	15	60	+ 2.22	0.050	- 8.2	3.3	0.02
2	3	30	113	- 0.08	0.041	- 2.9	0.7	0.01
3	3	50	135	+ 3.28	0.018	0.9	0.2	0.03
4	5	15	90	+ 1.34	0.014	-11.2	2.9	0.0
5	5	30	171	+ 1.57	0.009	- 4.9	1.9	0.07
6	5	50	303	x	x	+ 1.7	0.8	0.02
7	8	15	135	- 2.73	0.016	-10.2	5.3	0.05
8	8	30	259	x	x	+ 1.2	0.1	0.05
9	8	50	574	x	x	+ 3.4	1.2	0.36

* Branch and Bound.

** Ratio of the relaxation solution time to the branch and bound solution time.

bigger problems (more units) there are more spatially feasible solutions and the algorithm can find a good one more easily. We might also attribute these results to the properties of composite relaxation. Recall that theoretical results stated earlier indicate that composite relaxation yields better bounds than a single Lagrangean relaxation. Results in table 8 follow these statements since they show smaller deviation in harvest flows and additionally very small deviations from the optimal objective function value than results from a simple Lagrangean relaxation such as those shown in table 7 or the ones shown in Torres et al.,(1989).

Given that solution time for branch and bound increases exponentially with the number of variables in the problem, it is expected that relaxation will be more efficient as the number of harvest units is increased.

Although the sequential approaches might be slightly faster in providing solutions to the harvest scheduling problem with adjacency constraints, the composite relaxation approach presents several advantages over the former approaches, namely:

i) Composite relaxation performs a "suboptimization" of the problem, hence it does not require as input data the "target harvest levels" like the approaches by Sessions (1988) or Nelson et al., (1989). Binary search combined with a sequential approach could be used to compute the periodic harvest levels. However, the sequential approach only by chance could be as good as relaxation, given that the latter considers a "suboptimization" of all periods at the same time while the sequential approach optimizes period by period. Thus although the harvest levels of both approaches might be similar it is very likely that the relaxation approach will yield larger objective function values.

ii) The relaxation approach is able to easily handle additional sets of constraints while the sequential approaches are limited in ability to do so. This feature of relaxation is evident in the following discussion.

Area-based scheduling problems with flow, adjacency and wildlife constraints

For this kind of problem the relaxed problem is similar to problem (P3) and the solution method corresponds to the one described as Procedure 5. Relaxation was tested with 2 example forests. The first one is the 50 harvest unit forest described earlier. The scattering of the harvest units is depicted in figure 16. The second example is the 136 harvest unit forest also described earlier. The scattering of the harvest units is depicted in figure 17. This forest is a real example and corresponds to the Green River subbasin in the Siuslaw National Forest. Given the difficulty in obtaining optimal integer solutions for large problems we just tested the example forest with formulations for 3 periods. In each case the "optimal" integer solution was obtained by setting the harvest flow and the wildlife habitat deviations to the bounds defined by the relaxation solutions. Table 9 shows the comparative results. Observe that in the first problem the solutions are very close. In fact just one harvest unit yielded a different treatment in the final solution. In this case the integer solution was slightly lower but better met the flow constraints than relaxation and decreased the violation of one forage constraint. In the second example we were not able to verify an optimal solution after 52 hours of search. The solution depicted in table 9 corresponds to the best solution obtained at that time.

Table 9. Comparative results of relaxation and branch and bound for area-based problems with flow, wildlife habitat and adjacency constraints.

Problem No.	No. of stands	INTEGER PROGRAMMING SOLUTION				COMPOSITE RELAXATION SOLUTION					
		No. of vars.	No. of const.	Obj. fcn. value($) thousands	Time (hrs) *	Obj. fcn. value ($) thousands	Time (hrs)	max. flow deviation (%)	max. wildlife deviation Forage (%)	Hiding (%)	Thermal (%)
1	50	173	141	6766	50.500	6785	0.626	- 4.6	+ 10	26.83	0
2	136	436	327	18432	52.354	19485	7.405	- 5.5	+ 52	15.74	29

* Time correspods to a Compaq (386/25)

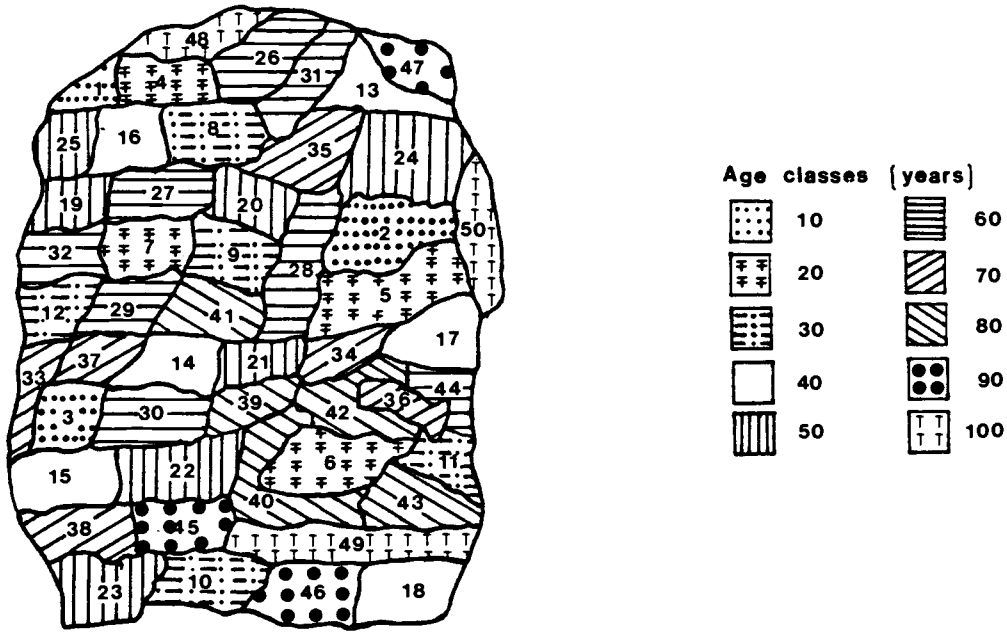


Figure 16. Pattern of harvest units for a 50 unit example forest.

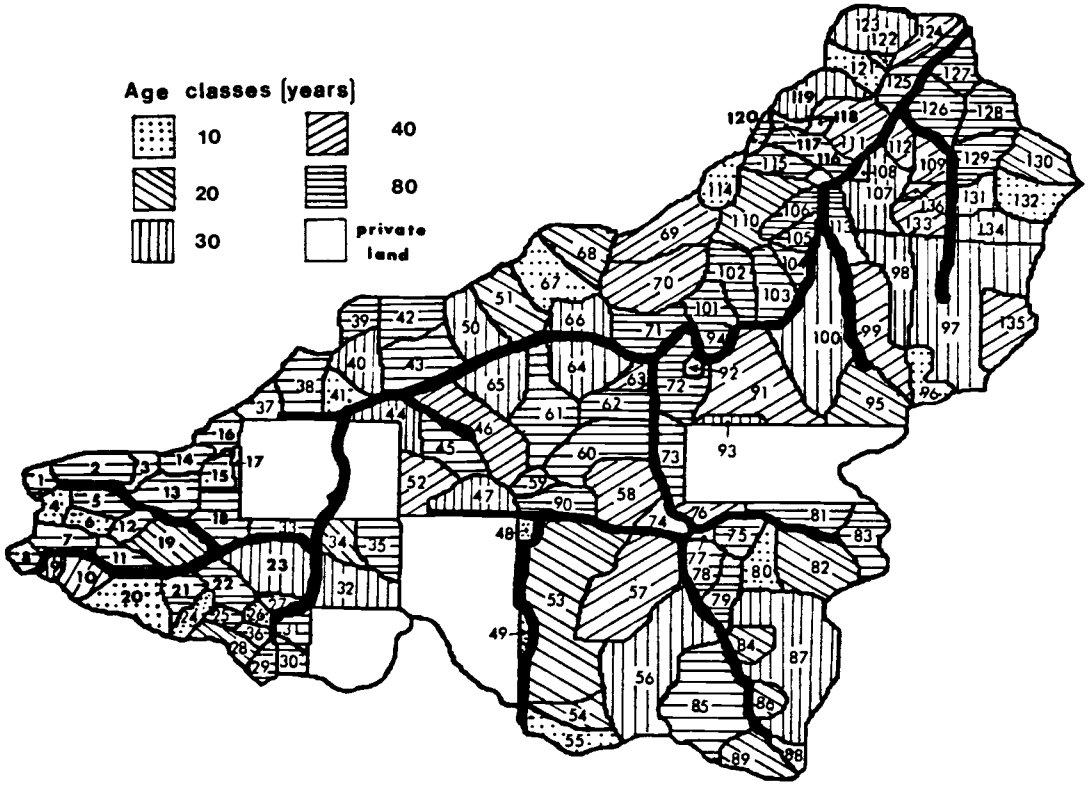


Figure 17. Pattern of harvest units for a 136 unit example forest.

Like the latter example, this solution has smaller objective function value than that reported by relaxation. Although in this case the harvest levels were lower, flow constraints were met better, but this was not true for the wildlife constraints which were met in the same proportion as the relaxation solution. In both examples the maximum deviation of harvest flow obtained by relaxation did not reach more than 5.5 percent. Hence we might consider the relaxation solution a better "practical solution".

Although it might be argued that two examples are not enough to compare the relaxation procedure with the true optimum, results obtained from previous combinations of constraints indicate that the relaxation approach will yield solutions with a trend similar to these examples. To reinforce the statement additional example problems were solved through relaxation. In all cases the infeasibilities were very small. Based on these results and the upper bound relationships stated in the second section of this paper (Gavish and Pirkul, 1985) we can assert that relaxation solutions are close to the optimal. Moreover, considering that relaxation yields solutions within the manager's permissible deviations in flows and additionally converts efficiently (with minimum deviations) the forest to the desired structure when wildlife habitat constraints are incorporated into the problem, the approach seems to be a useful tool to solve the habitat dispersion

problem. Even when optimal solutions are required, relaxation offers "good" bounds to adjust set the constraints of highly infeasible problems.

In order to follow the effect of different sets of constraints in the harvest levels we used the example forests depicted in figures 16 and 17 to obtain relaxation solutions for several planning horizons considering four combinations of constraints. Figures 18 and 19 show these results. In these figures each point represents a separate optimization solution.

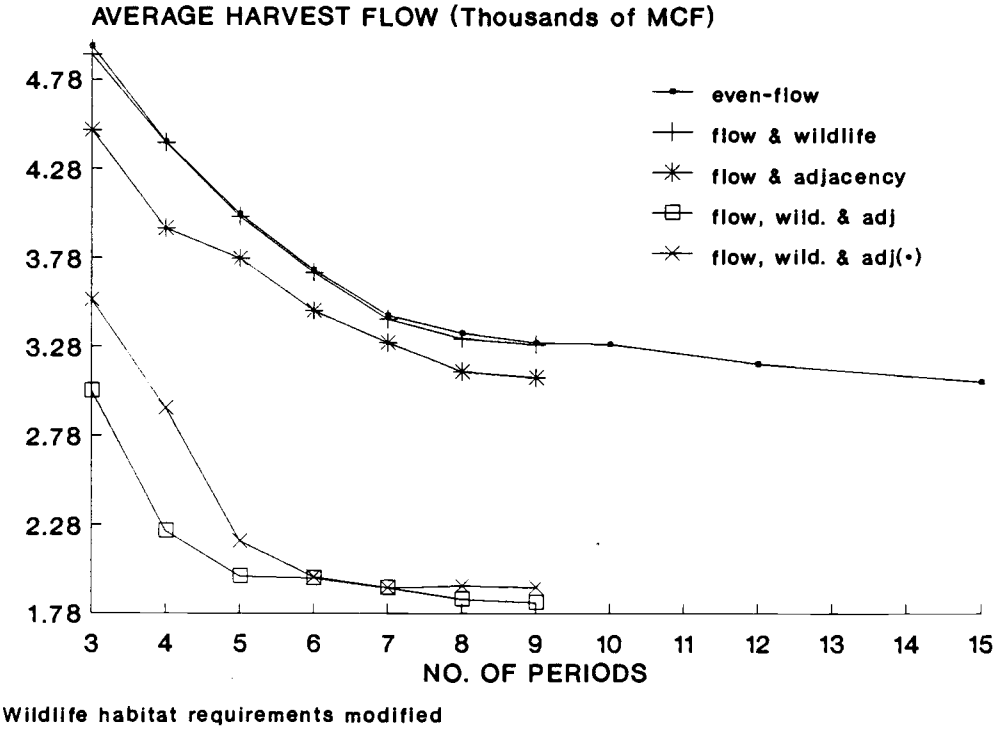


Figure 18. Harvest levels for different planning horizons in a 50 unit example forest under different sets of constraints.

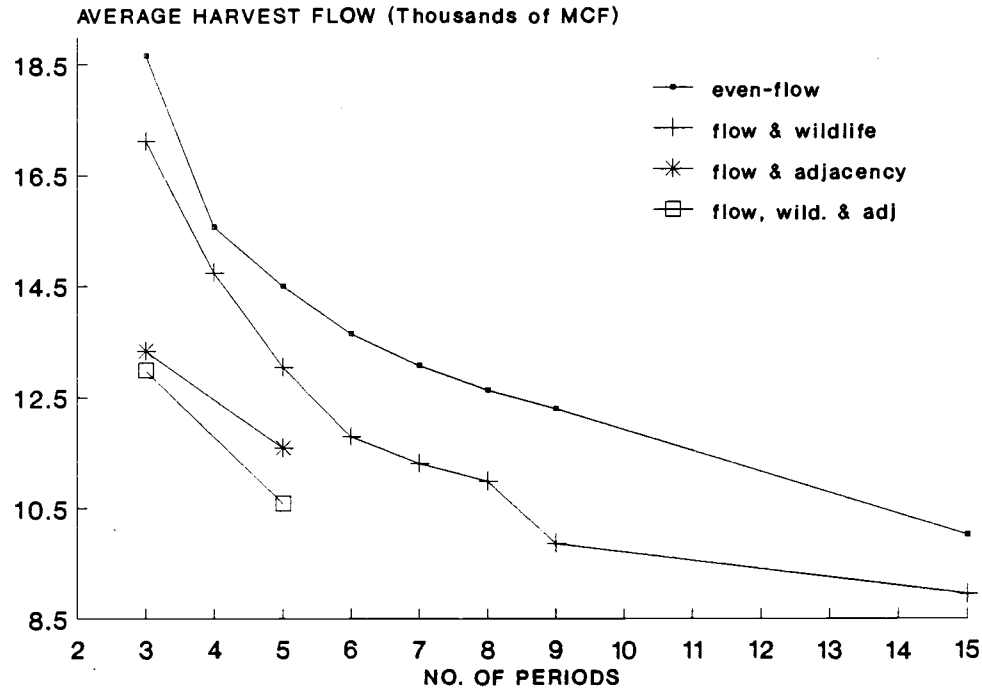


Figure 19. Harvest levels for different planning horizons in a 136 unit example forest under different sets of constraints.

Note that wildlife constraints alone do not have a strong effect in modifying the bounded harvest levels for the 50 unit forest (figure 18), although they affected the value of the objective function. For these examples the objective function value dropped in average 3 percent when wildlife constraints alone were considered. This behavior is probably due to the original age class distribution of the forest since its bell-shaped age class distribution fits into the wildlife constraints and permits a moderate harvest without altering the wildlife constraints. Hence the harvest flows without wildlife constraints are

maintained when these constraints are introduced. Observe that each point in the graph represents the average harvest for the specified number of periods. Hence such average harvest flow just guarantees the maintenance of wildlife habitat for that length of time. When the original age class distribution departs from the target wildlife distribution the wildlife constraints have a stronger effect on the harvest levels. The 136 harvest unit forest shows this behavior. Observe (figure 19) how wildlife constraints reduce the harvest levels an average of 17 percent as opposed to the 1 percent reduction in the 50 unit forest (figure 18). Regardless of the original age class distribution it is expected that in the long run wildlife habitat constraints will maintain a harvest level very close to the implied long term sustained yield (LTSY). Departures from the maximum LTSY will depend on the restrictions on the coverage of mature timber or forage requirements.

Adjacency constraints have a stronger effect in reducing the harvest levels. It is interesting to note in figure 18 that after the eighth period, when reharvests can be performed, solutions with just adjacency constraints yielded a harvest flow very close to the LTSY (20 periods) which might suggest that the pattern of harvests is repeated in the long run. It is expected that after the eighth period the fall down of the harvest level will be very smooth "converging" to the LTSY. However, we might

hypothetize that once all harvest units have been reharvested it would be very unlikely that the pattern of harvests repeats even with the harvest flow maintained in steady state. This means that the LTSY of a forest managed with spatial constraints will always be defined within certain bounds[2/]. Note that the only way a pattern of harvests is repeated in the long run is if the number of periods within the rotation age equals the chromatic number (the chromatic number is the minimum number of colors sufficient to color a planar map such that two contiguous areas do not have the same color) of the pattern of the harvest units in the forest and additionally all units are equally productive.

For problems with few planning periods the difference between the harvest flow with adjacency constraints and without them is highly related to the chromatic number of the pattern of the harvest units in the forest. Patterns with small chromatic numbers will have higher harvest flows than patterns with large chromatic numbers. This is basically due to the fact that small chromatic numbers permit more combinations of adjacency-feasible solutions

---

2/ Observe that if the number of periods within the rotation age is smaller than the chromatic number (irreal case) the long run harvest level for adjacency-constrained harvest problems will not converge to the LTSY.

than large chromatic numbers. Hence the adjacency constraints are less restrictive in the former case. The sequential approaches to solve the adjacency problem are less efficient for forests with patterns of harvest units with small chromatic numbers, simply because with an increased number of adjacency-feasible solutions the probability of reaching the "optimal" solution per period decreases since. The difference in harvest flows for problems with and without adjacency constraints also depends on the original scattering of the harvest units. If the original forest has groups of units of the same age (figure 17) instead of a random distribution (figure 16) the effect of the adjacency constraints is larger. This is due to the fact that small groups of same age harvest units quickly reduce the number of adjacency-feasible solutions. For instance, for a three period planning interval the reduction in harvest flow was 28 percent for the 136 unit forest while this reduction was just 9 percent for the 50 unit forest.

The combination of flow, wildlife, and adjacency constraints yielded unexpected results. Observe in figure 18 that the likely long run harvest level for problems with wildlife and adjacency constraints is reached at the 7th or 8th period with very low harvest levels. Note that only the harvest levels for a three period problem are close to the LTSY harvest level. The rest are very low and set about 33 percent lower than the LTSY for the 50 unit forest. A 5-

period solution for the 136 unit forest yielded a harvest flow 21 percent lower than the LTSY, although it is expected that for longer planning periods this percentage will increase.

These results indicate that "habitat dispersion" constraints are very restrictive. Wildlife habitat constraints per se just keep the minimum requirement of age classes. Hence most units are harvested a bit after the minimum harvest age. However, by introducing spatial objectives relatively large areas of over-mature timber are kept. This behavior could be attributed to the fact that the target distribution to meet wildlife requirements needs several age classes represented in the forest. By introducing spatial constraints to such distributions we decrease the number of regenerated harvest units, since the pattern of regeneration is modified. Thus the periodic harvest level is reduced and the harvest age is lengthened. To give an example, the solution for an 8-period planning interval with the 50 unit forest with just wildlife constraints permitted to harvest of all the stands, while the solution obtained by adding adjacency constraints left many units without harvest.

The likely long run harvest level with all habitat dispersion constraints will depend on the specified percentages of wildlife requirements and the chromatic number of the pattern of harvest units in the forest.

Hence these constraints completely drive the harvest.

Given that "very low" harvest levels are obtained when adjacency and wildlife constraints are introduced, more efficient mechanisms to include habitat dispersion objectives should be considered. Figure 18 shows how harvest levels can be increased just by reducing the lower bound of the thermal requirements by 5 percent. Similar strategies or procedures such as shorten the rotation ages to harvest middle aged timber, integration of LTSY target harvest levels with spatial concerns such as the approach by Nelson et al., 1989, or a more exact understanding of true habitat coverage needs could help to set better harvest levels and meet habitat dispersion objectives.

Judging form these examples, the combined effects of flow, wildlife and adjacency constraints may reduce harvest levels more than continuous models have led us to believe.

LITERATURE CITED

BALAS, E. and E. ZEMEL. 1980.

An algorithm for large zero-one knapsack problems. Opr. Res. 28(5):1130-1154.

BARE, B.B., B.H. FAALAND, and I. GUPTA. 1984.

Timber harvest scheduling in the presence of spatial constraints. Paper presented at the Joint National Meeting of IMS/ORSA. San Francisco CA, May 14-16.

DYER, M.E. 1980.

Calculating surrogate constraints. Math. Prog. 19(2):255-278.

GAVISH, B. and H. PIRKUL. 1985.

Efficient algorithms for solving multiconstraint zero-one knapsack problems to optimality. Math. Prog. 31(1):78-105.

GLOVER, F. 1975.

Surrogate constraint duality in mathematical programming. Opr. Res. 23(3):434-453.

GREENBERG, H. J. and W.P. PIERSKALLA. 1970.

Surrogate mathematical programming. Opr. Res. 18(4):924-939.

GROSS, T.E. and D.P. DYKSTRA. 1988

Harvest scheduling with nonadjacency constraints. In Proceedings, Society of American Foresters National Convention, October16-19, 1988. Wash. D.C. pp 310-315.

GUIGNARD, M. and S. KIM. 1987.

Lagrangean decomposition: a model yielding stronger Lagrangean bounds. Math. Prog. 39(2):215-228.

HOKANS, R.H. 1983.

Evaluating spatial feasibility of harvest schedules schedules with simulated stand selection decisions. J. For. 81(7):601-603, 613.

JOHNSON, K.N., and H.L. SCHEURMAN. 1977.

Techniques for prescribing optimal timber harvest and investment under different objectives: discussion and synthesis. For. Sci. Monograph No. 18.

KARWAN, M.H. and R.L. RARDIN. 1984

Surrogate dual multiplier search procedures in integer programming. Opr. Res. 32(1):52-69.

KIRBY, M. W., W. A. HAGER, and P. WONG. 1986

Simultaneous planning of wildland management and transportation alternatives. In System analysis in forestry and forest industries. Vol 21. Ed. M. Kalio, A.E. Andersson, R. Seppala, and A. Morgan. North-Holland/TIMS Studies in Management Science. Elsevier Pub. Co. New York. pp. 371-387.

KOWALIK, J. and M.R. OSBORNE. 1968.

Methods for unconstrained optimization problems. Elsevier Pub. Co.. New York.

MEALEY, S.P. and J.R. HORN. 1981.

Integrating wildlife habitat objectives into the forest plan. In Transactions of the 46th North American Wildlife and Natural Resources Conference. Ed. K. Sabol. Wash. D.C. March 21-25. pp. 488-500.

MEALEY, S.P., J.F. LIPSCOMB, and K.N. JOHNSON. 1982.

Solving the habitat dispersion problem in forest planning. In Transactions of the 47th North American Wildlife and Nat. Resources Conference. Ed. K. Sabol. Portland. OR March 26-31. pp.143-153.

MENEGHIN, B.J., M.W.KIRBY, and J.G. JONES. 1988.

An algorithm for writing adjacency constraints efficiently in linear programming models. In The 1988 Symposium on System analysis in Forest Resources, March 29 - April 1, Asilomar CA. USDA For. Serv. Gen. Tech. Rep. RM-161. pp 46-53.

NAUSS, R. M. 1976.

An efficient algorithm for the 0-1 knapsack problem. Mgmt. Sci. 23:27-31.

NELSON, J. 1988.

Integration of short-term spatially feasible harvesting plans with long-term harvest schedules using Monte-Carlo integer programming and linear programming. Ph. D. Dissertation, Dept. of Forest Management, Oreg. State Univ., Corvallis OR. 168 p.

NELSON, J., J.D. BRODIE, and J. SESSIONS. 1989.

Integrating short-term, area-based logging plans with long-term harvest schedules. (manuscript submitted to Forest Science).

NEMHAUSER, , G. L. and L. A. WOLSEY. 1988.

Integer and combinatorial optimization. John Wiley & Sons Inc. New York. 763 p.

O'HARA, A.J., B.H. FAALAND, and B.B. BARE. 1989.

Spatially constrained timber harvest scheduling. Can J. For. Res. 19(6):715-724

PIRKUL, H. 1987.

A heuristic solution procedure for the multiconstrained zero-one knapsack problem. Nav. Res. Log. Quart. 34(1):161-172.

ROISE, J. P. 1989.

Multicriteria nonlinear programming for optimal allocation of stands. For. Sci. (manuscript accepted for publication).

SALKIN, H.M. and C.A. DEKLUYVER. 1975.

The knapsack problem: a survey. Nav. Res. Log. Quart. 22(1):127-144.

SARIN, S., M.H. KARWAN, and R.L. RARDIN. 1987.

A new surrogate dual multiplier procedure. Naval Res. Log. 34(4)::431-450.

SESSIONS, J. 1988.

User's guide for Scheduling and Network Analysis Program (SNAP). Draft USDA For. Serv., Division of Timber Magmt., Portland, Oregon. 256 p.

THOMAS, J. W. 1979.

Wildlife habitats in managed forests: The blue mountains of Oregon and Washington. J. Thomas Editor. Agric. Handbook No. 553. USDA Wash. D.C.

THOMPSON, E.F., B. HALTERMAN, T. LYON and R. MILLER. 1973.

Integrating timber and wildlife management planning. For. Chronicle. 49(6):247-250

TORRES ROJO, J.M. and J.D. BRODIE. 1989.

A note on writing adjacency constraints in harvest scheduling. (manuscript submitted to Can. J. For. Res.).

TORRES ROJO, J.M., J.D. BRODIE. and J. SESSIONS. 1989.

Solution to the area-based harvest scheduling problem through Lagrangean relaxation (in journal review).

WEINTRAUB, A. 1982.

A heuristic approach for solving forest management transportation problems. Serie Documentos de Trabajo, Dpto. de Ingenieria Industral, Univ. de Chile. Pub. No. 5. Santiago, Chile. 19 p.

WEINTRAUB, A., R. EPSTEIN, and F. BARAHONA. 1988.

Integrating the habitat dispersion problem with timber management planning. Serie Documentos de Trabajo, Dpto. de Ingenieria Industrial, Univ. de Chile. Pub. No. 88/04/c. Santiago, Chile. 27 p.

YOSHIMOTO, A. and J.D. BRODIE. 1989.

A note on efficiency of algorithms to generate adjacency constraints (manuscript submitted to Forest Science).

CONCLUSIONS

This dissertation has presented the use of relaxation to solve area-based harvest scheduling problems. The pioneer work by Hoganson and Rose (1984) who addressed the non-declining even-flow harvest scheduling problem, has been redefined and expanded to include even-flow constriants and the basis for incorporating additional restrictions. The subgradient method has been introduced to compute the multipliers of the Lagrangean relaxation approach. This method with some modifications in the computation of the step size has proved to be efficient and better than the heuristics proposed by Hoganson and Rose (1984) or Elderd (1987). The Lagrangean relaxation approach yields good and quick solutions of harvest scheduling problems. Special constraints that must be met exactly have been handled through surrogate relaxation. This relaxation reduces vastly the size of the problem and also yields good solutions. Both relaxations were used to solve the habitat dispersion problem. Results yielded solutions within 2% of the optimal solution in a short time.

The solution procedure we presented for the habitat dispersion problem is efficient. However, it can be improved by developing a more effective procedure, perhaps a heuristic to solve the relaxed problem. Another source of future reasearch might be to improve the methodology to

find the surrogate multipliers, since if those multipliers are computed exactly the relaxed problems can be further reduced and the solution derived from those problems theoretically will be better (closer to the true optimum) and additionally could be obtained faster.

The relaxation approach could be expanded to use any other type of constraint that the common linear programming formulations of harvest scheduling problems can handle. Those constraints can be used to form a surrogate constraint or can be dualized in the objective function depending on whether or not small violations to those constraints can be incorporated into the planning process.

An extension of the relaxation approach applied to forest planning is the joint optimization of a road network and harvest schedules. The approach might require some sort of resource decomposition approach to reduce the size of the problem before applying the relaxation. This procedure might provide "good solutions" for those so far unsolvable problems.

If the planning of thinning activities are important in a forest level analysis, relaxation might be a good solution tool. Thinning activities might be incorporated in a Model I formulation of the harvest scheduling problem and treated as other activities for the same harvest units. Then the solution approach would follow the same strategy we recommended in this dissertation. More sophisticated approaches might include the use of dynamic programming in

conjunction with relaxation to derive thinning schedules for each harvest unit that satisfies forest wide requirements. This procedure could be a good approach for tactical planning (only a few periods in the optimization) since dynamic programming and relaxation can be used not only to solve the scheduling problem with additional side constraints but also the road network might be optimized in the sequential optimization performed by dynamic programming.

Given the flexibility of relaxation to incorporate many problem structures its use to solve harvest scheduling problems offers much future potential. This work represents a beginning effort in the use of these techniques to provide integer area-based planning analogous to current continuous models.

BIBLIOGRAPHY

ALLEN, E., R. HELGASON, J. KENNINGTON, and B. SHETTY. 1987. A generalization of Polyak's convergence result for subgradient optimization. Math. Programming 37(3):309-317.

APPEL, K.I. and W. HAKEN. 1976. Every planar map is four colorable. Bull. Am. Math. Soc. 82:711-712.

BALAS, E. and E. ZEMEL. 1980. An algorithm for large zero-one knapsack problems. Opr. Res. 28(5):1130-1154.

BARE, B.B., B.H. FAALAND, and I. GUPTA. 1984. Timber harvest scheduling in the presence of spatial constraints. Paper presented at the Joint National Meeting of TIMS/ORSA. San Francisco CA, May 14-16.

BARKER, B.R. 1989. Utilizing scheduling and network analysis program (SNAP) as a forest plan implementation tool on the Siuslaw National Forest. MS thesis; College of Forestry. O. S. U. Corvallis Oregon. 76 p.

BAZARAA, M.S. and C.M. SHETTY. 1979. Nonlinear programming: Theory and algorithms. John Wiley & Sons. 560 p.

BERCK, P. and T. BIBLE. 1984.

Solving and interpreting large-scale harvest scheduling problems by duality and decomposition. For. Sci. 30(2):173-182.

BERTSEKAS, D.P. 1982.

Constrained optimization and Lagrange multipliers methods. Academic Press. London. 395 p.

BRADLEY, G.H. 1971.

Transformation of integer programs to knapsack problems. Discrete Mathematics 1:29-45.

BRODIE, J.D., H.C. BLACK, E.J. DIMOCK II, C. KAO, and J.A. ROCHELLE. 1979.

Animal damage to coniferous plantations in Oregon and Washington:part II an economic evaluation. Forest Research Laboratory, O. S. U. Corvallis OR. Research Bull No. 26. 22p.

CAO, Q.V., H.E. BURKHART, and R.C. LEMIN. 1982.

Diameter distributions and yields of thinned loblolly pine plantations. School of For. and Wildlife Res., V.P.I. and State Univ. Publ. FWS-82,62p.

DREYFUS, S.E. 1957.

Computational aspects of Dynamic Programming. Opns. Res. 11(4):399-417.

DYER, M.E. 1980.

Calculating surrogate constraints. Math. Prog. 19(2):255-278.

ELDRED, P. 1987.

Evaluation of a shadow price search heuristic as an alternative to linear programming or binary search for timber harvest scheduling. Unpublished MS Thesis, College of Forestry. Oregon State University. 64 p.

EVERETT III, H. 1963.

Generalized Lagrange multipliers method for solving problems of optimum allocation of resources. Opr. Res. 11(3):399-347.

FIGHT, R.D., C.B. LEDOUX and T.L. ORTMAN. 1984.

Logging costs for management planning for young-growth coast Douglas-fir. USDA For. Serv. Pac. Northwest For. and Range Exp. Stn. Gen. Tech. Rep. PNW-176. 10p.

FISHBURN, P.C. and G.A. KOCHENBERG. 1985.

Aggregating assignment constraints. Nav. Res. Log. Quart. 32(4):653-663.

FISHER, M.L. 1981.

The Lagrangian relaxation method for solving integer programming problems. Mgmt. Sci. 27(1):1-18.

FISHER, M.L. 1985.

An applications oriented guide to Lagrangian relaxation. Interfaces 15(2):10-21.

FISHER, M.L., W.D. NORTHUP and J.F. SHAPIRO. 1975.

Using duality to solve discrete optimization problems. Math. Programming study No. 3. pp 56-94.

FISHER, M.L., JAIKUMAR, R. and VAN WASSENHOVE L. 1980.
A multiplier adjustment method for the generalized assignment problem. Dec. Sci. Working paper. Univ. of Pennsylvania.

GAVISH, B. and H. PIRKUL. 1985.
Efficient algorithms for solving multiconstraint zero-one knapsack problems to optimality. Math. Prog. 31(1):78-105.

GEOFFRION, A.M. 1974.
Lagrangean relaxation for integer programming. (Ed. M. L. Balinski), North-Holland/Amer. Elsevier. Mathematical Programming Study No. 2. pp 82 -114.

GLOVER, F. 1975.
Surrogate constraint duality in mathematical programming. Opr. Res. 23(3):434-453.

GOFFIN, J.L. 1977.
On convergence rates of subgradient optimization methods. Math. Programming study No. 13. pp 329-347.

GREENBERG, H. J. and W.P. PIERSKALLA. 1970.
Surrogate mathematical programming. Opr. Res. 18(4):924-939.

GROSS, T.E. and D.P. DYKSTRA. 1988.
Harvest scheduling with nonadjacency constraints. In Proceedings, Society of American Foresters National Convention, October 16-19 1988. Wash. D. C. pp 310-315.

GUIGNARD, M. and S. KIM. 1987.

Lagrangean decomposition: a model yielding stronger Lagrangean bounds. Math. Prog. 39(2):215-228.

HELD, M. and R.M. KARP. 1970.

The traveling salesman problem and minimum spanning trees. Opr. Res. 18(4):1138-1162.

HELD, M. and R.M. KARP. 1971.

The traveling salesman problem and minimum spanning trees: part II. Math. Programming 1(1):6-25.

HELD, M., P. WOLFE, and H.P. CROWDER. 1974.

Validation of subgradient optimization. Math. Programming, 6(1):62-88.

HOGANSON, H.M. and D.W. ROSE. 1984.

A simulation approach for optimal timber management scheduling. For. Sci. 30(2):220-238.

HOKANS, R.H. 1983.

Evaluating spatial feasibility of harvest schedules schedules with simulated stand selection decisions. J. For. 81(7):601-603, 613.

IBARAKI, T. and N. KATOH. 1988.

Resource allocation problems. Algorithmic approaches. Found. of Computing Series. MIT Press. Cambridge, Massachusetts. 229 p.

JOHNSON, K.N. and H.L. SCHEURMAN. 1977.

Techniques for prescribing optimal timber harvest and investment under different objectives: Discussion and synthesis. For. Sci. Monograph No. 18. 31 p.

KANNAN, R. 1983.

Polynomial time aggregation of integer programming problems. J. Ass. of Computing Machinery. 30(1):133-145.

KARWAN, M.H. and R.L. RARDIN. 1984.

Surrogate dual multiplier search procedures in integer programming. Opr. Res. 32(1):52-69.

KIRBY, M. W., W. A. HAGER, and P. WONG. 1986.

Simultaneous planning of wildland management and transportation alternatives. In System analysis in forestry and forest industries. Vol 21. Ed. M. Kalio, A.E. Andersson, R. Seppala, and A. Morgan. North-Holland/TIMS Studies in Management Science. Elsevier Pub. Co. New York. pp. 371-387.

KOWALIK, J. and M.R. OSBORNE. 1968.

Methods for unconstrained optimization problems. Elsevier Pub. Co.. New York.

LORIE, J.N. and L.J. SVAGE. 1955.

Three problems in rationing capital. J. of Business. 28(4):229-239.

MEALEY, S.P. and J.R. HORN. 1981.

Integrating wildlife habitat objectives into the forest plan. In Transactions of the 46th North American Wildlife and Natural Resources Conference. Ed. K. Sabol. Wash. D.C. March 21-25. pp. 488-500.

MEALEY, S.P., J.F. LIPSCOMB, and K.N. JOHNSON. 1982.

Solving the habitat dispersion problem in forest planning. In Transactions of the 47th North American Wildlife and Nat. Resources Conference. Ed. K. Sabol. Portland. OR March 26-31. pp.143-153.

MENEGHIN, B.J., M.W. KIRBY, and J.G. JONES. 1988.

An algorithm for writing adjacency constraints efficiently in linear programming models. The 1988 Symp. on Systems analysis in Forest Resources. March 29-April 1 1988. Gen. Tech. Rep. RM-161 Rocky Mtn For. & Ran. Exp. Stn., Ft. Collins Colorado.

MEYER, G.G.L. 1987.

Convergence of relaxation algorithms by averaging. Math Programming 40(2):205-212.

NAUSS, R. M. 1976.

An efficient algorithm for the 0-1 knapsack problem. Mgmt. Sci. 23:27-31.

NELSON, J. 1988.

Integration of short-term spatially feasible harvesting plans with long-term harvest schedules using Monte-Carlo integer programming and linear programming. Ph. D. Dissertation, Dept. of Forest Management, Oreg. State Univ., Corvallis OR. 168 p.

NELSON, J., J.D. BRODIE and J. SESSIONS. 1988.

Integrating short term spatially feasible harvest plans with long term harvest schedules using Monte-Carlo integer programming and linear programming. In: The 1988

Symp. Syst. Analysis on For. Res. (B. Kent and L. Davis Ed.) U.S.D.A. For. Serv. Gen. Tech. Rep. RM-161. pp. 224-229.

NEMHAUSER, , G. L. and L. A. WOLSEY. 1988.

Integer and combinatorial optimization. John Wiley & Sons Inc. New York. 763 p.

O'HARA, A.J., B.H. FAALAND, and B.B. BARE. 1989.

Spatially constrained timber harvest scheduling. Can J. For. Res. 19(6):715-724.

ONEYKWELU, D.C. 1983.

Computational viability of a constraint aggregation scheme for integer linear programming problems. Opr. Res. 31(3):795-801.

PADBERG, M.W. 1972.

Equivalent knapsack-type formulations of bounded integer linear programs: an alternative approach. Nav. Res. Log. Quart. 19(4):699-708.

PAREDES VELOSO, G.L. and J.D. BRODIE. 1986.

Efficient specification and solution of the even-aged rotation and thinning problem. For. Sci. 33(1):14-29.

PIRKUL, H. 1987.

A heuristic solution procedure for the multiconstrained zero-one knapsack problem. Nav. Res. Log. Quart. 34(1):161-172.

POLYAK, B.T. 1969.

Minimization of unsmooth functionals. USSR Computational Mathematics and Mathematical Physics, 9(3):14-29

RINGEL, G. 1959.

Farbungsprobleme auf Flachen und Graphen. Mathematische Monographien Numer 2. Veb Deutscher Verlag der Wissenschaften. Berlin, 132 p.

ROISE, J. P. 1989.

Multicriteria nonlinear programming for optimal allocation of stands. For. Sci. (manuscript accepted for publication).

SAATY, T.L. and P.C. KAINEN. 1977.

The four-color problem. McGraw-Hill Inc. Great Britain. 217 p.

SALKIN, H.M. and C.A. DEKLUYVER. 1975.

The knapsack problem: a survey. Nav. Res. Log. Quart. 22(1):127-144.

SARIN, S., M.H. KARWAN, and R.L. RARDIN. 1987.

A new surrogate dual multiplier procedure. Naval Res. Log. 34(4)::431-450.

SESSIONS, J. 1988.

User's guide for Scheduling and Network Analysis Program (SNAP). Draft U.S.D.A. For. Serv., Division of Timber Mgmt., Portland, Oregon. 256 p.

SHAPIRO, J.F. 1971.

Generalized Lagrange multipliers in integer programming. Opr. Res. 19(1):68-76.

SHAPIRO, J.F. 1979.

A survey for Lagrangean techniques for discrete optimization. Ann. Discrete Math. Vol. 5. pp. 113-138.

SHOR, N.Z. 1964.

On the structure of algorithms for the numerical solution of optimal planning and design problems. Dissertation Cybernetics Institute, Academy of Sciences USSR.

THOMAS, J. W. 1979.

Wildlife habitats in managed forests: The blue mountains of Oregon and Washington. J. Thomas Editor. Agric. Handbook No. 553. USDA Wash. D.C.

THOMPSON, E.F., B. HALTERMAN, T. LYON and R. MILLER. 1973.

Integrating timber and wildlife management planning. For. Chronicle. 49(6):247-250

TORRES R., J.M. and J.D. BRODIE. 1989.

A note on writing adjacency constraints in harvest scheduling. (manuscript submitted to Can. J. For. Res.).

TORRES R., J.M., J.D. BRODIE. and J. SESSIONS. 1989.

Solution to the area-based harvest scheduling problem through Lagrangean relaxation (in journal review).

TSENG, P. and D.P. BERTSEKAS. 1987.

Relaxation methods for linear programs. Math. Opr. Res. 12(3):569-596.

WEINTRAUB, A. 1982.

A heuristic approach for solving forest management transportation problems. Serie Documentos de Trabajo, Dpto. de Ingenieria Industral, Univ. de Chile. Pub. No. 5. Santiago, Chile. 19 p.

WEINTRAUB, A., R. EPSTEIN, and F. BARAHONA. 1988.

Integrating the habitat dispersion problem with timber management planning. Serie Documentos de Trabajo, Dpto. de Ingenieria Industrial, Univ. de Chile. Pub. No. 88/04/c. Santiago, Chile. 27 p.

WOOD, D.C. 1969.

A technique for colouring a graph applicable to large scale timetabling problems. Computer Journal 12:317-319.

YOSHIMOTO, A. and J.D. BRODIE. 1989.

A note on efficiency of algorithms to generate adjacency constraints (manuscript submitted to Forest Science).