Multiple Solution Harvest Scheduling

Paul C. Van Deusen

Van Deusen, P.C. 1999. Multiple solution harvest scheduling. Silva Fennica 33(3): 207–216.

Application of the Metropolis algorithm for forest harvest scheduling is extended by automating the relative weighting of objective function components. Previous applications of the Metropolis algorithm require the user to specify these weights, which demands substantial trial and error in practice. This modification allows for general incorporation of objective function components that are either periodic or spatial in nature. A generic set of objective function components is developed to facilitate harvest scheduling for a wide range of problems. The resulting algorithm generates multiple feasible solutions rather than a single optimal solution.

Keywords Metropolis algorithm, simulated annealing, Gibb's sampler
Author´s address Principal Research Scientist, NCASI, Northeast Regional Center, Tufts University, 1 Anderson Hall, Medford, Massachusetts 02155. USA
Fax 617-627-3831 **E-mail**:pvandeus@tufts.edu
Received 11 March 1999 **Accepted** 27 July 1999

1 Introduction

Forest land managers need flexible planning tools that allow them to identify management schedules for their holdings that meet a wide range of objectives. Typically, they want to obtain an even flow of goods over time while simultaneously meeting economic and spatial requirements related to clearcut blocksize and wildlife habitat. In fact, most industrial forest land owners in the United States have agreed to follow a set of voluntary sustainable forestry initiatives (SFI) developed by the American Forest and Paper Association (AF&PA). These initiatives include restrictions on clearcutting until adjacent areas have greened-up, and making efforts to enhance and provide an array of wildlife habitats. Individual States may also have regulations that add spatial requirements in addition to the AF&PA regulations, e.g. California, Maine, Oregon, and Washington.

A number of algorithms have been proposed for handling harvest scheduling with adjacency constraints. For example, solutions can be provided by integer or mixed-Integer programming (MIP) for problems that are not too large (Meneghin et al. 1988, Torres-Rojo and Brodie 1990, Jones et al. 1991, Yoshimoto and Brodie 1994, Snyder and ReVelle 1996). Larger problems can be dealt with by using multi-year cutting periods (Carter et al. 1997), but AF&PA green-up constraints are best handled with single-

year periods. Linear programming (LP) solutions are often used as a first step when adjacency constraints are required, and the LP solution is subsequently massaged to integer values by various heuristic algorithms (Jamnick and Walters 1993, Weintraub et al. 1994, Carroll et al. 1995). MIP approaches have also been demonstrated for habitat scheduling applications (Hof and Joyce 1993, Hof and Raphael 1993, Hof et al. 1994) for small problem sizes.

A number of published studies have used simulated annealing (SA) for harvest scheduling. The Metropolis algorithm (MA) forms the basis for SA, which is a procedure for slowly coercing MA to converge to a single solution by adjusting a temperature parameter. SA and MA have been applied to harvest scheduling without the traditional focus on finding an optimal financial solution (Lockwood and Moore 1993, Van Deusen 1996). However, SA has also been used where the goal is to locate a financial optimum (Murray and Church 1995, Tarp and Helles 1997). The algorithm proposed here does not focus on optimization, but incorporation of financial objectives will be specifically discussed and demonstrated.

Tabu search (TS) has been used to solve harvest scheduling problems with spatial constraints. TS allows for great flexibility relative to incorporating problem specific knowledge (Glover and Laguna 1993), and applications have demonstrated that TS works well (Bettinger et al. 1997, 1998) for harvest scheduling. Other work suggests that TS has methodological flaws (Mayer et al. 1998) that limit its applicability to higher dimensional problems such as harvest scheduling. Regardless, a comparison of SA and TS (Murray and Church 1995) suggested that TS often produced better solutions than SA. However, this application used a very traditional and restricted form of SA where the temperature parameter is monotonically decreased. More flexible control of temperature can greatly improve the performance of SA (Osman 1993), which is analogous to what is being proposed here. An algorithm is developed that can flexibly meet the user's objectives, yet doesn't require problem specific knowledge to be successful.

The basic management unit used here is a polygon, which can include stands of trees as well as ponds, fields, streams, or riparian areas. A polygon rather than forest-stand based approach is important, since proximity to water and other resources could be a necessary consideration. A management schedule involves assigning a specific management regime to each of N polygons. Regimes are discrete, include the actions and years when they occur, and can be quite numerous. For example, clearcutting a stand in year 1 is a different regime from clearcutting in year 2. The intent is to assign management regimes to polygons such that the users goals or constraints are met in a nearly optimal fashion. There are T years in the planning horizon, where T is determined by when the last output or activity will occur under any of the user supplied regimes.

The next section presents an overview of the proposed algorithm, which is related to Markov chain Monte Carlo methods (MCMC) (Geman and Geman 1984, Besag 1986, and Besag et al. 1995) and SA (Lockwood and Moore 1993). The algorithm uses methods similar to some previously advocated (Lockwood and Moore 1993, Van Deusen 1996) but includes important practical enhancements. The generic objective function and solution algorithm are described, example objective function components are developed, and an example application demonstrates the utility of the approach.

2 Objective Function

The algorithm is intended to seek management schedules that are feasible and produce nearly minimal values of the objective function. Feasible schedules are those that satisfy the users goals. The solution evolves iteratively, and the value of the objective function at iteration r is

$$E(X^r) = \sum_{j=1}^{J} w_j^{r-1} C_j(X^r) \quad (1)$$

where X^r denotes the management schedule at iteration r, w^r_j is the weight based on the iteration r schedule, and $C_j(X^r)$ is the jth objective function component whose value is evaluated at the rth schedule. The vector, X^r, contains the list of regimes assigned to polygons 1 through N, $x_1{}^r$, ..., $x_N{}^r$.

The weights are re-evaluated after each iteration based on user defined goal functions, g_j, for each component. Weights are adjusted as follows:

if $g_j(X^r) > U_j$ then $w_j^r = a w_j^{r-1}$,
if $g_j(X^r) < L_j$ then $w_j^r = w_j^{r-1}/a$,

where U and L are upper and lower limits defined for each component and a is an adjustment factor between 0 and 1. Therefore, the component weight is decreased if it's goal is exceeded or increased if the goal isn't attained. There is also a level of attainment for which the weight is deemed to have converged and is left unchanged, i.e. between U and L. The goal functions are evaluated at the current schedule, X^r.

3 Algorithm

The method for obtaining solutions to objective function (1) is the Metropolis et al. (1953) algorithm. The specific algorithm recommended is

(1) Choose an objective function in the form of (1), and an adjustment factor $0 < a < 1$.
(2) Initialize X^1 by choosing x_i for each polygon at random, let $w^0_j = 1$ for $j = 1,..,J$, and set $r = 0$,
(3) Increment r and for polygon i from 1 to N:
  a) Perturb X^r into Z by choosing regime $k \in \{1,...,K\}$ at random for polygon i. Hold all polygons other than i at their current regimes.
  b) Let $p^* = \min \{1, \exp[E(X^r) - E(Z)]\}$
  c) Replace x_i by k with probability p^*.
(4) Evaluate the goal functions $g_1,...,g_J$ and adjust the weights, w_j^r accordingly.
(5) Repeat (3) and (4) until the weights have converged or you deem the problem infeasible.

After the weights have converged, step 3 of the Metropolis algorithm can be repeated to generate many feasible schedules that could be evaluated for conditions not controlled by the objective function. If the weights won't converge the problem is infeasible as currently stated, and the limiting goals must be made less restrictive. The user will normally understand their problem well enough to know which goal is limiting, otherwise some trial and error is required to change the goal limits. An important aspect of this algorithm is that evaluating $E(X) - E(Z)$ requires only a few computations involving polygon i and its neighbors. Furthermore, the computations can be performed independently for each objective function component. When the objective function becomes smaller as a result of changing polygon i to regime k (Step 3a), then this change is automatically accepted. When $E(Z)$ is larger than $E(X)$, the move to regime k may still be accepted with probability p^*, which helps prevent the algorithm from getting stuck at local minima.

4 Objective Function Components

Objective function components can generally be categorized as pertaining to periodic or spatial issues. Flow components are commonly used and are clearly periodic, whereas a component to control maximum blocksize is spatial. There might also be a need for space-time components. For example, a component that controls habitat output over time could depend on assignment of schedules within neighboring blocks of pixels.

Rather than dwell on theoretical discussion about components, some generic examples of periodic and spatial components are developed. The handful of example components developed below are sufficient to solve a rather sophisticated scheduling problem and are subsequently used in the example application. However, it should be clear that components to serve many other purposes could be developed by following the same recipe.

The Metropolis algorithm requires only the change in the objective function component, $\Delta C_j(i)$, that would occur if a single polygon's regime is changed from it's current value, x_i, to a proposed value, z. For some components, it is easier and clearer to present $C_j(X^r)$ and for others to present $\Delta C_j(i)$.

5 Periodic Components

A typical flow component can be written as follows:

$$C_j(X^r) = \sum_{t=1}^{T} (y_t - \hat{y}_t)^2 / F \quad (2)$$

where y_t represents the total output of some good at time t from all polygons, $\hat{y}_t$ is the target output for time t, there are T time periods in the schedule, and F is a scaling factor. F serves to scale C_j so that similar weights are effective regardless of the units that y is measured in. Setting F equal to the mean sum of the squares of target values serves as a good scaling factor in most cases. The details of computing $\Delta C_j(i)$ for this component can be determined by the programmer. The issue of how to provide a target value is discussed below.

Each component needs associated goals, and 2 goal functions are suggested for flow components:

$$g1(t) = 1 - \min\left(\frac{\text{abs}(y_t - \hat{y}_t)}{\hat{y}_t + \delta}, 1 \right) \; t = 1,...,T \quad (3)$$

and

$$g2(t) = 1 - \min\left(\frac{\text{abs}(y_t - \hat{y}_{t-1})}{\hat{y}_{t-1} + \delta}, 1 \right) \; t = 2,...,T \quad (4)$$

where δ is a small positive number to prevent dividing by 0. Goal 1, $g_1(t)$, controls deviations from the target values, and goal 2, $g_2(t)$, controls year-to-year deviations. As an example, one might set the limits for both goals to be: U = .9 and L = .8. Therefore, if both goals are above U at the end of a Metropolis iteration, the component weight (w_j) in equation (1) is decreased. If either goal is below L then the weight is increased. This ensures that both goals are attained at least to the lower limit standard and avoids any unresolvable conflicts between the 2 goals. The goals are also constrained to be between 0 and 1, with 0 implying no attainment and 1 meaning full attainment.

Numerous methods could be used to determine the target value for y_t. For example, Van Deusen (1996) suggested smoothing equations and Lockwood and Moore (1993) used predefined values. Also, the absolute value of deviations rather than squared deviations could be used as the basis for the periodic component.

6 Spatial and Non-periodic Components

Spatial issues are the driving force behind the need for new harvest scheduling algorithms. Possibly the most important spatial issue is the control of clearcut blocksize. A component for this purpose can be written as:

$$C_j(X^r) = \sum_{i=1}^{N} I(\text{minSize} < \text{Block}_i < \text{maxSize}) \quad (5)$$

where $I(.) = 0$ when the size of the block containing polygon i is between minSize and maxSize, and $I(.) = 1$ otherwise. The algorithm for computing blocksize and the constants minSize and maxSize must be defined by the user. For this component, $\Delta C_j(i) = 0$ when the proposal regime leads to a spatial configuration that is equal to the current configuration, otherwise $\Delta C_j(i) = 1$ if the proposal is better than the current regime, or $\Delta C_j(i) = -1$ if the current regime is better.

An appropriate goal function is:

$$g_j = 1 - \sum_{i=1}^{N} \frac{I(\text{minSize} < \text{Block}_i < \text{maxSize})}{N} \quad (6)$$

which gives the proportion of polygons that are contained within conforming blocks. Now upper and lower bounds to force absolute conformance to the blocksize limits would be $U = 1$ and $L = 1$. The weight is increased for the blocksize component whenever any blocks are non-conforming, which is basically the approach used in Van Deusen (1996). Lockwood and Moore used a method that progressively discourages blocksizes as they get larger and also discourages blocksizes below a certain limit. Lockwood and Moore's component could be used here as well, but the algorithm would automatically seek the appropriate weight based on user specified goals.

A second non-periodic component that is quite useful could be termed a suitability component. Van Deusen (1996) presented a biological component that is closely related, and Lockwood and Moore's component for penalizing harvest of stands with low volume per area ratio is similar in spirit. Begin by considering a suitability index to rank polygon i's relative suitability for each management option, say $s_{i1},...,s_{iK}$. The effect of changing polygon i's regime for this component is

$$\Delta C_j(i) = -I(s_{ix} > s_{iz}) + I(s_{iz} > s_{ix}) \quad (7)$$

where $\Delta C_j(i) = 1$ when the proposal schedule is more desirable than the current schedule, $\Delta C_j(i) = -1$ when it is less desirable, and $\Delta C_j(i) = 0$ when they are equal.

The suggested goal function is

$$g_j = \frac{1}{N}\sum_{i=1}^{N} I\left(s_{ix} \geq s_{id}\right) \quad (8)$$

which gives the proportion of polygons currently managed under an option that is at least of rank d, where d is specified by the user. Asymptotically, this biases polygon assignments toward the more suitable options. Reasonable upper and lower limits might be $U = .8$ and $L = .7$. As usual, the weight is increased if $g_j < L$ and decreased if $g_j > U$.

The goal function for this component can also be modified to facilitate maximizing $\sum s_{ix}$. For example, s_{ix} might be the net present value (NPV) that results from assigning regime x to polygon i. In this case, the goal function should be:

$$g_j = \frac{\sum_{i=1}^{N} s_{ix}}{\sum_{i=1}^{N} s_{i1}} \quad (9)$$

where s_{i1} represents the highest ranking regime for polygon i. Therefore, g_j gives the ratio of the total achieved by the current schedule relative to the completely unconstrained schedule.

7 Optimal Solution

After the weights have converged, the proposed algorithm will continue to run and produce variations on previous solutions that will each be feasible. However, the user might want to find a schedule that is near optimal in some sense. Simulated annealing depends on constructing a "cooling schedule" with a parameter, Q, such that step 3b of the Metropolis algorithm involves evaluating $\exp[E(X) - E(Z)]^{1/Q}$. As $Q \to \infty$, the posterior distribution (Geman and Geman 1984) represented by exp(.) becomes uniform over the space of all possible management schedules, which means that the objective function would have no influence and the schedules being generated would be completely random. As $Q \to 0$ the distribution is concentrated over the estimate of X that minimizes the objective function. The cooling schedule idea is analogous to cooling metals slowly so that they form a good crystalline structure. Besag (1986) points out that SA can reach the optimal solution, but can not detect when it has done so.

The optimal solution from a harvest scheduling perspective is the solution that maximizes some quantity subject to a number of constraints. As such, the objective function presented here and elsewhere (Lockwood and Moore 1993, Van Deusen 1996) will not yield the desired optimal solution when SA is applied in the traditional way. However, SA can be applied by increasing the weight on one objective function component while holding the other weights constant. This will bias MA toward solutions that emphasize the selected component while still enforcing the "constraints" imposed by the components whose weights remain fixed. This is demonstrated in the example application by creating a suitability component (equation 7) where the ranking is based on NPV and using goal function (9). While holding other component weights fixed, the weight on the NPV component can be slowly increased by increasing the target goal to find solutions with ever larger NPV that also meet the other component goals. This is simply a form of SA that is directed toward finding the desired optimal solution. This approach to SA is implemented in the example application by increasing the goal slowly and interactively as the algorithm runs. The weight on the associated component then increases until the goal is attained or other components can no longer attain their goals, which indicates infeasibility.

The proposed algorithm also makes it easy to solve harvest scheduling problems where maximization over a single quantity is not the objective. There are government agencies and companies that don't seek to maximize NPV as their scheduling objective. For example, some companies want to focus on cutting their stands as close as possible to the time of maximum mean annual increment (Pers. commun.: Steve Prisley, Westvaco Corporation). This can be accomplished with objective function component (7) and goal function (8).

8 Convergence of the Algorithm

The issue of when and if the proposed management scheduling algorithm has converged is considered here. The discussion refers to the MA stage and the adaptive determination of objective function weights. First, use of MA implies that concern lies with convergence in distribution rather than convergence to an optimal solution. MCMC theory (Geman and Geman 1984, Besag et al. 1986) assures us that MA will converge so that it generates samples from the target distribution after enough iterations have passed. The target distribution of interest here is the distribution that generates feasible schedules – one of which is the so called optimal solution. However, it may not be clear when the algorithm has converged, just as it is unclear when SA has found the optimal solution. Slowly changing weights after each iteration is very similar to the cooling schedule concept from SA. Given enough time, MA is known to converge (Geman and Geman 1984) at each new setting of the cooling schedule parameter. Therefore, the algorithm will converge for different objective function component weights for a feasible problem, and the example application supports this contention.

As the algorithm proceeds, it makes sense to monitor output summaries to aid in determining when convergence has occurred. Meaningful summaries would include graphs of flow vs. year, average blocksize vs. year, and minimum and maximum blocksize vs. year. Also, monitoring the weights as they are adjusted each iteration is helpful. After convergence, the summary graphs and the weights should remain relatively constant from one iteration to the next. Determining whether the algorithm has truly converged in distribution is not as relevant for this application as for the usual statistical applications of MCMC. Practically, if the summary statistics and weights have converged, the algorithm has converged.

Formal proofs related to convergence in distribution of MA rely on the concept of irreducibility (York 1992). Simply stated, the scheduling problem is irreducible if for any states x and x^r such that $P(X = x) > 0$ and $P(X = x^r) > 0$, there is a positive probability that x^r can be reached from x in a finite number of transitions. However, it may be difficult to prove that any given scheduling problem is irreducible. Problems involving adjacency constraints are particularly troublesome in this respect. In particular, a large polygon could prevent its neighbors from being cut if their combined area exceeds a blocksize limit. To prevent this type of restriction on irreducibility, each polygon should have a do-nothing option.

9 Example Application

The example uses a basic harvest scheduling data set consisting of 419 loblolly pine stands from the southeastern US. However, this algorithm can handle much larger problems. The size of the stands ranges from 2 to 200 acres (1 ha = 2.47 ac) with a mean of around 30 acres. Stands have from 0 to 9 neighbors and are generally young plantations (< 25 years old). Management regimes (options) 1 through 15 are to clearcut the stand in years 1 through 15 and option 16 is to do nothing with the stand during the 15 year planning period. The objective function contains 2 flow components, a blocksize component, and a suitability component as outlined above. The first flow component controls the flow of tons of wood, and the second controls the amount of area clearcut each year. The suitability component biases results toward assigning regimes that yield the highest possible NPV. The adjustment parameter for component weights is a = 0.9. For a-values closer to 1, the algorithm converges more slowly, whereas *a*-values closer to 0 produce more rapid weight changes. Setting $a < 0.8$ caused convergence problems in some cases, but $a = 0.9$ seems to work for a wide range of problems.

The green-up window for blocks is set to 2 years, so a stand cut in the year 2000 would be greened-up starting in 2003. The blocksize goal was to keep all blocks less than size 180 and greater than 20. Area was measured in acres. Flow was measured in tons per year.

Flow was allowed to deviate within 15 to 25 percent from year-to-year as well as around the target. A model was used to set a target flow that increased by a specified rate, which for this example was 2 percent per annum. Year 0 flow was specifically set to 140 000 tons and the goal

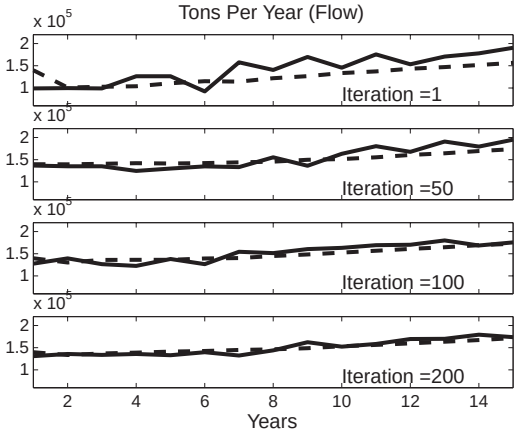

Fig. 1. Actual flow (solid line) and target flow (dashed line) for iterations 1, 50, 100, and 200 of the Metropolis algorithm.

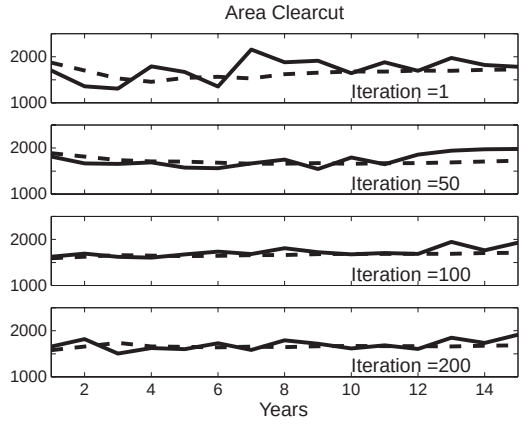

Fig. 2. Actual area clearcut each year (solid line) and target area (dashed line) at designated iterations of the Metropolis algorithm.

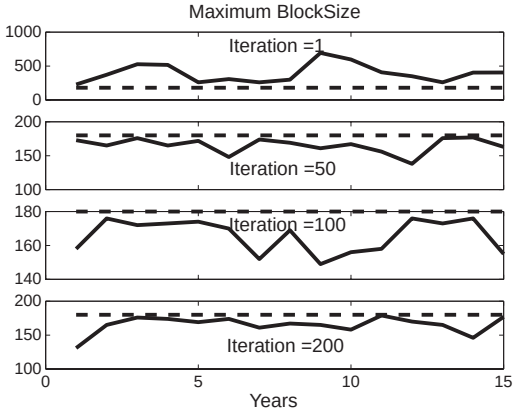

Fig. 3. Maximum blocksize by year at designated iterations of the Metropolis algorithm (dash line = 180).

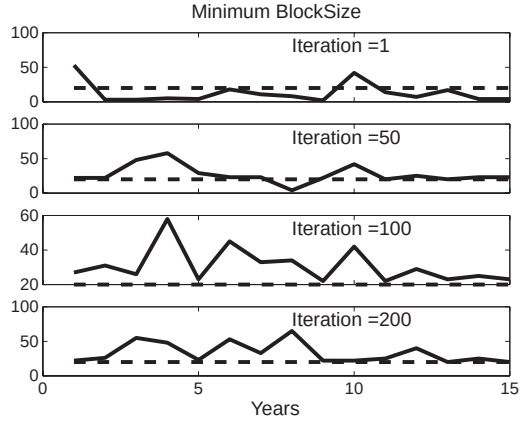

Fig. 4. Minimum blocksize by year at designated iterations of the Metropolis algorithm (dash line = 20).

was to be within 5 to 15 percent of this starting target. The flow model for clearcut area set a flat trend as the target, and allowed clearcut area to deviate by 15 to 25 percent from the target and from one year to the next. The target models are smoothing functions that depend on the data, but predetermined targets could be used as well.

The suitability index was based on net present value (NPV). The regime that gave the most NPV was ranked the highest and the lowest NPV regime was ranked the lowest. The goal was to attain a percentage of the maximum unconstrained NPV while still maintaining the flow and blocksize goals. The maximum unconstrained NPV was defined as the amount that would be achieved if all stands were assigned to the maximum NPV regime.

The results for iterations 1, 50, 100 and 200 (Figs 1–4) show how the algorithm proceeded. Flow of wood (Fig. 1) converged fairly rapidly and showed little change after 100 iterations. The actual flow (solid line) deviated within the required limits relative to the target value (dashed line). Clearcut area (Fig. 2) remained relatively constant over the planning period. Maximum blocksize (Fig. 3) for any particular year was

often above 500 acres for the first few iterations. After 50 iterations, the weight on maximum blocksize was sufficient to keep blocksizes under the 180 maximum size limit (dash line). Minimum blocksize (Fig. 4) is also below 20 (dash line) for the first few iterations. However, by iteration 100 the minimum and maximum blocksizes were within the specified limits. The computer program ran at the rate of about 100 Metropolis iterations per minute on a 270 MHZ computer with all components in the objective function.

The suitability component plays a key role in this example. With no other constraints, the suitability component would lead to assigning the maximum NPV regime to each stand. Without the suitability component included, there would be no requirement for the algorithm to attain any economic goal. The only requirements would be for flow to be relatively smooth and blocksizes to be between 20 and 180. With all components in the objective function, 78 percent of the unconstrained NPV can be attained easily. At about 79 percent of NPV, the upper blocksize limit can no longer be held. The problem was also run with all components except the blocksize component to investigate its cost. Without blocksize restrictions, 92 percent of the unconstrained NPV can be attained, where unconstrained NPV is $560 000 for this problem. The clearcut area begins to trend upward toward the end of the planning horizon at 94 percent of unconstrained NPV, which indicates an unsustainable harvest level, i.e. an infeasible schedule.

10 Discussion and Conclusions

An algorithm has been presented that can produce feasible land management schedules while simultaneously handling adjacency constraints. The algorithm proceeds iteratively until the user specified goals are obtained for each component in the objective function. The method can generate many feasible schedules that have the characteristics specified by the user by repeated application of the Metropolis algorithm. However, it is also possible to converge on a locally optimal solution by slowly increasing the weight on a particular objective function component. The algorithm can be run several times, starting from different random initial solutions, to give some assurance that the globally best solution has been attained.

This approach differs from linear programming (LP), which has historically been the prevalent method for deriving non-spatial harvest schedules. Published applications of LP to spatial harvest scheduling (e.g. Weintraub et al. 1994, Carroll et al. 1995) first produce a non-spatial solution that is massaged by a heuristic algorithm in a second phase to meet spatial constraints. However, the second phase result will not have the optimal properties of the first phase LP solution.

The algorithm proposed here incorporates what are usually considered to be constraints directly into the objective function. For example, even flow is controlled by an objective function component with other components controlling financial objectives and blocksize. This leads to a uniform method for handling objective function weights for any type of component. It also makes it possible to add components as the algorithm proceeds without disrupting progress. A component is added with a relatively small weight so that it will have little influence. The weight is slowly increased until the component's associated goals are obtained. This may also reveal that other components can no longer attain their goals and that the new component causes the problem to become infeasible. Likewise, a component can be removed by slowly decreasing its weight without disrupting the algorithms progress.

The proposed algorithm functions best when a new objective function component is added only after the existing component weights have converged. This avoids the situation where weights on 2 or more components change in tandem and are in fact competing for the same resources. Further research is needed to develop a method whereby multiple components can be simultaneously entered into the objective function without such weight adjustment conflicts being an issue.

Blocksize computation is required repeatedly by this algorithm, which must be done efficiently to minimize overall processing time. The more general spatial capabilities of this method have not been completely demonstrated or tested. However, a procedure has been outlined where-

by additional spatial components can be designed to meet other spatial objectives. This spatial manipulation capability could be beneficial for creating desirable habitat configurations on the landscape. Additional flow terms might be added as well. For example, a component that gets it's data from values that are computed as the simulation proceeds could compute a habitat suitability index (HSI) to keep cumulative HSI relatively constant over time. The algorithm presented here provides a flexible approach that can incorporate economic and environmental goals.

References

Besag, J. 1986. On the statistical analysis of dirty pictures. J. R. Statist. Soc. B 48(3): 259–302.

Bettinger, P., Sessions, J. & Boston, K. 1997. Using tabu search to schedule timber harvests subject to spatial wildlife goals for big game. Ecological Modeling 94: 111–123.

— , Sessions, J.& Johnson, K.N. 1998. Ensuring the compatibility of aquatic and commodity production goals in eastern Oregon with a tabu search procedure. Forest Science 44(1): 96–112.

Carroll, B., Landrum, V. & Lisa Pious, L. 1995. Timber harvest scheduling with adjacency constraints: Using ARC/INFO to make FORPLAN realistic. IN: 1995 ESRI User Conference Proceedings.

Carter, D., Vogiatzis, M., Moss, C. & Arvanitas, L. 1997. Ecosystem management or infeasible guidelines? Implications of adjacency restrictions for wildlife habitat and timber production. Canadian Journal of Forest Research 27: 1302–1310.

Geman, S. & Geman, D. 1984. Stochastic relaxation, Gibbs distribution, and the Bayesian restoration of images. IEEE Trans. Patt. Anal. Mach. Intell. PAMI-6: 721–741.

Glover, F. & Laguna, M. 1993. Tabu search. In: Modern heuristic techniques for combinatorial problems. Ed. Collin Reeves. Halsted Press. New York.

Hof, G.H. & Raphael, M.G. 1993. Some mathematical programming approaches for optimizing timber age-class distributions to meet multispecies wildlife population objectives. Canadian Journal of Forest Research 23: 828–834.

— & Joyce, L.A. 1993. A mixed integer linear programming approach for spatially optimizing wildlife and timber in managed forest ecosystems. Forest Science 39(4): 816–834.

— , Bevers, M., Joyce, L. & Kent, B. 1994. An integer programming approach for spatially and temporally optimizing wildlife populations. Forest Science 40(1): 177–191.

Jamnick, M.S. & Walters, K.R. 1993. Spatial and temporal allocation of stratum-based harvest schedules. Canadian Journal of Forest Research 23: 402–413

Jones, J.G., Meneghin, B.J. & Kirby, M.W. 1991. Formulating adjacency constraints in linear optimization models for scheduling projects in tactical planning. Forest Science 37: 1283–1297.

Lockwood, C. & Moore, T. 1993. Harvest scheduling with spatial constraints: a simulated annealing approach. Canadian Journal of Forest Research 23: 468–478.

Mayer, D.G., Belward, J.A. & Burrage, K. 1998. Tabu search not an optimal choice for models of agricultural systems. Agricultural Systems 58(2): 243–151.

Meneghin, B.J., Kirby, M.W. & Jones, J.G. 1988. An algorithm for writing adjacency constraints efficiently in linear programming models. In: Kent, B.M. & Davis, L. (tech. coords.) Proc., the 1988 symp on systems analysis in forest resources. USDA For. Ser. Gen. Tech Rep. RM-161. p. 46–53.

Metropolis, N., Rosenbluth, A., Rosenbluth, M., Teller, A. & Teller, E. 1953. Equation of state calculations by fast computing machines. J. Chem. Physics 21: 1087–1091.

Murray, A.T. & Church, R.L. 1995. Heuristic solution approaches to operational forest planning problems. OR Spektrum 17: 193–203.

Osman, I. H. 1993. Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem. Annals of Ops. Res. 41.

Snyder, S. & ReVelle, C. 1996. Temporal and spatial harvesting of irregular systems of parcels. Canadian Journal of Forest Research 26: 1079–1088.

Tarp, P. & Helles, F. 1997. Spatial optimization by simulated annealing and linear programming. Scandinavian Journal of Forest Research 12: 390–402.

Torres-Rojo, J.M. & Brodie, J.D. 1990. Adjacency constraints in harvest scheduling: an aggregation heuristic. Canadian Journal of Forest Research 20: 978–986.

Van Deusen, P. 1996. Habitat and harvest scheduling using bayesian statistical concepts. Canadian Journal of Forest Research 26: 1375–1383.

Weintraub, A., Barahona, F. & Epstein, R. 1994. A column generation algorithm for solving general forest planning problems with adjacency constraints. Forest Science 40(1): 142–161.

York, J. 1992. Use of the Gibb's sampler in expert systems. Artificial Intelligence 56: 115–130.

Yoshimoto, A. & Brodie, J. D. 1994. Comparative analysis of algorithms to generate adjacency constraints. Canadian Journal of Forest Research 24: 1277–1288.

Total of 25 references