



Remsoft Inc.
332 Brunswick Street,
Fredericton, NB,
Canada E3B 1H1
1-800-792-9468 or
1-506-450-1511

www.remsoft.com

Proceedings

International Symposium on Systems Analysis and Management Decisions in Forestry

Forest Management and Planning in a Competitive and Environmentally Conscious World

March 9 - 12,
1993

Valdivia, Chile

Design and development of a generalized forest management modeling system: Woodstock

Karl R. Walters

Remsoft Inc.
332 Brunswick Street, Fredericton
NB, Canada E3B 1H1

Abstract

This paper describes Woodstock, a forest management modeling system that accommodates binary search, Monte Carlo simulation and linear programming based models within a single input file format. By employing a language interpreter approach to modeling, Woodstock enables the construction of forest models that can be analyzed using very different solution techniques with only minor changes in syntax.

Keywords: Harvest scheduling, simulation, optimization, object-oriented programming.

Introduction

Harvest scheduling models typically embrace a single analytical technique: simulation, binary search, linear programming (LP), dynamic programming (DP), *et cetera*. Each of these approaches has distinct advantages and disadvantages, depending on the problem to be addressed. Because of the complexity of forest management planning problems, an analyst may wish to apply several of these modeling techniques to address specific concerns, such as the effect of stochastic timber yield variables on LP results (Hof et al. 1988; Leefers 1991) or the risk of catastrophic events when developing a harvest schedule (Moll 1991). However, in order to do so, one usually must resort to different models or modeling systems, which usually are not equivalent in capabilities or features.

Some differences between models are inherent to the analytical technique being used – binary search models cannot handle the many constraints that are easily accommodated within a linear programming formulation, for example. Recognizing that there are a number of consistent features across forest planning models, regardless of how solutions are derived, *Woodstock* was developed. *Woodstock* is a modeling system which permits analysis of forest management problems using binary search, Monte Carlo simulation and linear programming techniques.

Conceptual overview

The fundamental design consideration in developing *Woodstock* was model independence – the provision of powerful modeling capabilities without imposing a particular structure or conceptual point of view.

Other than basic like aging stands and calculating inventories, *Woodstock* provides no built-in actions or outputs whatsoever. Instead, the analyst must define the actions and outputs to be modeled, using the *Woodstock* syntax. Therefore *Woodstock* is rather like a programming interpreter such as Qbasic in that it cannot do very much on its own, but it can be programmed to perform a variety of useful things.

Developing models with *Woodstock* requires understanding of six key concepts: landscape themes, development types, actions, transitions, yield components and outputs. Each of these concepts represents a particular aspect of any *Woodstock* model and input files must have sections devoted to each of them.

Landscape themes are a method of classifying a forest using either stratum-based or area-based themes. Up to sixteen landscape themes can be specified where each theme represents a set of related attributes that describe the forest. Within a theme, any number of attributes may be specified. For example, typical themes might include working group (species), site quality, management emphasis, management intensity, management unit, habitat zone, watershed, topography, et cetera. Theme attributes can be grouped together using attribute aggregates (exclusive sets) or by using wildcards (inclusive sets). Landscape themes are functionally similar to the level identifiers used in FORPLAN Version 2 (Johnson et al. 1986) or thematic layers in a geographic information system coverage.

Development types are unique combinations of landscape themes which represent the basic forest management units in a forest. A development type represents all stands or stand types within the forest that are expected to respond to management in the same way. Development types are represented in a *Woodstock* model as age class distributions identified by a particular sequence of landscape theme identifiers called a mask. This differs from most forest planning models where the forest is divided into analysis areas which represent only single age classes.

Actions are management interventions performed on the forest but they may also include natural events such as forest fire. Actions are defined in terms of (1) whether regeneration occurs following the action, (2) which development types are eligible for an action (or susceptible to one) and when, (3) how yields associated with the action should be calculated and (4) whether all area in an eligible development type should in fact be considered available. The

Woodstock syntax is sufficiently flexible that virtually any silvicultural regime can be modeled successfully.

Actions are always associated with transitions, a description of the changed developmental pattern(s) of the affected forest. Transitions are defined in terms of the pre-treatment development type (source) and the post-treatment development type(s) (targets). All development types for which an action can be applied must have a corresponding transition statement. Transitions arising from an action can be specified on an areal, proportional or Monte Carlo basis. Transitions may be single outcome or multiple outcome with up to 10 target development types.

Yield components are indicators of the state of the forest, usually representing desired benefits from forest management. They may be anything that can be represented on an areal basis: timber yields, soil sedimentation, treatment costs, et cetera. Presently, there are two kinds of yield specifications in a *Woodstock* model: age dependent or time dependent. An age dependent yield is simply one that is a function of stand age (e.g., stand volume in m³/ha) whereas a time dependent yield varies with planning periods (e.g., kilometres of access road).

Outputs are quantities that are to be reported or controlled within the model. They are triggered by actions and are a function of the treatment area and one or more yield components associated with the development type treated. Typical outputs in a *Woodstock* model might be harvest volume, inventory of particular components, total cost of particular treatments, et cetera. Outputs may be defined directly in terms of their triggering actions and associated yield components, or they may be defined in terms of previously defined outputs (e.g. total harvest volume may be defined as total softwood harvest volume plus total hardwood harvest volume).

Reports may be directed to the screen, or to DOS files in ASCII text format or as spreadsheet files in WK1 format. Outputs may also be directed to the screen as run-time line graphs. In addition to built-in reports designed to help debug forest planning models, the user may specify reports on individual actions or outputs for any or all planning periods in the planning horizon.

Object oriented program structure

Woodstock was written in Borland Pascal 7.0 using an object-oriented programming approach. Object-

oriented programming is simply a mechanism for creating data types which inherit characteristics of simpler, more general types. In traditional programming, computer programs are separated into data and code structures, where the latter manipulates the former. In object-oriented programming, data and code are packaged together into a single entity or object and the routines to manipulate data in an object are part of the object itself. The advantage of this is that objects can be copied, and the copy inherits not only the data structure of the parent object, but also the parent's functions and procedures. If the new object must perform new functions or perform an inherited function differently, new procedures or functions can simply be added when the new object is defined. Newly defined functions simply replace inherited functions of the same name. Thus, object-orientation makes it relatively easy to add new features to a program without significantly modifying the existing code.

Woodstock is made up of families of program objects that perform particular functions in the program such as calculating outputs, performing actions, aging development types, et cetera. Each object in a family has a common `_ancestor_` from which it inherits basic properties. For example, all yield curve related functions in *Woodstock* are performed by objects based on the `_basic_curve_type` object. This object is made up of variables which keep track of the number of entries in the yield table, a pointer list storing the yield curve data itself and functions that can initialize the object, read yield information from an input file, normalize yield information, copy yield information from other curves and store yield information. Every instance of `_curve_type` objects inherit the properties of `_basic_curve_type`, enabling it to deal with yield curve information but each has additional functions to handle particular types of yield curves or special cases.

Although the object-orientation of the program code does not affect the user directly, it is the cornerstone of *Woodstock's* flexibility. Model input files are essentially free format ASCII files. Because there are no fixed column formats or requirements for sections to be arranged in a particular order, *Woodstock* input routines need smart parsing capabilities. By separating the input routines into objects specific to particular sections, the objects themselves are responsible for finding the appropriate information in the input files and for checking syntax. If an error is encountered, the information is passed to an error

handling object which reports an appropriate error message. Along with a program structure arranged more along functional lines, another added benefit is that when a new feature is added to *Woodstock*, only objects specific to the new feature need to be created or changed – all other program code is unaffected which substantially reduces development time and facilitates code maintenance.

Model formulations

Inventory Projection Model

Like all simulation models, *Woodstock* requires that the user specify the magnitude of all management interventions and order in which to apply them. In a *Woodstock* model, this information is presented in the queue section. The queue lists for each planning period, target limits for outputs and the actions which generate those outputs. *Woodstock* performs each action in the order in which it appears in the queue, up to the specified limit which may be area, volume generated, cost incurred, et cetera. The order in which actions are applied to individual development types is determined by selection rules specified in the action definition.

In simulation mode, *Woodstock* evaluates every run in terms of success or failure. Success is assumed whenever output levels meet target limits specified by the user in every period of the planning horizon (i.e., levels exceed minimum limits, levels do not exceed maximum limits and levels just meet equality limits). If any output level does not meet a target limit, *Woodstock* evaluates the run as a failure and terminates the simulation immediately.

The user may specify one output limit as a binary search criterion which *Woodstock* can vary up or down to converge to an optimal solution. In binary search mode, the binary search criterion is increased when a run succeeds and is decreased when a run fails – each time the simulation begins again with the newly calculated binary search criterion. This process continues until the change in the binary search criterion falls below a user-specified tolerance. Binary search can only be used with deterministic formulations, but it may be used for area control or volume control formulations. Furthermore, binary search may be turned off completely and harvests scheduled according to fixed methods of area control, percentage of inventory or an absolute amount.

Alternatively, the user may wish to investigate the impacts of random variations in the application of

silvicultural treatments or to probabilistic outcomes in yield response or regeneration. In these cases, *Woodstock* uses a Monte Carlo simulation approach to random events. Again, each run is deemed a success if all target limits are met. However, instead of changing a binary search criterion, *Woodstock* generates random outcomes in each planning period, reinitializes the forest for each successive run and keeps track of the number of successful attempts out of the total number of runs specified by the user. At the end of the specified number of runs, *Woodstock* reports the success rate as a percentage.

Optimization

In optimization mode, *Woodstock* acts as a linear programming matrix generator and report writer. Instead of target limits in a queue, the user specifies an objective function and constraints in the Optimize section of the input file. Unlike simulation mode, *Woodstock* does not require selection rules for actions or a specific sequence in which to apply actions in optimization mode. However, in all other respects, optimization models are identical to simulation models in *Woodstock* syntax.

Woodstock uses many of the simulation routines which calculate outputs, age development types and apply actions to generate coefficients for a LP matrix using a generalized Model II formulation (Iverson and Alston 1986). Once the lp matrix is generated, an optimal solution can be found using any solution software capable of reading LINDO or MPS format input files. A utility program reads the solution file generated by the solver and produces a sequence file which lists the actions, development types, area treated and period of treatment corresponding to the basic variables of the optimal solution. This sequence file is then read into *Woodstock*. The sequence file drives the simulator directly with reports and graphic displays working the same way they do in simulation mode.

Example

The following simple model will illustrate *Woodstock*'s capabilities in formulating binary search, Monte Carlo simulation, and optimization forest models. Presented first are the sections common to all three formulations, followed by the specific syntax required for each type of model. Finally, results from each formulation are presented in tabular form.

In the following example input file, *Woodstock* keywords are in upper case and user-specified values are in lower case.

```
CONTROL
*LENGTH 16 {planning horizon=length*5 yrs}
*RUNS 20 {perform 20 Monte Carlo sim's}
*OPTIMIZE OFF {on for lp matrix generation}
*SCHEDULE OFF {on for lp solution reports}
*QUEUE ON {on for simulation models only}
*WARNINGS ON {report non-fatal errors}

LANDSCAPE
*THEME Cover Type
sp Spruce
po Poplar
nsr Not Satisfactorily Regenerated

LIFESPAN {maximum age any dev type may reach}
sp 50
po 30
nsr 40

AREAS {area distribution: dev. type and age}
*A sp {begins 1 period of age}
1 461 53 670 49 345 357 93 13 118 67
140 120 415 314 173 149 389 400 273 235
948 607 280 229 616 196 257 62 163 4
*A po {begins 1 period of age}
1 430 471 18 75 57 248 9 11 48 69
324 11 709 93 155 206 799 499 1 16
262 24 744 72 29 666 24 565 589
*A nsr
1 194 668 279 1 636

YIELDS {yield components for Sp and Po types}
*Y sp {spruce yields, begin in period 8}
pulp 8 63.8 72.5 77.5 81.3 83.8 85.0 85.0
84.0 82.0 79.0 75.0 70.5 65.5
*Y po {poplar yields, begin in period 8}
pulp 8 33.8 37.3 39.8 41.8 43.3 44.3 44.8
45.0 45.0 44.8 44.3 43.8 43.3

ACTIONS
*ACTION clct Y Clear Cut {resets age}
*OPERABLE clct
sp _age 9 Pulp 40 {FH 45,UNTIL PULP < 40}
po _age 9 Pulp 35 {FH 45,UNTIL PULP < 35}
*SELECT clct {selection rule for simulation}
_MAX_AGE {oldest first rule (MAX _age)}
*ACTION plnt Y Plant NSR {resets age}
*OPERABLE plnt
nsr _age 0 _age 50
*SELECT plnt
_MAX_AGE {plant oldest cutovers first}

TRANSITIONS
*CASE clct
*SOURCE sp {for spruce dev. types:}
*TARGET sp 70 {70% regenerate Sp}
*TARGET po 20 {20% regenerate Po}
*TARGET nsr 10 {10% do not regenerate}
*SOURCE po {for poplar dev. types:}
*TARGET sp 20 {20% regenerate Sp}
*TARGET po 75 {75% regenerate Po}
*TARGET nsr 5 {5% do not regenerate}
*CASE plnt
*SOURCE nsr {for planted nsr dev. type:}
*TARGET sp 100 {100% regenerate sp}

OUTPUT
*OUTPUT tot_pulp Total Pulp {pulp from}
*SOURCE ? clct pulp {both Sp & Po types}
*OUTPUT nsr_area NSR Area {NSR area from}
*SOURCE nsr _Invent _Area {NSR type only}
*OUTPUT plnt_area Plantation {plnt area from}
*SOURCE nsr plnt _Area
```

Binary Search Model

To create a binary search formulation, the model requires a Queue section to specify the required levels of outputs and the order in which to apply actions. To invoke the binary search algorithm, the MAX keyword is included in the first limit statement (i.e., tot_pulp is the binary search criterion).

```
QUEUE
*TARGET tot_pulp = 50000 _MAX 1.._LENGTH
 *SOURCE ? clct 100 {from all clearcuts}
*TARGET plnt_area < 250 {no more than 250 ha}
 *SOURCE nsr plnt 100
```

After running this model through *Woodstock*, the maximum even-flow harvest level was found to be 66 940.5 m³/period. The tolerance level for the binary search was 1 m³ and 19 iterations were required to converge to the solution value.

Monte Carlo Simulation Model

To formulate the example problem as a Monte Carlo simulation, the Transitions section must be changed to simulate random regeneration outcomes:

```
*CASE clct
 *SOURCE sp {for spruce dev. types:}
 *TARGET sp 70 {70% probability-> Sp}
 *TARGET po 20 {20% probability-> Po}
 *TARGET nsr 10 {10% probability-> NSR}
*SOURCE po {for poplar dev. types:}
 *TARGET sp 20 {20% probability-> Sp}
 *TARGET po 75 {75% probability-> Po}
 *TARGET nsr 5 {5% probability-> NSR}
```

Instead of the fixed proportions used in the binary search model, the Monte Carlo simulation model uses probabilities of regeneration for each development type. The *LIMIT R 100 statement can be interpreted as _for each 100 ha random allocation unit in the development type, assign regeneration outcomes randomly according to the following expected values_. By increasing or decreasing the size of random allocation units, the user can increase or decrease the magnitude of the variation in outcomes across simulation runs.

Using the harvest level found in the binary search model, the Monte Carlo simulation model successfully sustained that harvest level 11 times out of 20. When the harvest was reduced by 2.5%, the harvest level was sustainable in all 20 runs.

Linear Programming Model

To formulate a Model II linear program, an Optimize section must be added to the input file. This section specifies the objective function, any constraints to be applied in the model and the format the matrix should be written in (LINDO or MPS). To

be approximately equivalent to the other formulations, this example model maximizes first period harvest subject to non-declining yield and planting no more than 250 ha per period.

```
OPTIMIZE
*OBJECTIVE
 _MAX Tot_Pulp 1 {maximize 1st period}
*CONSTRAINTS
 _NDY(Tot_Pulp) 1.._LENGTH {nondeclining yld}
 plnt_area = 250 1.._LENGTH
*FORMAT MPS {use MPS format}
```

The optimal solution produced an objective function value of 67 470 m³/period which was marginally higher than that found using binary search. With more complex models, the difference would likely be more substantial (Jammick 1990).

Potential Uses Of Woodstock

The example presented in this paper is an exceedingly simple model, but it demonstrates effectively how *Woodstock* can handle very different model formulations with ease: with only a few changes to the input file, three distinct kinds of analysis could be performed in minutes. However, even with much larger and more complex *Woodstock* models, it is trivial to convert one model type into another.

One of the greatest strengths of *Woodstock* is its potential as a teaching tool. New users can quickly develop simple models which can be used to learn about forest dynamics (e.g., demonstrate linkages between harvest levels and forest structure, tradeoffs among harvest and silviculture options, relative efficacy of alternative selection rules, et cetera). As users become more proficient at using *Woodstock*, they compare different model formulations to determine how best to formulate a model to answer relevant management concerns.

As a harvest scheduling model, *Woodstock* models can address virtually any kind of silvicultural or harvesting regime. Complicated silvicultural prescriptions can be modeled as individual actions performed sequentially or as a single intervention _ which method to use is up to the model builder. Because all actions and outputs are user-defined, *Woodstock* can be used to model forest management problems which are not timber oriented, such as fire or vegetation management plans for parks and conservation areas.

The Monte Carlo simulation features are useful for evaluating stochastic variations in timber yields, for evaluating impacts of landowner behavior for NIPF

wood supply analyses or impacts on forest structure and harvest levels in presence of fire or pest risks. *Woodstock* is currently being used by forest products marketing boards to estimate wood supply from non-industrial private forest land in their regulation areas.

Future developments

One of the difficulties in developing a system like *Woodstock* is developing a structure which is straightforward and concise, yet retains sufficient flexibility to address many different problems. For example, in many cases the functionality of a lp model cannot be replicated in a simulation model and so, wherever possible, features specific to a type of model are isolated in particular sections (i.e., the Queue section for simulation models and the Optimize section for LP models). However, it is not always possible to do this, with the result that some sections require alternative syntax constructions that apply to only particular model types.

The syntax developed so far has been remarkably robust, requiring few changes despite the extensions that have been incorporated into *Woodstock*. However, it is anticipated that adjustments will need to be made to the syntax to improve the usability of the system by maintaining consistency of meaning for keywords and constructs used in multiple sections of an input file.

Presently, all efforts are being directed toward finalizing the LP extensions to *Woodstock*. In the future, the intent is to strengthen *Woodstock* capabilities in the areas of economic analyses and land allocation problems. Specifically, these enhancements would include the ability to formulate models similar to ECHO (Walker 1971), to conduct price responsive timber supply analyses, and to formulate models employing aggregate emphasis or coordinated allocation choices.

Acknowledgement

The development of the linear programming extensions to *Woodstock* was partially funded by Forestry Canada and the British Columbia Ministry of Forests as part of the Canada-British Columbia Partnership Agreement on Forest Resource Development: FRDA II - a 4 year (1991-1995) \$200 million program cost-shared equally by the federal and provincial governments.

Literature cited

- Hof, J.G., Robinson, K.S. and Betters, D.R. 1988. Optimization with expected values of random yield coefficients in renewable resource linear programs. *For. Sci.* 34: 634-646.
- Iverson, D.C. and Alston, R.M. 1986. The Genesis of FORPLAN: A Historical and Analytical Review of Forest Service Planning Models. Gen. Tech. Rep. INT-214, USDA Forest Service, Intermountain Research Station, Ogden, UT. 31pp.
- Jamnick, M.S. 1990. A comparison of FORMAN and linear programming approaches to timber harvest scheduling. *Can. J. For. Res.* 20: 1351-1360.
- Johnson, K.N., Stuart, T.W. and Crim, S.A. 1986. FORPLAN Version 2: An Overview. USDA Forest Service, Land Management Planning Systems Section, Washington, DC. 98+xii pp.
- Leefers, L.A. 1991. Incorporating linear programming and Monte Carlo simulation in a spreadsheet-based harvest scheduling model. IN Buford, M.A., comp. 1991. Proceedings of the 1991 Symposium on Systems Analysis in Forest Resources, March 3-6, 1991. Charleston, SC. Gen. Tech. Rep. SE-74, USDA Forest Service, Southeastern Forest Experiment Station, Asheville, NC. 423pp.
- Moll, R.H.H. 1991. Modelling regeneration and pest control alternatives for a forest system in the presence of fire risk. IN Buford, M.A., comp. 1991. Proceedings of the 1991 Symposium on Systems Analysis in Forest Resources, March 3-6, 1991. Charleston, SC. Gen. Tech. Rep. SE-74, USDA Forest Service, Southeastern Forest Experiment Station, Asheville, NC. 423pp.
- Walker, J.L. 1971. An economic model for optimizing the rate of timber harvesting. Ph.D. dissertation, University of Washington, Seattle, WA. 117pp.