A Stochastic Heuristic Object-Oriented Approach to Satisficing Multiple-Objective Forest Management Problems *

Stephen M. Dewhurst and Oscar García
University of Northern British Columbia †

Abstract

Lurch, a forest management decision support tool utilizing an object-oriented architecture, has been developed and tested in the context of multiple-objective and multiple-stakeholder forest management planning problems in British Columbia. A satisficing approach based upon Criteria and Indicators, goal programming, and stochastic heuristic sub-optimization is used. The system is controlled interactively through an intuitive graphical user interface.

The object-oriented architecture provides fast response, for a very high rate of combinatorial simulation and evaluation cycles. The result is an accessible interactive environment for exploring potential management solutions to complex forest management planning problems. Lurch has been applied to real-world issues in British Columbia.

1 Introduction

Planning for, and implementing, the integrated resource management of public forest lands for multiple values is a daunting task. In British Columbia (BC, Canada), the public forests are still managed primarily for

*Presented at the Symposium on Systems and Models in Forestry, Punta de Tralca, Chile, March 4–7, 2002.

†3333 University Way, Prince George, B.C., Canada V2N 4Z9. dewhurst@unbc.ca, garcia@unbc.ca

timber production, but as public land they must also be managed for a variety of other values as well. To a large extent, the continuation of the public mandate for industrial timber production in Crown forests is dependent upon assuring the people of BC, and the domestic and international customers for BC forest products, that the forests are being managed in a sustainable and responsible manner. This involves explicit recognition of forest values such as biodiversity, wildlife and visual quality, and the opportunity for public participation.

Optimization approaches that have been successful in other situations (see, e.g., Dykstra 1984, García 1990, and references therein), are not that useful for forest management planning and analysis in public forests that provide a wide variety of "resources" to an even wider range of stakeholder groups. The artificial translation of complex management situations into single factor objective function formulations, using such measures as net present value, becomes inadequate in integrated resource management planning situations involving significant non-market resource values. A *satisficing* approach (Simon 1957) may be more appropriate. When dealing with politically contentious issues in public planning contexts, requiring that a broker (the analyst) take the objectives of the stakeholder, translate those objectives to a complex mathematical programming formulation, conduct the analysis, and interpret the abstract analytical results in terms of the original desires and objectives of the stakeholders can lead to accusations of bias, hidden agendas, and insincerity. What is needed are tools that allow all participants to examine tradeoffs in a transparent and easily comprehensible way.

In response to these challenges, an Operational Research tool, Lurch, has been developed in the context of the management of Crown forest lands in British Columbia, attempting to overcome some of the difficulties. Lurch implements some aspects of the Criteria and Indicators framework, and caters to the demand from an increasingly sophisticated public for a much more direct role in the analysis and management planning of our public forests. A combination of goal programming and heuristic optimization within a satisficing framework is used, together with an interactive graphical user interface. The implementation, based on novel object-oriented techniques, reduces software complexity and enhances algorithmic efficiency and responsiveness.

Lurch has been placed in the public domain subject to the Gnu Public License (GPL), and can be obtained from `http://researchforest.unbc.`

ca. It runs on any computer platform supporting the SUN Java Virtual Machine, version 1.3 or higher.

2 The forest model

Users interact with Lurch through the graphical user interface shown in Figure 1. Spatial distribution of indicators over the forest in any selected time period are visualized in the right-hand-side panel. The user-supplied indicators are listed and selected for display above the map, and the time periods below it. Commonly, the planning horizon is divided into 4 to 15 periods of 10–20 years each.

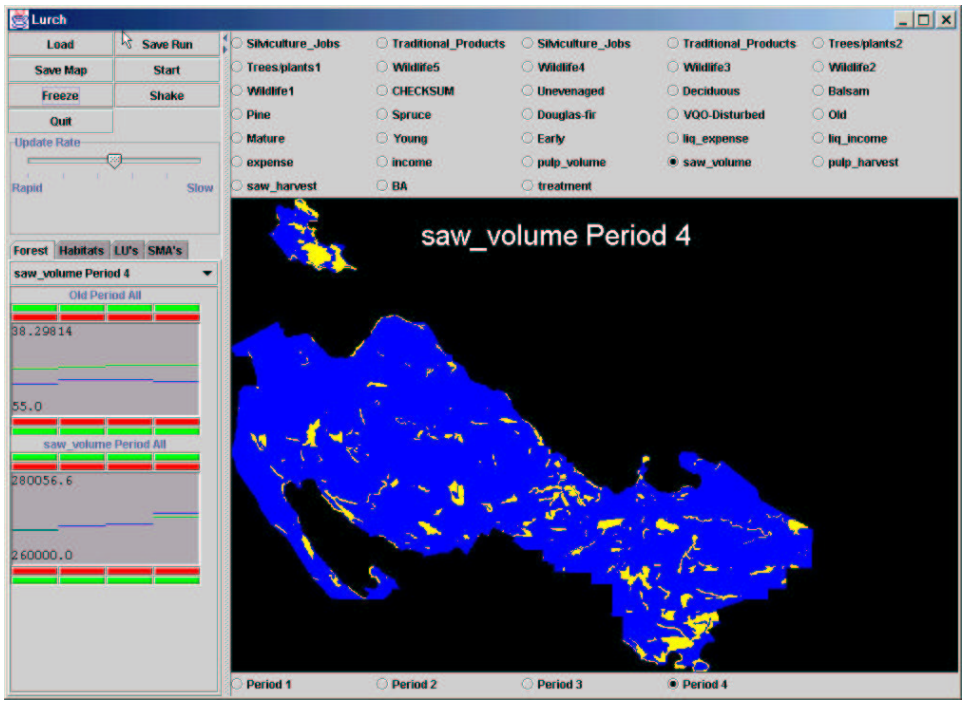

Figure 1: The Lurch graphical user interface

Indicator targets and their preference weights are specified graphically on the left. Targets apply to *Tally Units*, which may be the whole forest, or members of any one of three kinds of land groupings indicated in the tabs. In addition, a target may apply to the full planning horizon, or individually

to each planning period. The objective function to be minimized is therefore of the form

$$\sum_{u,t,i} (w^+_{uti} d^+_{uti} + w^-_{uti} d^-_{uti}) \ , \tag{1}$$

where u, t and i indicate tally unit, period and target, respectively, d^+ and d^- are the deviations above and below target, and w^+ and w^- are preference weights. By selecting a tab, any targets associated with the grouping can be chosen for display and/or modification. Both the specified targets and currently realized results are shown. Targets for each period are set by clicking on the graph, and preference weights are modified through the rows of buttons shown above and below the targets graph. The goal-seeking algorithm runs continuously, adjusting to altered targets and weights in real-time. A slider adjusts the display updating frequency, relative to the algorithm cycling rate which is of the order of 1000 changes per second. Several other functions are controlled through the cluster of buttons on the upper-left.

The targets correspond to the indicators, applied on a per hectare basis. These indicators can be real numbers, e. g., volume per hectare, or qualitative binary variables, e. g., presence of old growth.

The three land groupings that form tally units are called *Landscape Units*, *Habitat Types*, and *Special Management areas*. These are overlayed static land classifications associated with different sets of indicators (Figure 2). Although their precise interpretation is flexible, Landscape Units are typically administrative subdivisions, such as compartments, while Habitat Types represent bio-ecological classifications or site types. Unlike these, the optional Special Management Areas are not a partition of the forest, but represent discrete areas where certain specific targets are relevant, for instance, riparian zones or caribou breeding areas.

Besides these classifications with fixed attributes, within each tally unit the land is subdivided according to cover type (species, etc.) and age class. The contiguous areas with a same cover type, age class, and tally unit, are *stands*, the atomic management units (Figure 2, also called *blocks* in the program documentation). There are typically several thousand stands. The cover type and age class of a stand can change with management prescriptions and the passage of time, but the stands are indivisible. Age classes are the same size as time periods, and uneven-aged stands are handled as being in a steady-state, with every age class having the same properties.

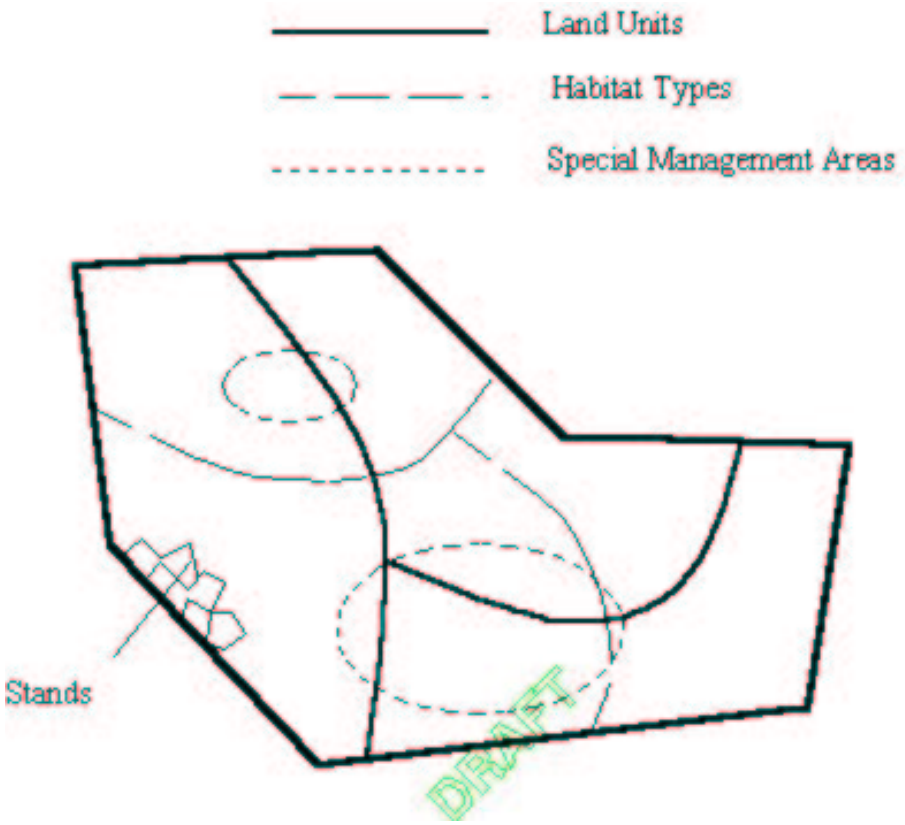

Figure 2: Land classification. Landscape Units, Habitat Types and Special Management areas are generically called *Tally Units*.

Management options cause a stand to change its state, that is, change cover type and/or age class. For instance, a clearcut changes the age to zero, with regeneration into the same or a different cover type. Only single-entry regimes were initially implemented, that is, there was only one treatment within the planning horizon; generalization to multiple entries has just been completed. Therefore, a management option is characterized by a *timing*, when the treatment(s) occur(s), and a *prescription*, the change(s) in cover type. The set of feasible options is specified based on the habitat type and cover type. The result of a management option is a particular sequence of indicator values for the stand.

3 Algorithms

Lurch uses an object-oriented paradigm, which obviates the need for explicitly maintaining and handling complicated data structures. For our purposes, we can view an *object* as an encapsulation of data, which defines the object current state, and procedures which can be applied to it. The procedures either change or interrogate the state. An object procedure is activated by passing a *message* (Gosling et al. 1996).

Written in Java, Lurch implements almost everything as objects. However, conceptually we can limit a description to three simple types of objects: a *Control* agent, *Stand* objects, and *Tally Units*. Oversimplifying a little, the algorithm that seeks a minimum for (1) works as follows (Figure 3).

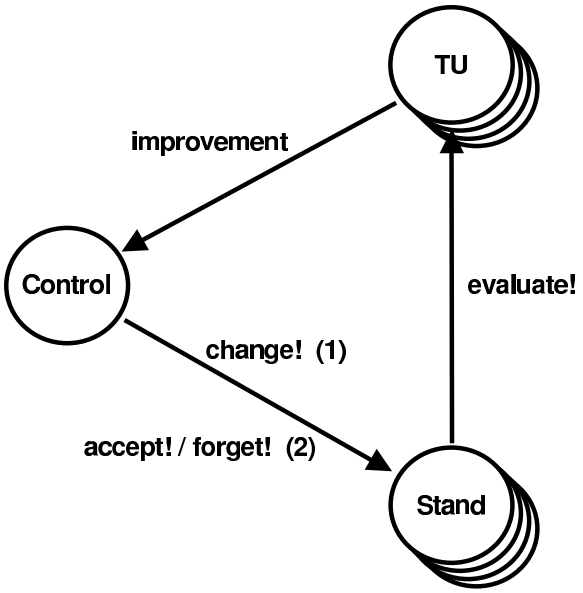

Figure 3: Goal-seeking algorithm loop. (1) and (2) indicate messages on first and second time around.

Initialization:

1. Each stand is assigned at random a feasible management option (timing and prescription).
2. The stand calculates the sequence of indicator values, and passes it to its tally unit.

3. The tally units aggregate the indicator streams, and passes the result to the control agent, which computes the total objective function value.

Loop:

1. Control picks a stand at random, and tells it to try a new management option.
2. The stand chooses to change both timing and prescription with probability 0.1, or timing only with probability 0.9. Calculates the sequence of indicator values, and passes it to its tally unit.
3. The tally unit calculates the resulting change in its contribution to the objective function, and communicates it to Control.
4. Control checks if there is an improvement in the objective. If there is, it tells the stand to make effective the change. Otherwise, the change is forgotten.

The process runs continuously, tracking target and weight changes made by the user. The *Shake* button causes the algorithm to randomly reinitialize the timing and prescription of each stand, as a guard against possible local optima. In practice, it has been found that this simple greedy algorithm approach is quite effective, and differences between repeated solutions are generally negligible. The initial intention of implementing an annealing strategy (Lockwood and Moore 1993) has been found largely unnecessary, although experimentation with these techniques continues.

4 Discussion

The object-oriented message-passing mechanism avoids complex explicit data structures, and automatically exploits sparsity in a very efficient manner. Only the minimal set of changes in indicators arising from the stand state change in each cycle is calculated, furthering efficiency.

As an example, in a problem with 6800 stands, 10 planning periods, 3 landscape units, 3 habitat types, 15 special management areas, and 20 indicators, approximately 1000 stand changes per second were evaluated on a 1 GHz Pentium III computer. Given 30 goals, from a random start this typically

leads to convergence at a near-optimal solution in less than 30 seconds. The response to incremental changes in targets and weights is very fast.

Some extensions and refinements are under development. Experimentation with various annealing strategies, and the handling of multiple-entry stand management regimes, has already been mentioned. Simultaneously changing more than one stand in each cycle is also under investigation. A recent addition uses a planar adjacency graph data structure to implement adjacency constraints.

The most extensive real-life application of Lurch to date has been to participatory, strategic-level planning on the John Prince Research Forest, which is jointly managed by the Tl'azt'en Nation and the University of Northern British Columbia (Karjala 2001). Members of the native community, including elders with little formal education and no background or training in systems analysis or forestry, were able to be fully involved in the planning and decision-making. They found Lurch to be useful and accessible as a tool to help them understand the concepts of management for multiple objectives, analysis of tradeoffs between competing goals, and long-term effects of alternative management policies.

References

Dykstra, D. P., 1984. Mathematical Programming for Natural Resource Management. McGraw-Hill.

García, O., 1990. Linear Programming and related approaches in forest planning. New Zealand Journal of Forestry Science 20, 307–331.

Gosling, J., Joy, B., Steele, G., 1996. The Java Language Specification. SunSoft Java Series. Addison Wesley Developers Press,, Menlo Park, CA.

Karjala, M. K., December 2001. Integrating aboriginal values into strategic-level forestry planning on the John Prince Research Forest, Central Interior, British Columbia. Master's thesis, University of Northern British Columbia.

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

Simon, H. A., 1957. Models of Man. Wiley.