Eight Heuristic Planning Techniques Applied to Three Increasingly Difficult Wildlife Planning Problems

Pete Bettinger, David Graetz, Kevin Boston, John Sessions and Woodam Chung

Bettinger, P., Graetz, D., Boston, K., Sessions, J. & Chung, W. 2002. Eight heuristic planning techniques applied to three increasingly difficult wildlife planning problems. Silva Fennica 36(2): 561–584.

As both spatial and temporal characteristics of desired future conditions are becoming important measures of forest plan success, forest plans and forest planning goals are becoming complex. Heuristic techniques are becoming popular for developing alternative forest plans that include spatial constraints. Eight types of heuristic planning techniques were applied to three increasingly difficult forest planning problems where the objective function sought to maximize the amount of land in certain types of wildlife habitat. The goal of this research was to understand the relative challenges and opportunities each technique presents when more complex difficult goals are desired. The eight heuristic techniques were random search, simulated annealing, great deluge, threshold accepting, tabu search with 1-opt moves, tabu search with 1-opt and 2-opt moves, genetic algorithm, and a hybrid tabu search / genetic algorithm search process. While our results should not be viewed as universal truths, we determined that for the problems we examined, there were three classes of techniques: very good (simulated annealing, threshold accepting, great deluge, tabu search with 1-opt and 2-opt moves, and tabu search / genetic algorithm), adequate (tabu search with 1-opt moves, genetic algorithm), and less than adequate (random search). The relative advantages in terms of solution time and complexity of programming code are discussed and should provide planners and researchers a guide to help match the appropriate technique to their planning problem. The hypothetical landscape model used to evaluate the techniques can also be used by others to further compare their techniques to the ones described here.

Keywords spatial harvest scheduling, forest planning, adjacency constraints
Authors´ addresses *Bettinger & Graetz*: Department of Forest Resources, Oregon State University, Corvallis, OR 97331; *Sessions & Chung*: Department of Forest Engineering, Oregon State University, Corvallis, OR 97331; *Boston*: Carter Holt Harvey Forest Fibre Solutions, Tokoroa, New Zealand
E-mails Pete.Bettinger@orst.edu; graetzd@ucs.orst.edu; Kevin.Boston@chh.co.nz; john@sessions.cof.orst.edu; Woodam.Chung@orst.edu
Received 19 September 2000 **Accepted** 14 February 2002

1 Introduction

The spatial arrangement of wildlife habitat and forest management activities is important for a number of reasons, such as to comply with regulatory restrictions and organizational goals and policies, and to maintain or improve aesthetic conditions. Forest regulations, for instance, are placing increasingly restrictive limits on the size and spatial relationships of harvest units (Daust and Nelson 1993). As a result of a need to manage forest land within regulatory frameworks, forest management planning now often attempts to achieve multiple resource goals and often uses spatial constraints on the selection of management activities (O'Hara et al. 1989).

Forest planning models that optimize the spatial arrangement of forest resources to meet a set of management goals vary from the more traditional optimizations techniques, such as linear or mixed integer programming (e.g., Hof et al. 1994), to non-traditional heuristic programming techniques (e.g., Murray and Church 1995). Forest planning problems that incorporate spatial goals, such as clearcut adjacency restrictions, are combinatorial problems by nature. Thus as the problem size increases, the solution space also increases, yet at a disproportionately greater rate (Lockwood and Moore 1993). Mixed integer programming and integer programming techniques have been used to produce management plans with adjacency concerns, but these techniques have substantive limitations (directly related to problem size) when applied to large combinatorial problems (Lockwood and Moore 1993).

The use of heuristic techniques for forest management planning is becoming more prevalent. Many types of complex, non-linear goals (e.g., spatial and temporal distribution of elk habitat, as described in Bettinger et. al. 1997), which have traditionally been considered too complex to solve with traditional optimization techniques, are now being considered. In recent years, heuristic programming techniques have been applied to traditional forest harvest scheduling problems (Hoganson and Rose 1984) as well as to forest transportation problems (Pulkki 1984, Nelson and Brodie 1990, Weintraub et al. 1994, Murray and Church 1995, Weintraub et al. 1995), wildlife conservation and management (Arthaud and Rose 1996, Haight and Travis 1997, Bettinger et al. 1997), aquatic system management (Bettinger et al. 1998), and the achievement of biological diversity goals (Kangas and Pukkala 1996). Comparisons of a few of these techniques have been made in Nelson and Brodie (1990), Murray and Church (1995), Csuti et al. (1997), Pressey et al. (1997), and Boston and Bettinger (1999). The comparisons have generally been made on a limited number of techniques, and were applied to a small range of problem complexities.

This research examines the use of eight heuristic techniques applied to three increasingly difficult wildlife planning problems. The eight techniques include random search, simulated annealing, great deluge, threshold accepting, tabu search with 1-opt moves, tabu search with 1-opt and 2-opt moves, a genetic algorithm, and a hybrid tabu search / genetic algorithm search process. The wildlife planning goals increase in complexity from non-spatial seral-stage goals (acquire the most acres in certain forest age classes), to minimum patch size goals (acquire the most acres in patches of a certain type of forest larger than a minimum size), and then to complimentary, adjacent patch goals (acquire the most acres in patches of a certain type of forest larger than a certain minimum size, that are next to another type of forest that is larger than a certain minimum size). The purpose of this research is to illustrate the opportunities and challenges to using heuristic techniques for forest planning efforts where wildlife habitat goals are one of the main objectives, and to discuss the relative trade-offs among a broad range of techniques in terms of the quality of solutions and the effort required to obtain a solution.

2 Methods

In most planning processes, consideration is given to the decision variables which can be modified and the rules for assigning activities to variables, the rules for selecting new plan configurations, and the length of time the activity selection (i.e., search process) is allowed to proceed (i.e., how long the computer program is run). Quantitative

relationships, or rules, to constrain or guide the assignment of activities can be categorized in many ways; one such categorization is whether the relationships require spatial information. The use of spatial information can make goal achievement a very complex procedure in forest planning applications. Of the three wildlife species habitat goals we describe below, two require spatial information in their computations. The other, a non-spatial goal, consists of simply assembling a number of acres in certain age classes, or strata. Spatial goals can include a wide variety of configurations. We will examine two types of spatial goals: those which require minimum patch sizes, and those which require adjacent habitat types of minimum sizes. We will first describe the three quantitative relationships, or wildlife goals, that we hope to maximize over time, and call them problems A, B, and C. We will then describe the eight heuristic techniques we use to develop forest plans which achieve these goals. Finally, we describe the hypothetical landscape, which is available for others to use as a standard problem set when pursuing further analytical efforts along these lines.

2.1 Wildlife Habitat Goals

Three types of wildlife habitat goals are examined, each utilize increasingly complex criteria in their measurement. The first, problem A, utilizes *non-spatial goals*, where no spatial components are needed to evaluate the goals, and the objective is to achieve the most acres of land over time meeting the conditions we note below. These have often been considered *strata-based* goals. The second, problem B, is called the *minimum patch size goal*, where the objective is to develop the most land area over time in patches of a certain size and condition. The third, problem C, is called the *complementary patch goal*, where the objective is to develop the most land area over time in patches of a certain size and condition that are next to other patches of a certain size and condition. Wildlife biologists associated with a project aimed at developing habitat relationships for vertebrates in the USA Pacific Northwest (Johnson and O'Neil 1999) were asked to provide examples for these three goals. These examples should be viewed as preliminary quantitative relationships, subject to change based on further assimilation of data and further evaluation of research by wildlife professionals.

In all three problems we will assume that the decision variables related to management units are binary (0,1), where an activity assigned to a management unit is assigned to the entire management unit, and not some portion of the management unit. In addition, the activities we consider are two: clearcut harvest or no harvest. A minimum harvest volume is required each time period, and a minimum harvest age is required before a management unit can be harvested. The time horizon is 50 years, we evaluate wildlife habitat goals every 5 years, thus there are 10 5-year time periods. Further, we assumed that the harvesting activities take place in the middle of a time period, and that the evaluation of habitat is a post-harvest evaluation for that time period.

2.1.1 Problem A: Non-spatial Goals

Non-spatial goals do not require spatial information in the computation of their achievement, and generally are based on achieving the *amount* of some resource in a planning area. For example, goals could be developed with the criteria that some amount of habitat, such as old-seral habitat, will be achieved. Four examples of non-spatial goals related to the type of habitat required in the USA Pacific Northwest include:

Sharp-shinned hawk (*Accipiter striatus*) prefers 25–50 year-old even-aged conifer stands.

Cooper's hawk (*A. cooperii*) prefers 30–70 year-old even-aged conifer stands.

Northern goshawk (*A. gentilis*) prefers 150+ year-old conifer stands.

Red tree vole (*Phenacomys longicaudus*) prefers old-growth forests that are ≥195 years old.

Our evaluation of these goals is simply based on the age of the forests in each management unit. The planning problem for maximizing the amount of land in these age classes can be defined as this,

Maximize

$$\left(\sum_{j=1}^{t}\sum_{i=1}^{n}\sum_{k=1}^{m} A_i H_{i,j,k}\right) / \left(\sum_{i=1}^{n} A_i\right) \quad (1)$$

Where:

j = a time period
t = total number of time periods
i = a management unit
n = total number of management units
k = a wildlife species
m = total number of wildlife species
A_i = area of management unit i
$H_{i,j,k}$ = a binary variable indicating whether (1) or not (0) management unit i is considered habitat for species k during time period j

As we will note later, the landscape is composed mainly of conifer stands, and we assume regenerated stands will come back as conifer stands. In the riparian area, we assume the tree composition is mainly hardwood, thus riparian areas, while contributing to the landscape size, will always have a $H_{i,j,k}$ of 0. The potential objective function values amount to (10.0 * k). For example, assume we have 4 wildlife species, and that the entire landscape is considered habitat for these species in every time period. The resulting objective function value is then ((10 * Σ A_i * 4) / (Σ A_i)), or 40. The results we will soon show are less than this theoretical maximum, however, since not all of the landscape can be considered habitat for any (or all) species in every time period. Finally, none of the heuristic techniques described shortly consider infeasible solutions as *current solutions* to the problem (and thus requiring penalty functions to drive them back to feasibility).

Three constraints are also imposed. First, we assume that only one regeneration harvest is allowed per management unit during the planning horizon.

$$\sum_{j=1}^{t} X_{i,j} \le 1 \qquad \forall\, i \quad (2)$$

Where: $X_{i,j}$ = a binary variable indicating whether (1) or not (0) a management unit is harvested in time period j.

There is no direct link between $X_{i,j}$ values and $H_{i,j,k}$ values, since $X_{i,j}$ is a characterization of harvest activity, and $H_{i,j,k}$ is a characterization of whether or not a management unit is considered wildlife habitat for species k. It just so happens that in this example neither values can equal 1 at the same time, but we shall see that this is not necessarily true for wildlife species that consider clearcuts part of their habitat.

Second, the minimum harvest age is 40 years.

$$\text{If } AGE_{i,j} < 40, \quad X_{i,j} = 0 \quad (3)$$

$$\text{If } AGE_{i,j} \ge 40, \quad X_{i,j} \in \{0,1\} \quad (4)$$

Where: $AGE_{i,j}$ = the stand age of management unit i during time period j.

Thus when stands are 40 years old or greater, $X_{i,j}$ will take on a value of either 0 or 1 (from the set {0,1}), yet can only have a value of 1 once over the entire 50-year planning horizon.

Third, the total volume produced from timber harvests must also exceed a minimum volume goal:

$$\sum_{i=1}^{n}\left(A_i X_{i,j} V_{i,j}\right) \ge \text{minimum volume goal } j \qquad \forall\, j \quad (5)$$

Where: $V_{i,j}$ = the timber volume per unit area in management unit i during time period j.

The minimum volume goal for the evaluation of the heuristic techniques was set at 3000 units per time period, which was based on stand age.

2.1.2 Problem B: Minimum Patch Size Goal

Some forest planning goals utilize spatial characteristics of the landscape in the determination of their value to particular wildlife species. One example would be a goal which requires forest patches to be of a minimum size, and composed of a certain type or age of forest, before this area can contribute positively toward the achievement of habitat. For example, the following is a generalization of habitat requirements for three forest birds:

Varied thrush (*Lxoreus naevius*), winter wren (*Troglodytes troglodytes*), and Hammond's flycatcher

(*Empidonax hammondii*) need intact stands of mature or old-growth forests greater than 20 hectares in size.

We will assume in our subsequent analyses that mature or old-growth forests are stands which are greater than 80 years of age, although this is debatable and certainly a different age could be assumed (or possibly some other site-specific factor). Our problem formulation simply seeks to maximize the amount of land in these types of patch conditions.

The problem formulation is similar to that of the non-spatial goal problem formulation, except that the evaluation of habitat is different. Here, a recursive function, using what Murray (1999) describes as an area restriction method, is used to evaluate the size of patches that consist of forests that are ≥ 80 years old. So, management units that are ≥ 80 years of age do not necessarily have an $H_{i,j,k}$ equal 1.0, since they must also comprise, or be a part of, a patch that is at least 20 ha in size.

$$\text{If } A_i B_{i,j} + \sum_{z \in Ni \cup Si} A_z B_{z,j} \geq \text{minimum patch size} \quad (6)$$

$$H_{i,j,k} = 1$$

$$\text{Else } H_{i,j,k} = 0$$

Where:

$B_{i,j}$ = a binary variable indicating whether (1) or not (0) management unit i is greater than or equal to the minimum desired age (80 years) during period j

N_i = the set of units adjacent to management unit i

S_i = the subset of adjacent units to the neighbors of management unit i and all units adjacent to neighbors of neighbors, and so on

z = a single management unit from the set S_i

2.1.3 Problem C: Complementary, Adjacent Patch Goal

This third and final planning goal seeks to achieve a landscape with the most area in complementary, adjacent patches. These goals indicate that one type of habitat (such as a patch of older forest of a certain size, for nesting and roosting) should be placed adjacent to another (such as a patch of young forest of a certain size, for foraging), to be of most benefit to a particular species. For example:

Great gray owl (*Strix nebulosa*) prefers early seral stage forests (clearcuts) for foraging, yet they should be adjacent to mature or old-growth stands.

In this case we will assume that "old forest stands" are those with an average age greater than 80 years, and that "early seral stage forests" are those with an age of 10 years or less. In addition, to count towards the complementary habitat goals we assume that the size of the old forest stand must be 20 ha or greater, and that the size of the adjacent early seral forest is 10 ha or greater.

This problem formulation is also similar to that of the non-spatial goal problem formulation, except that once again the evaluation of habitat is different. Here, two recursive functions, using what Murray (1999) describes as an area restriction method, are used to evaluate the size of patches that consist of forests that are ≥ 80 years old, and forest that are ≤ 10 years old. These require similar formulations to those described in equation 6. Then a process using what Murray (1999) describes as a unit restriction method, is used to determine whether any of the older forest management units that are part of patches ≥ 20 ha are touching early seral units that are part of early seral patches ≥ 10 ha. So, management units that are ≥ 80 years of age, or ≤ 10 years of age do not necessarily have an $H_{i,j,k}$ equal 1.0. given the rules for evaluating the complementary patch goal.

$$\text{If } O_{i,t} = 1 \text{ and } \sum_{z \in Ni} Y_{z,t} > 1 \quad \text{then } H_{i,j,k} = 1 \quad (7)$$

$$\forall\, i,t$$

$$\text{Else if } Y_{i,t} = 1 \text{ and } \sum_{z \in Ni} O_{z,t} > 1 \quad \text{then } H_{i,j,k} = 1$$

$$\forall\, i,t$$

$$\text{Else } H_{i,j,k} = 0$$

Where:

$O_{i,t}$ = a binary variable indicating whether (1) or not (0) management unit i belongs to a set of management units that describe a patch where all of the management unit are ≥ 80 years of age and the total patch size is ≥ 20 ha

$Y_{z,t}$ = a binary variable indicating whether (1) or not

(0) management unit z belongs to a set of management units that describe a patch where all of the management unit are ≤ 10 years of age and the total patch size is ≥ 10 ha
N_i = the set of units adjacent to management unit i
z = a single management unit from the set N_i

To increase the complexity of this problem, the size of the openings created in each time period are limited to 48.56 ha (120 acres), to resemble a forest green-up policy. The green-up constraint also uses an area restriction model technique to determine how large the clearcuts are in any one time period.

$$A_i X_{i,j} + \sum_{z \in Ni \cup Si} A_z X_{z,j} \leq \text{maximum clearcut size} j \quad (8)$$
$$\forall i, j$$

Where:
N_i = the set of units adjacent to management unit i
S_i = the subset of *treated* adjacent units to the neighbors of management unit i and all units adjacent to neighbors of neighbors, and so on
z = a single management unit from the set S_i

Thus clearcuts that, in aggregate, are larger than 48.56 ha, result in an infeasible solution.

2.2 Heuristic Planning Techniques

Eight types of heuristic techniques were used to solve the three wildlife planning problems. The techniques include random search, simulated annealing, the great deluge algorithm, threshold accepting, tabu search with 1-opt moves, tabu search with 1-opt and 2-opt moves, a genetic algorithm, and a hybrid tabu search / genetic algorithm search process. We next briefly describe these algorithms and provide a flow chart for each detailing how they were used to solve the wildlife planning problems. In addition, problem A was solved using an integer programming technique, thus providing an optimal solution to compare against the heuristic techniques. Problems B and C were not solved with an integer programming technique, due to the complexity of the wildlife goals in these problems.

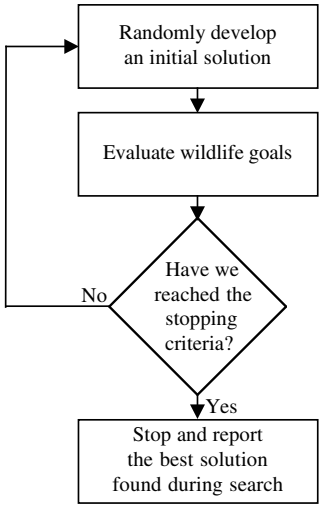

Fig. 1. A flow chart of the random search process.

2.2.1 Random Search

Random search serves as a baseline method of scheduling; to be considered viable, an optimization technique should perform better than random searching for solutions (Valsta 1993). Monte Carlo integer programming techniques have long been studied for use in forest management (e.g., Nelson and Brodie 1990). There are a wide variety of Monte Carlo methods, some variants of which we discuss shortly, but when we illustrate "random search" (RS) results in this research, we indicate that we are examining a very simple Monte Carlo technique that incorporates no information about the problem to help guide the search process, and simply randomly assigns harvest timing choices to management units. The process (Fig. 1) works like this: randomly assign a harvest timing (including the possibility of a no-harvest prescription) to all management units; evaluate the wildlife goals; if the resulting objective function is feasible, and better than the best solution that has been located to this point, save the solution as the best solution. We developed 100 random search solutions, each representing the best solution from an independent process that evaluated 2 million random solutions. Thus 2 million independent, and ran-

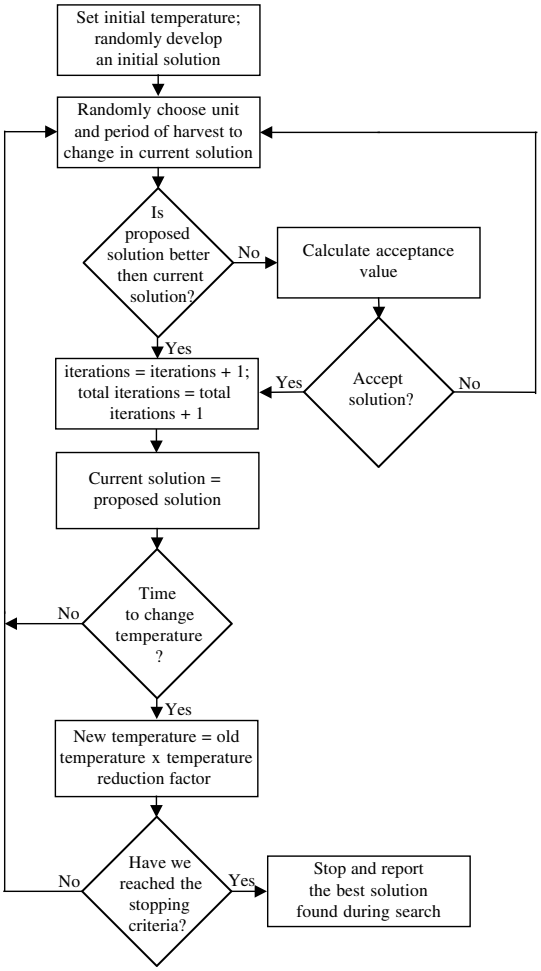

Fig. 2. A flow chart of the simulated annealing search process.

domly defined solutions are generated for each 100 attempts to solve the problem. This process is different from the following seven other processes, where generally only a small change is made to a solution from one iteration to the next. There may be other ways to enhance a Monte Carlo search process, but we have chosen to use a generic process here, random chance, to compare the other techniques against.

2.2.2 Simulated Annealing

Simulated annealing (SA) is a search technique that began to be used in a widespread manner in the early 1980s (Dowsland 1993). The ideas that form the basis for SA were first published by Metropolis et al. (1953) in an algorithm to simulate the cooling of materials in a heat bath – a process known as annealing. The approach is a Monte Carlo method that uses a local search in which a subset of solutions is explored by moving from one solution to a neighboring solution. To avoid converging and becoming stuck in a local maximum (or minimum) the procedure provides for an occasional acceptance of an inferior solution to allow it to move away from a local maximum. SA has been used in a wide variety of disciplines to solve optimization problems. In forestry, SA has been investigated by a number of researchers to solve spatial harvest scheduling problems involving adjacency constraints, including Lockwood and Moore (1992), Murray and Church (1995), and Öhman and Eriksson (1998). The SA process is illustrated in Fig. 2.

In our implementation of SA, we used the following parameters, which were based on several trial and error runs of the search process on the hypothetical landscape:

Problem A
Beginning temperature: 0.05
Ending temperature: 0.00001
Repetitions between temperatures: 400
Temperature reduction factor: 0.999

Problem B
Beginning temperature: 0.05
Ending temperature: 0.0001
Repetitions between temperatures: 400
Temperature reduction factor: 0.99

Problem C
Beginning temperature: 0.15
Ending temperature: 0.0001
Repetitions between temperatures: 200
Temperature reduction factor: 0.999

The beginning and ending temperatures were selected to provide an appropriate range of probabilities of acceptance of solutions during the

search process. The temperature reduction process was employed after the number of repetitions were made at each temperature level. Feasibility with respect to the constraints was evaluated as each new solution was proposed; infeasible proposed solutions were not allowed, and did not count toward the number of repetitions between temperatures.

2.2.3 Great Deluge Algorithm

The *great deluge algorithm* (GDA) is a recently developed variant on simulated annealing. It is similar to SA in that only a single change is considered to a "current" solution, the resulting temporary solution is evaluated, and a decision is made whether or not to convert the temporary solution to the current solution. The GDA was introduced by Dueck (1993) and proved superior to similar Monte-Carlo based algorithms in solving a 442-city and 532-city Travelling Salesman Problem. The form of the GDA as presented by Dueck (1993) consisted of using a single parameter in the determination of whether or not to convert the temporary solution to the current solution (and perhaps change to an inferior solution). The use of one parameter rather than two, as in a simulated annealing algorithm, is believed to de-sensitize the algorithm thus leading to equally good results even when parameter estimation and formulation is poor.

The GDA derives it name from the conceptual framework on which the algorithm works. Consider a problem where the objective is to find the highest elevation in a fictitious landscape by simply walking around the landscape and measuring the elevation. Logically you would want to continuously measure higher and higher ground rather than lower and lower ground (or the entire landscape). The GDA algorithm would start at some unknown location in the landscape, and subsequently it would begin to "rain without end", flooding the landscape and making it easier to locate the higher elevations. As the water rises, the GDA algorithm "walks" around the landscape trying to "keep its feet dry" (by only walking on higher and higher ground). However, if we were to further humanize this process, the algorithm will tolerate walking in water up to its ankles

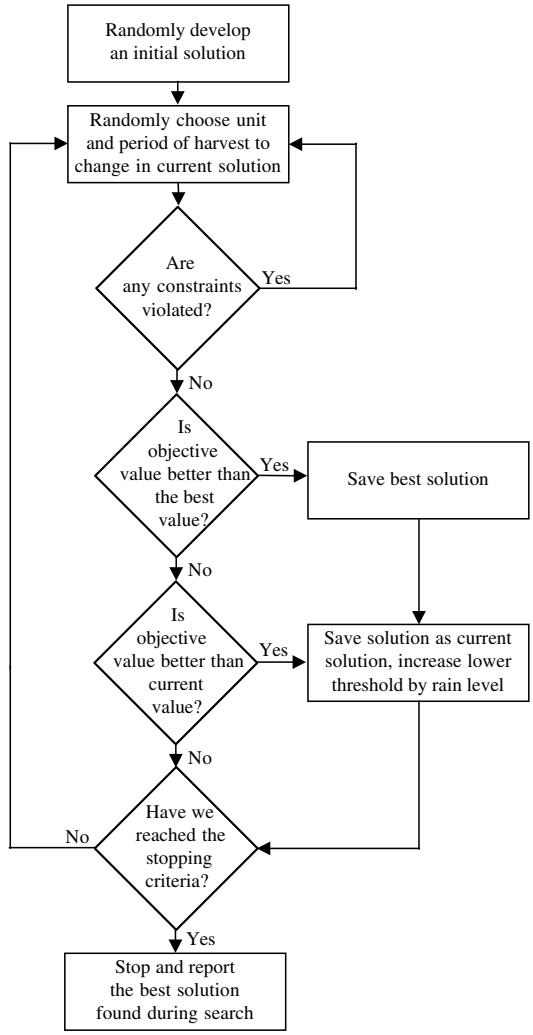

Fig. 3. A flow chart of the great deluge search process.

(accepting a small subset of lower quality solutions) and so is allowed to walk in some inundated areas with the hope that there is higher dry land nearby. Since the rain never ends, the water continues to rise and the amount of dry land and acceptable ankle-deep water diminishes until what is left is only (hopefully) the highest point in the landscape. The rain intensity in this process is typically constant but a process can be devised to make it rain more in the earlier stages of the search process, to rather quickly get to higher ground, and to reduce the computational time requirements.

The formulation of the GDA used here is primarily a stochastic process. The algorithm starts by generating a feasible random solution then calculating the objective function to obtain an initial objective function value (Fig. 3). Through a trial and error process (for each goal), a "subtraction" value is determined and used to subtract from the objective function value, the result of which is used as the initial "water level" (which represents the lower threshold value above which only solutions of this value are acceptable – i.e., getting the ankles wet). A programming loop is started that undergoes several steps until the stopping criteria is met. First, a random management unit is selected for a "move". A move represents changing the solution with a 1-opt move. The harvest volume, harvest age, and habitat constraints are then evaluated. If the constraints are violated, the move is rejected and another random management unit is selected for a move. If the constraints are not violated the new solution's objective function value is calculated. If the new solution is better than the current "best" solution, it becomes the best solution. If the new solution is better than the current solution, the rain level is increased, by an amount equivalent to a "rain" event, closing the gap between the lowest acceptable solution value and the current solution value. If the move's objective function value is lower than the lower threshold, the move is rejected. This process continues until the lower threshold level is equal to the best solution value. At this point, the search process is allowed to continue for a set number of additional iterations before stopping. The number of iterations and the level by which the water rises (increasing the lower threshold) were made through trial and error for each of the three planning problems.

In our implementation of GDA, we used the following parameters, which were based on several trial and error runs of the search process on the hypothetical landscape:

Problem A
Subtraction value: 1.00
Rain event: 0.00009
Additional iterations: 70 000

Problem B
Subtraction value: 0.05
Rain event: 0.00005
Additional iterations: 40 000

Problem C
Subtraction value: 0.08
Rain event: 0.000035
Additional iterations: 20 000

2.2.4 Threshold Accepting

Threshold accepting (TA) is similar to both simulated annealing and the great deluge process, and was introduced by Dueck and Scheuer (1990). TA, as implemented here, also examines a single change to a current solution, yet uses a process which has a different set of acceptance rules than SA. TA accepts every new (proposed) solution which is *not much worse* than the previous current solution (within a pre-set limit of the value of the current solution), whereas in SA there is only a small probability that a worse proposed solution would replace the current solution.

In the TA process, the initial threshold level T is set by the user, then a random initial solution is generated (Figure 4). A management unit and a proposed new harvest timing for that unit are then randomly selected. The difference (ΔE) between the resulting proposed solution (if the harvest timing were actually changed) and the current solution is computed by subtracting the current solution's objective function value from the proposed solution's objective function value. If ΔE is greater than $-T$, and the proposed solution is feasible with respect to the constraints, the proposed solution becomes the current solution. If ΔE is not greater than $-T$, and it has not been a "long time" (defined by the user as the number of iterations of the process using this T) since the quality of the best solution has changed, the process continues. If it has been a "long time" since the quality of the best solution has changed, the threshold is made smaller ($T = T - \Delta T$). We used three stopping criteria: (1) the number of iterations since the best solution has been improved (number of "non-improving iterations") exceeds a maximum level C; (2) the total number of search process iterations (number of "total iterations") exceeds a maximum level S; or (3) T reaches a level denoted as the stopping point. If any of these

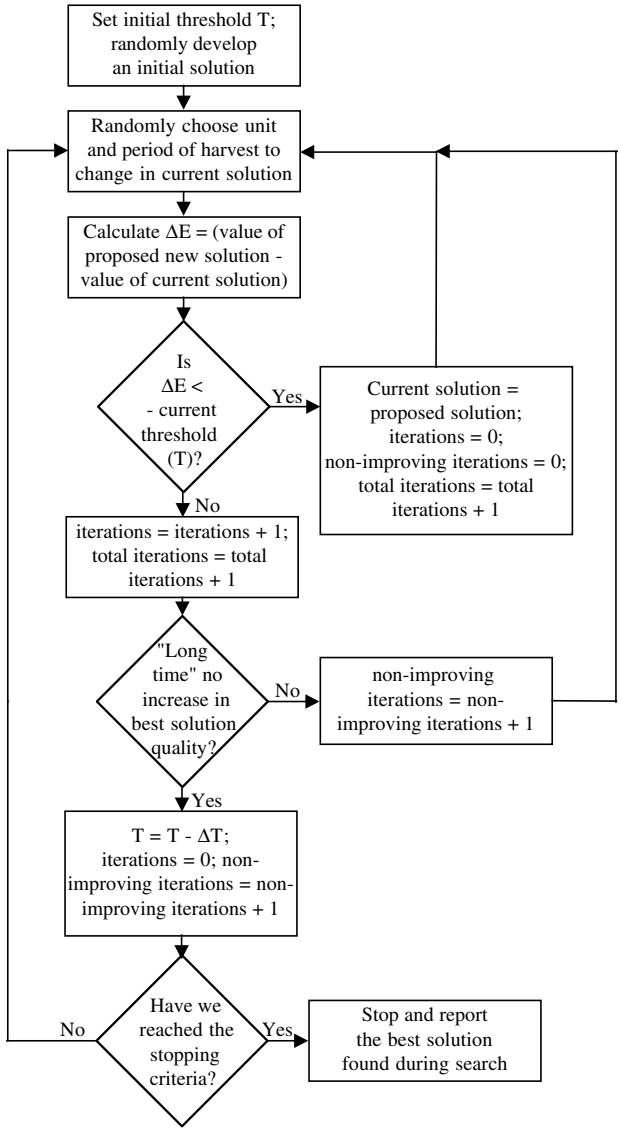

Fig. 4. A flow chart of the threshold accepting search process.

are true, the search process ends and the best solution is reported. Feasibility with respect to the constraints was evaluated as each new solution was proposed; infeasible proposed solutions were not allowed, and did not count toward the number of iterations within threshold levels.

In our implementation of TA, we used the following parameters, which were based on several trial and error runs of the search process on the hypothetical landscape:

Problems A, B, and C

T: 0.05

Change in T (ΔT): 0.001

Stopping point for T: 0.002

Maximum number of "total iterations" (S): 2 000 000

Maximum number of "non-improving iterations" (C): 200 000

Number of iterations with no change in solution value before changing T: 20 000

2.2.5 Tabu Search with 1-opt Moves

Tabu search originated as a method for solving real-world combinatorial problems in scheduling, and has been successfully applied to a number of important problems outside of forestry and wildlife management, such as telecommunications, transportation, shop sequencing, machine scheduling, and layout and circuit design problems (Glover 1990, Glover and Laguna 1993). Within forestry it has been applied, e.g., to problems formulated for scheduling timber harvests subject to adjacency (green-up) requirements (Murray and Church 1995), for meeting spatial goals for elk (Bettinger et al. 1997) and aquatic habitat (Bettinger et al. 1998). Our implementation here is similar to SA in that only a single change is considered to a "current" solution, the status of the proposed change and the resulting temporary solution are evaluated, and a decision is made whether or not to convert the temporary solution to the current solution. The neighborhood, in effect, is a full neighborhood of 1-opt moves, calculated by temporarily changing the timing of harvest (including considering a change to "no-harvest") of a single management unit. The neighborhood values can be considered to be the potential objective function values *if* the 1-opt move is made. Infeasible potential moves are assigned a potential objective function value so inferior that they are never selected as potential moves. The move selected from the neighborhood is the move with the best potential objective function value. The tabu state (z) is then considered, along with, perhaps aspiration criteria.

Tabu search, in general, is a hill-climbing procedure consisting of two key characteristics: (1) the search is constrained by considering certain choices as forbidden (i.e., tabu), and (2) when encountering a forbidden choice, the search can be freed by a memory function (aspiration criterion) that allows "strategic forgetting" that certain choices are forbidden (Glover 1989). The process we implemented is illustrated in Fig. 5. Feasibility is maintained at all times, thus strategic oscillation is not used here. Based on trial runs of the algorithm, where z values ranged from 50 to 400 moves, z values were set at 200 moves. The total number of iterations of the tabu search process for each of the 100 independent runs was limited to 5000, which was based on an examination of the number of iterations required to reach a steady state (see Bettinger et al. 1997), and the amount of time required for each independent run.

2.2.6 Tabu Search with 1-opt and 2-opt Moves

While λ-opt (1-opt, 2-opt, 3-opt, etc.) moves have been evaluated with heuristic search processes in the broader literature (e.g., Glover 1996, Hanafi and Freville 1998), 2-opt (and greater) moves have been little used in forestry, but have shown good results when applied to a forestry problem which contained an even-flow goal and adjacency constraints (Bettinger et al. 1999). 2-opt moves involve simultaneously changing an attribute of one management unit with that of another. These moves have been shown to reduce the magnitude of the impact on the objective function value, as compared to 1-opt moves, and allow a heuristic technique to refine the solution to a management problem. And, it is not necessarily true that two 1-opt moves, made in sequence, would produce the same solution as a single 2-opt move. Therefore, the use of 2-opt moves may allow refinements in the exploration of the solution space, and allow an exploration of more of the solution space than the 1-opt moves allow.

The tabu search process we implemented (TS2) is similar to the one described above, with a 1-opt (changing the harvest timing of a single unit) neighborhood being available to select moves from during every iteration of the process, yet a 2-opt (swapping the harvest timing of two units) neighborhood is also available every other iteration of the search process (Fig. 6). Again, feasibility is maintained at all times, and strategic oscillation is not used here. The best move from the current solution is temporarily selected after examining the available neighborhood(s). The tabu criteria is similar to that described above, with the unit / harvest timing combination being tabu for 1-opt moves, yet the unit / unit combination being tabu for 2-opt moves. If a 1-opt move is selected and permanently changes the solution, it is given a tabu state value of z, and the tabu state values of all other tabu moves (related to both 1-opt and 2-opt moves) are decreased by

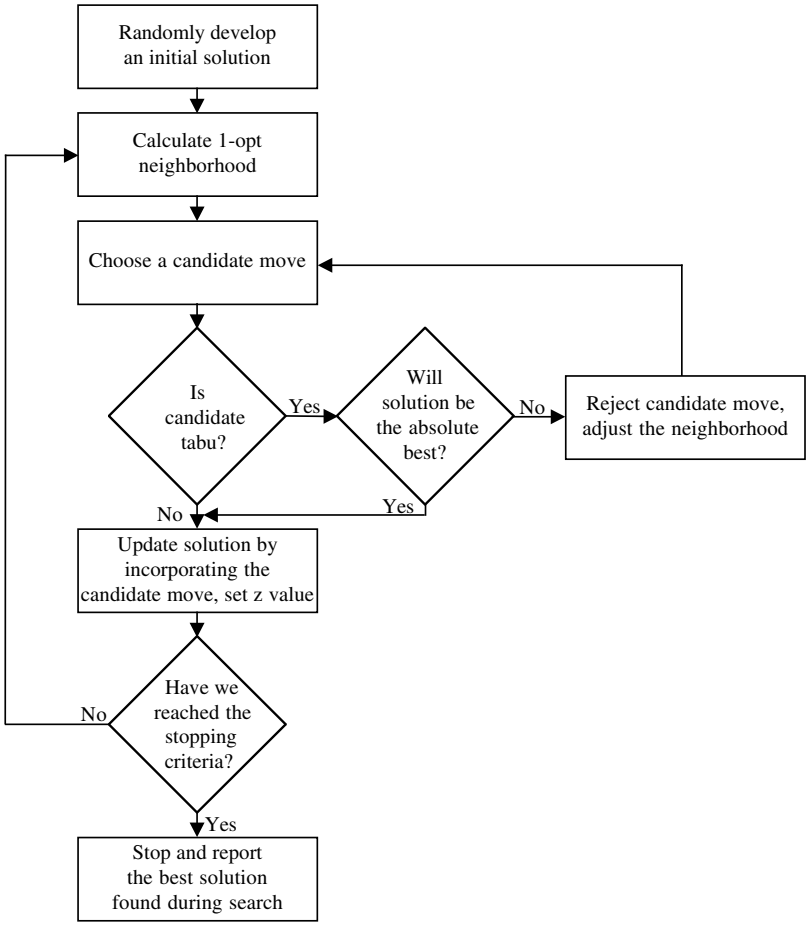

Fig. 5. A flow chart of the 1-opt tabu search process.

one until they equal 0 once again, and are then not tabu.

Aspiration criteria are still utilized if we find that the move chosen is tabu, as are the other operations in the 1-opt tabu search process described above. Based on trial runs of this algorithm, where z values ranged from 50 to 500 moves, z values were set at 400 moves. The total number of iterations of the tabu search process for each of the 100 independent runs was limited to 2000, which was based on an examination of the number of iterations required to reach a steady state (see Bettinger et al. 1997), and the amount of time required for each independent run.

2.2.7 Genetic Algorithm

Genetic Algorithms (GA) were developed initially by Holland (1975) and his associates in the 1970s. GAs are optimization heuristics that are used to search for good solutions to complex problems (Mullen and Butler 2000). Diverse areas such as music generation, genetic synthesis, strategic planning, and machine learning have profited from these methods (Srinivas and Patnaik 1994). In forestry, GAs have been applied, e.g., to forest operational planning and harvest scheduling problems by Falcao and Borges (2001), Lu and Eriksson (2000), and Mullen and Butler (2000).

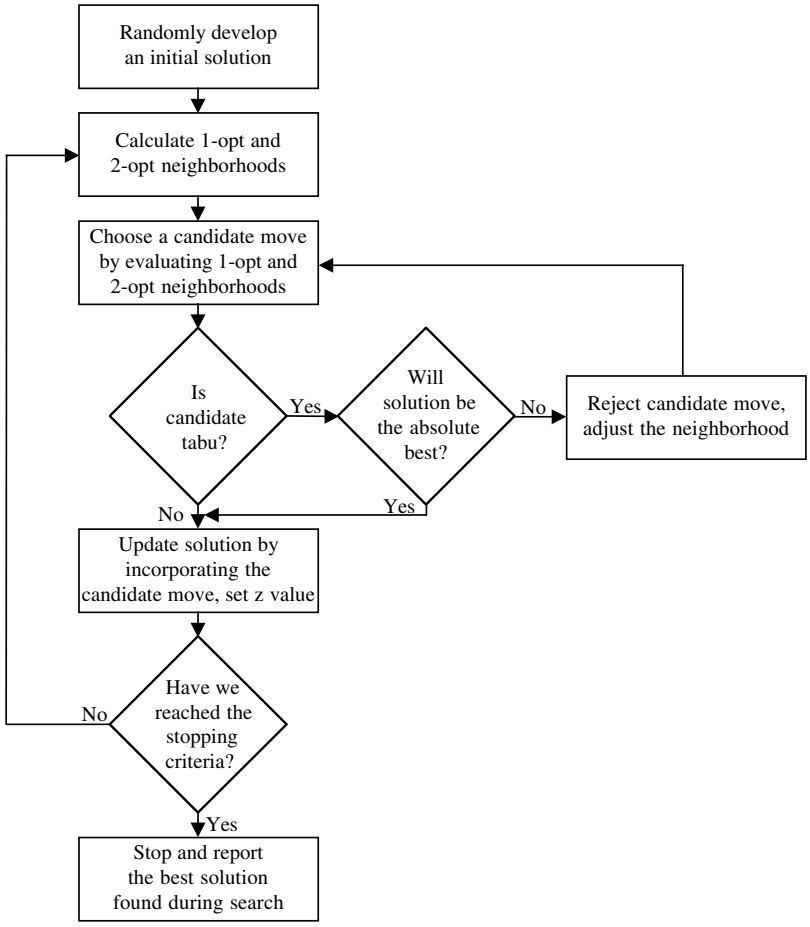

Fig. 6. A flow chart of the 1-opt and 2-opt tabu search process.

GAs are based on the mechanics of natural selection and genetics (Holland 1975). A GA technique starts with a set of feasible solutions (a population) with each solution corresponding to a chromosome. Solutions are selected from the population either randomly or according to their fitness (objective function value), and are combined to form new solutions (offspring). This process is repeated until a stop criterion (for example number of generations, improvement of the best solution, or homogeneity of solutions) is satisfied. Crossover and mutation are two basic aspects of GAs. A crossover routine denotes the place where two parents are split, then re-combined to form offspring, allowing beneficial genes on two different parents to be combined in their offspring and hopefully to produce better solutions. Mutation, generally applied in a random manner, provides background variation, and occasionally introduces beneficial material into chromosomes (Davis 1987).

In our implementation of a GA technique, we start with a randomly generated set (a population) of feasible solutions (chromosomes, Fig. 7). Population size was chosen after several initial trials, and was based on computing time and quality of the final solution generated from the technique. A chromosome (a single solution) consists of 74 genes representing units and each gene is encoded as a harvest period from 0 to 10 (0 means not harvesting the unit).

Each chromosome in the initial population is

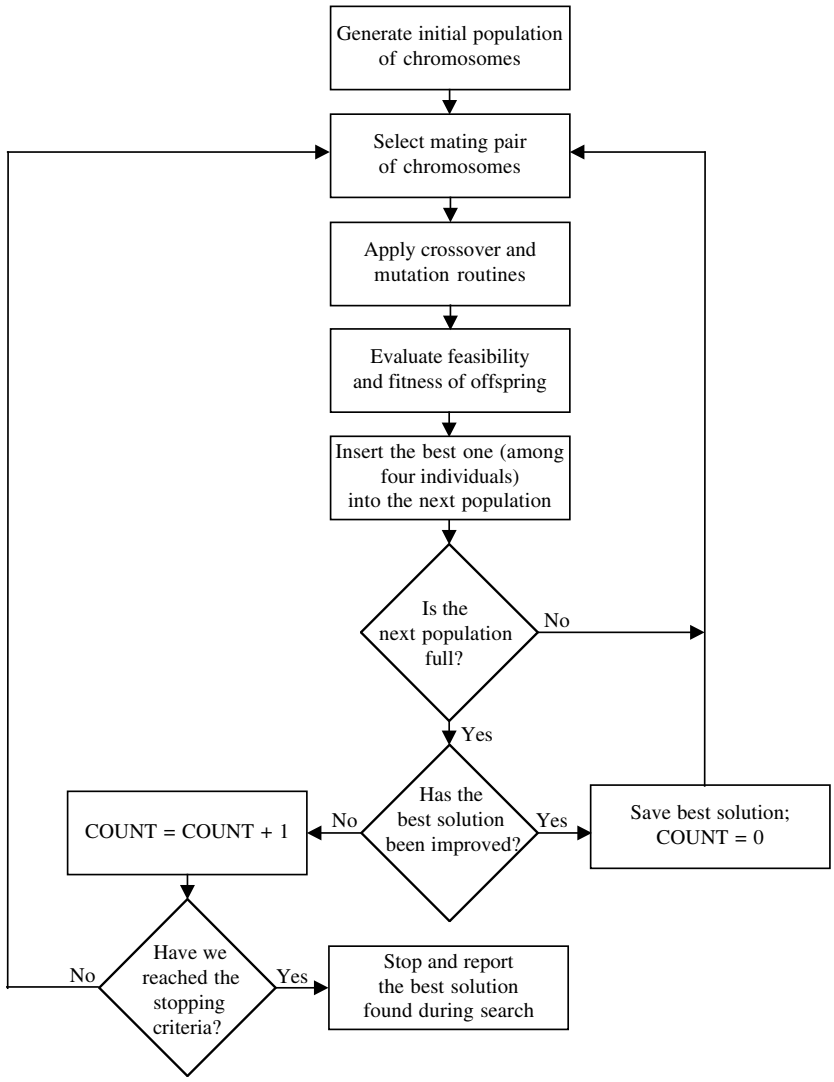

Fig. 7. A flow chart of the genetic algorithm search process.

evaluated by computing the objective function value, thus each solution must be feasible with respect to the constraints. One parent chromosome is then selected based on fitness (the better the fitness value [objective function value], the higher the chance of it being chosen), while the other parent is chosen randomly. They are then 'mated' by choosing a crossover point at random, then the crossover occurs, and two offspring chromosomes (two new solutions) result. For example, if we have two solutions X and Y, each having 5 harvest units,

X = (9,4,0,0,2)
Y = (6,2,7,4,0)

and if the crossover point is noted as being just before the management unit 3 values, the pieces prior to the crossover would be

X_1 (9,4) X_2 (0,0,2)
Y_1 (6,2) Y_2 (7,4,0)

and the resulting offspring would become:

X_1Y_2 (9,4,7,4,0)
X_2Y_1 (6,2,0,0,2)

A random mutation may then be applied to these offspring. If a random number on the range between 0 and 1 is less than the mutation probability (set to 0.01 in this implementation), the current harvest period of a randomly chosen gene (a harvest unit) will randomly change. After the harvest volume and wildlife goals are evaluated for the offspring, the best one of the feasible offspring (assuming it is feasible with respect to the volume and wildlife goals) and parents will be kept as new chromosomes in the next generation. If neither offspring are feasible with respect to the constraints, only the better parent is kept for the next generation. The process ends when 100 generations have passed without improving upon the very best solution located during the search process.

In our implementation of GA, we used the following parameters, which were based on several trial and error runs of the search process on the hypothetical landscape:

Problem A
Population size: 5000
Mutation rate: 0.01

Problem B
Population size: 1000
Mutation rate: 0.01

Problem C
Population size: 2000
Mutation rate: 0.01

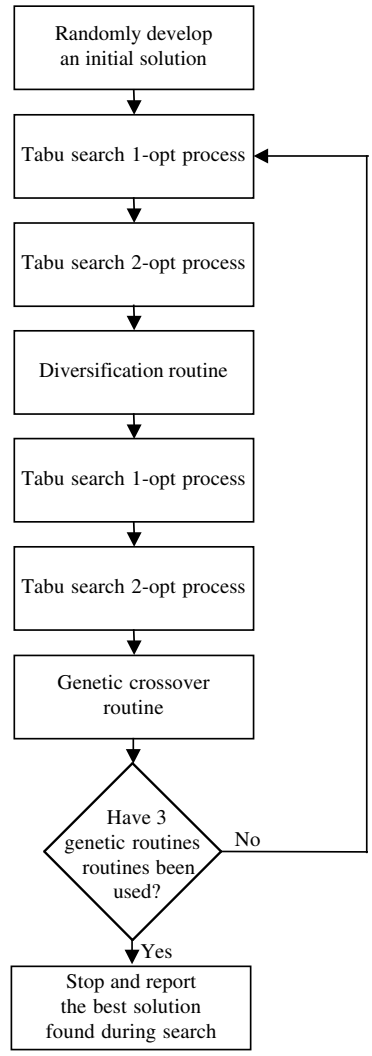

Fig. 8. A flow chart of the hybrid tabu search / genetic algorithm process.

2.2.8 Hybrid Genetic Algorithm / Tabu Search

The hybrid genetic algorithm / tabu search (GA/TS) heuristic technique utilizes techniques we have previously described: a 1-opt tabu search process, a 2-opt tabu search process, and a genetic algorithm crossover process (Boston and Bettinger 2002). In addition, the GA/TS technique uses a diversification routine in an attempt to move the search process to other regions of the solution space. In the tabu search processes of this technique a change is made to a current solution by either altering the harvest timing of a single, or of two, management units. In the genetic algorithm process, the two best solutions (saved in memory) are mated to form two new solutions.

The GA/TS technique begins with a random starting solution (Fig. 8). In this random start process, management units and their harvest timing are randomly selected until 10% of the harvest volume goal has been met in each time

period. The GA/TS technique then uses 1000 iterations of a 1-opt tabu search process, and subsequently 200 iterations of a 2-opt tabu search process. The two best solutions (one from the 1-opt process, the other from the 2-opt process) are saved in memory throughout the scheduling process. The diversification routine is then employed, where the management units which have been evaluated the least (so far) are scheduled for harvest. The diversification routine unschedules all management units from harvest, then schedules (by randomly selecting a harvest timing) the least-evaluated units until 10% of the volume goal has been met. This process essentially re-starts the heuristic by forcing into the solution those management units which were least used in the tabu search processes.

The 1-opt and 2-opt processes are then repeated before the genetic crossover routine is used. The two best feasible solutions (again, one from the 1-opt process, the other from the 2-opt process) found to this point in the search process become the parents for the mating. A random crossover point was determined, and the two chromosomes were split, and then re-combined to form two new solutions. The resulting child with the highest objective function value becomes the starting solution for another loop through the tabu search and diversification processes. For Problem C, if feasibility with regard to clearcut size limitation is not maintained, one (or more) of the units affecting infeasibility is randomly unscheduled from harvest until feasibility is once again achieved.

After 6 sets of 1-opt and 2-opt tabu search processes, 3 diversification routines, and 3 genetic crossover routines, the search process stops and reports the best solution that it located. Based on trial runs of the algorithm, the tabu state, z, for the 1-opt process was set to 100 iterations, while z for the 2-opt process was set to 20 iterations.

2.2.9 Comparing Heuristic Techniques

The solutions generated by the eight heuristic techniques are compared in several ways: the best solutions from 100 independent runs of each heuristic are compared; the minimum, maximum, mean, and standard deviation from each of the three planning problems are compared; the global optimum solution for each of the three problems is either generated or estimated, and the percentage of solutions within 1% of this value is presented; and while differences in computers may provide the least serious impediment to competitive testing (Hooker 1995), the time necessary to generate a single solution on a single computer (a Pentium III 550 MHz computer) is presented. More difficult to measure are the differences in coding skill, fine-tuning of algorithms, and testing of parameters (Hooker 1995), all areas which we fail to address here, but leave for further exploration. In addition, Hooker (1995) suggests that the amount of processing time per iteration, and its effect on total computation time, is important. This too, we leave for future exploration.

When using heuristic techniques, one cannot be certain that the global optimum solution to a planning problem will be found, nor that the resulting solutions are even close to the global optimum. To evaluate the quality of the solutions that are produced by heuristic techniques, one can solve the global optimum solution to a problem using a traditional mathematical technique, and subsequently the comparisons can be made. For example, an integer programming formulation was developed for the non-spatial goals (Problem A), and the optimal solution produced was 9.8105. This is difficult to do for more complex goals, however, thus integer programming formulations were not developed for the minimum patch size or complementary patch problems, since these are in fact very complex problems. A relaxed linear programming problem, (where some of the spatial constraints are ignored) could also be formulated and assumed as an upper-bound on the potential solutions derived from heuristic techniques, although we did not develop relaxed linear programming formulations for the three wildlife planning problems. Finally, it may be possible to develop an estimate of the global optimum solution using extreme value theory, which is described in Bettinger et al. (1998), Los and Lardinois (1982), and Golden and Alt (1979). We developed estimates of the global optimum solutions for the three wildlife planning problems, and compared them against the solutions generated by the heuristic techniques.

Table 1. Quality of the best solutions generated by the eight heuristic techniques.

Heuristic technique	Problem A	Problem B	Problem C
Random Search (RS)	8.4607	1.3548	1.7254
Simulated Annealing (SA)	9.8079	2.0475	3.0323
Great Deluge (GDA)	9.8084	2.0475	3.0559
Threshold Accepting (TA)	9.8085	2.0475	3.0591
Tabu Search 1-opt (TS1)	9.7434	2.0045	3.0158
Tabu Search 1-opt and 2-opt (TS2)	9.8093	2.0475	3.0522
Genetic Algorithm (GA)	9.7941	1.9912	2.9823
Genetic Algorithm / Tabu Search (GA/TS)	9.7996	2.0475	3.0148

2.3 Hypothetical Landscape

The landscape we apply the three increasing difficult wildlife planning goals to is 2500 acres in size, and consists of 74 management units. A GIS database of the hypothetical landscape, along with a list of management unit acres, initial age (age during period 1, if not harvested during period 1), and potential volumes is available on the internet (http://cof.orst.edu/cof/fr/people/bettingp/wlc/index.htm). Adjacent units are those units that share an edge, rather than both an edge or a corner (point). This is a hypothetical landscape, and can be used as a standard model for others to use in their efforts to solve these problems. Users of this data should note that one management unit consists of hardwood stands and represents riparian areas, thus while it contributes to the total number of acres on the landscape, it does not count toward wildlife goal achievement.

3 Results

To compare the results, we produced 100 solutions with each of the eight heuristic techniques for each of the three wildlife planning problems. These 100 resulting solutions can be considered an independent sample from a population of solutions, since the starting point for each of the search processes was a randomly developed solution. We employ several methods to evaluate the quality of the resulting solutions, including a comparison of the statistics (mean, median, mode) of the solution values, a discussion of the quality of the entire set of 100 solutions, and a discussion of the estimated global optimum solutions from the sample solutions of each technique.

Of the eight heuristic techniques, six found very good solutions to Problem A, the non-spatial wildlife goals (Table 1). The values represented in Table 1 represent the best solution of the 100 sample solutions generated by the eight techniques. Only the RS and TS1 techniques located considerably lower-quality solutions. Problem B, with the minimum patch size goals, was more difficult for the GA technique to solve, and RS and TS1 also produced lower results than the other 5 techniques. In Problem C, with the complementary, adjacent patch goals, four techniques produced the best solutions: SA, GDA, TA, TS2. The TS1, GA, GA/TS techniques also produced very good solutions to Problem C.

Now that we have illustrated how the "best" solution from each technique compares to the best from the other techniques, one may ask how the quality of the entire set of 100 solutions compare, to give us an understanding of the variation in solution quality among techniques. In Problem A (Table 2) several notable results can be seen. First, SA, GDA, and TS2 produced the highest minimum solution values, so their worst solutions were better than the best solution for TS1, yet TA had a much lower minimum solution value than SA, GDA, and TS2. However, the average solution values for SA, GDA, TA, and TS2 were all very similar. Of the 100 sample solutions produced by each technique, those produced by SA had the lowest variation, and those produced by TS1 had the highest variation in solution quality.

In Problem B, while 5 of the techniques pro-

Table 2. Statistics regarding the sample of 100 solutions generated by the eight heuristic techniques for Problem A, the non-spatial wildlife goals.

Heuristic technique	Maximum	Minimum	Mean	Standard deviation
RS	8.4607	8.3168	8.4010	0.0396
SA	9.8079	9.7775	9.7933	0.0055
GDA	9.8084	9.7493	9.7838	0.0106
TA	9.8085	9.6479	9.7750	0.0354
TS1	9.7434	9.2481	9.5222	0.1092
TS2	9.8093	9.7651	9.7928	0.0106
GA	9.7941	9.6481	9.7456	0.0275
GA/TS	9.7996	9.7060	9.7579	0.0174

Table 3. Statistics regarding the sample of 100 solutions generated by the eight heuristic techniques for Problem B, the minimum patch size wildlife goals.

Heuristic technique	Maximum	Minimum	Mean	Standard deviation
RS	1.3548	1.2387	1.2941	0.0359
SA	2.0475	1.9485	2.0330	0.0156
GDA	2.0475	2.0057	2.0277	0.0114
TA	2.0475	2.0033	2.0342	0.0112
TS1	2.0045	1.6123	1.9226	0.0666
TS2	2.0475	2.0263	2.0397	0.0088
GA	1.9912	1.8110	1.9309	0.0356
GA/TS	2.0475	2.0181	2.0410	0.0087

Table 4. Statistics regarding the sample of 100 solutions generated by the eight heuristic techniques for Problem C, the complementary, adjacent patch wildlife goals.

Heuristic technique	Maximum	Minimum	Mean	Standard deviation
RS	1.7254	1.6047	1.6680	0.0415
SA	3.0323	2.9228	2.9897	0.0217
GDA	3.0559	3.0008	3.0282	0.0130
TA	3.0591	2.9769	3.0403	0.0148
TS1	3.0158	2.2683	2.7212	0.1666
TS2	3.0522	2.5932	2.9817	0.0791
GA	2.9823	2.5317	2.8553	0.0648
GA/TS	3.0148	2.8087	2.9555	0.0370

duced what seems to be the global optimum solution (2.0475), the minimum solution value produced by TS2 and GA/TS was higher than that produced by the other techniques (Table 3). These two techniques also had the lowest variation in solution values. The mean values for SA, TA, TS2, and GA/TS were also very good compared to the other techniques. In Problem C, TS2 produced one of the best overall solutions, yet its lowest valued solution was much lower than that produced by SA, GDA, or TA (Table 4). The story for TS1 is similar; it produce a fairly good maximum, but its minimum solution value was quite low. This indicates that while the potential to produce good solutions using TS1 or TS2 is good, there is a chance, if only a few solutions are generated, that the resulting solutions are not very good. The mean values, however, for SA, GDA, TA, TS2, and GA/TS were all very similar. The techniques which produced solutions with the most variation in solution quality were TS1, TS2, and GA, although TS1 and GA solution values were generally lower than TS2 solution values.

Since the global optimum solution for Problem A was solved using integer programming techniques, we can directly compare it to the heuristic results. The best solutions from SA, GDA, TA, and TS2 were all within 0.02% of the global optimum solution. The best solution from TS1 (within 0.69% of the global optimum), GA (within 0.17%), GA/TS (within 0.12%), were also very good. The best RS solution was within 13.76% of the global optimum.

Estimates of the global optimum solution for Problems B and C were generated using techniques described in Bettinger et al. (1998), which views the set of solutions from each technique as an independent sample, continuously distributed, from a population of solution values. A three-parameter Weibull curve is fit to the sample solutions, and the location parameter of the resulting Weibull curve is used as the estimate of the global optimum. To verify the goodness of fit of these curves, the distribution of sample solutions was rotation about the location parameter, and re-fit using BestFit software (Palisade Corporation 1997), which fits a two-parameter Weibull curve (assumes the location parameter has the value 0), and also tests the goodness of fit using Chi-square and Anderson-Darling statistics.

The estimates of the global optimum solutions are presented in Table 5, and show limited usefulness of this approach to gauge the quality of

Table 5. Estimated global optimum from the eight heuristic techniques.

Heuristic technique	Problem A	Problem B	Problem C
RS	8.4612	1.4609	1.7585
SA	9.8109 [a]	2.0480	3.0361 [a]
GDA	9.8115 [a]	2.0477	3.0633
TA	9.8086	2.0490	3.0593 [a]
TS1	9.8011 [a]	2.0051	3.0657 [a]
TS2	9.8110 [a]	2.0490	3.0522
GA	9.7959 [a]	1.9958 [a]	2.9869
GA/TS	9.8051	2.0501	3.0184 [a]

[a] Results were not rejected by Chi-square, Kolmogorov-Smirnov, nor Anderson-Darling test statistics using BestFit software (Palisade Corporation 1997).

solutions generated by heuristic techniques. For example, in Problem B only one technique (GA) produced a distribution of results which when fit with a Weibull curve, was not rejected by the test statistics, yet we know this estimated global optimum (1.9958) to be well below other actual values produced by other heuristic techniques. In Problem C we see similar results from the estimated global optimum values for SA and GA/TS heuristic techniques. The estimated global optimum values from TA and TS1 are both plausible, but TS1 had more variation in its results than TA, and TA results were clustered more closely around its estimated global optimum. For Problem A, we found three estimated global optimum values (from SA, GDA, and TS2) which were very close to the integer programming solution, providing some reassurance that using extreme value theory to estimate the global optimum may have some value, yet researchers and planners should take caution when using this process, as noted by the results above and by Boston and Bettinger (1999).

We compared the eight heuristic techniques one final way using the information we have compiled regarding the optimum solutions for each of the three problems. We first make some assumptions regarding the best possible solution values from the three planning problems. The optimum solution for Problem A (9.8105) was derived from the integer programming results, for Problem B (2.0475) was concluded based on the consistency which several of the techniques located these solutions and found no better, and for Problem C (3.0593) was derived from the estimate of the global optimum using the TA technique. We then ask how many, of the 100 sample solutions generated by each heuristic technique, were within 1% of these values. As Table 6 shows, there are some techniques which are well suited to certain problems, while other techniques, using our implementation of the techniques, may need some refinements to be able to consistently produce higher quality solutions.

Table 6. Percentage of sample solutions that are within 1% of the global optimum or assumed "best" values for each of the three planning problems.

Heuristic technique	Problem A	Problem B	Problem C
RS	0	0	0
SA	100	69	2
GDA	100	68	67
TA	89	77	81
TS1	3	0	0
TS2	100	82	22
GA	87	0	0
GA/TS	97	80	0

Table 7. Average time to develop a single solution for each of the three planning problems (minutes), using a Pentium III 550 MHz processor, and all algorithms coded in the C programming language.

Heuristic technique	Problem A	Problem B	Problem C
RS	5.0	5.2	13.0
SA	0.8	0.3	8.0
GDA	0.3	1.5	9.0
TA	5.0	12.5	22.0
TS1	4.5	11.5	27.0
TS2	1.5	12.0	30.0
GA	4.3	4.0	45.0
GA/TS	12.0	50.0	72.0

Solution times for each of the eight heuristic techniques is presented in Table 7. These solution times should be viewed from a broad perspective, since the heuristic programs were developed, in some cases, independently of the others, and standard logic or protocols were not employed. What is clear is that solution times increase as problem complexity increases, and that solution

time increases as the complexity of heuristic process increases. We found that the opinion of each researcher varied regarding the degree of complexity required to develop each technique. The level of familiarity each of us has with the intricacies of the eight search techniques probably influences these opinions.

4 Discussion

If an organization makes a decision to utilize a heuristic programming technique to develop a land management activity schedule, the level of sophistication of the resulting technique will vary depending on the type of system desired and the time allowed to develop the system. For example, heuristic programming techniques can be closely integrated with geographic information systems (GIS) or simply linked to GIS via the transfer of certain databases (inventory, adjacency, etc.) as was illustrated with each of the eight techniques in this research. In addition, the programming language employed is important, as some languages may provide efficiencies in programming logic, may provide faster computations for similar tasks, may be easier to use (for the people involved in scheduling efforts), and may result in a product with better end-user acceptability.

The time to develop a solution is a function of the programming skill employed, which determines the efficiency at which the search proceeds. Although none of the researchers were professional programmers, all had considerable programming experience, yet each may use different programming logic, and each may have preferences for certain processes, such as checking solution values, which may affect how fast a solution is ultimately generated. Thus the efficiency of generating solution values for the eight techniques should be viewed in a general sense. What is important is that the time to generate an adequate number of solutions should be short enough so that an analyst has time to develop alternative formulations and thus to test the robustness of the model and the planning problem. Others exploring the use of heuristics should keep in mind that each method does require a certain amount (defined by the comfort of the programmer) of skill and creativity to make a complex heuristic perform both well, and fast.

When considering development time for a heuristic scheduling process, analysts should consider the time to develop the scheduling code, to verify the logic, to develop the databases, and to develop graphic capabilities. Code development generally consists of input and output functions, logic to measure or evaluate management goals, logic to schedule activities, and criteria to determine when to stop the process. Verification of this code is time-consuming, and may consist of tracing the location of errors with "print statements" located at strategic points in the computer code, manually simulating the scheduling process to determine what the process should be doing, checking loops and the conditions that satisfy entering or leaving a loop, and isolating sections of code, checking them separately (Rojiani 1996). Manual verification of solutions on maps is a simple, yet invaluable technique that should be employed. In fact, some spatial scheduling problems may only be noticed once the maps have been developed. A standard program was used here to check the quality of the solutions to each problem; it was developed independently of the eight heuristic techniques. A separate goal evaluation technique is often employed within a scheduling problem to periodically examine the solution values and to verify feasibility. If one were using techniques such as strategic oscillation, which allows some deviation from feasibility during the search process, a periodic independent check of feasible solutions may be appropriate.

Database development is often one of the most under-appreciated tasks in planning efforts, and often scheduling problems can be traced to inadequate examination of the quality of the databases. Graphics capabilities are available for most programming software packages; a decision one must make regarding viewing the graphics is whether one needs to view the solution within a programming language structure, or within GIS. This, of course, assumes that the two are not integrated, and is moot if they are, in fact, integrated.

There are a variety of trade-offs associated with the eight techniques employed in this research (Table 8). Some of these techniques examine only

Table 8. Trade-offs associated with the eight heuristic techniques employed in this research.

Heuristic technique	No. of changes per iteration	Random or deterministic changes	Level of acceptable change per iteration	Speed per iteration
RS	multiple	random	unlimited	fast
SA	one	random	limited	fast
GDA	one	random	limited	fast
TA	one	random	limited	fast
TS1	one	deterministic	unlimited	moderate
TS2	one/two	deterministic	unlimited	slow
GA	multiple	random	unlimited	slow
GA/TS	varies[a]	both	unlimited	slow

[a] The number of changes per iteration depends on the type of search process used at each iteration: 1-opt tabu search, 2-opt tabu search, diversification, or genetic crossover.

a single change to a solution with each iteration of an algorithm, while others examine multiple changes to a solution with each iteration. Many of the techniques use random changes to a solution to move about the solution space, in fact tabu search, which generally uses deterministic changes, can be formulated to use random changes as well. The level of change allowed in the value sequential solutions that are generated can vary considerably, especially with Monte Carlo techniques that allow multiple changes to sequentially generated solutions. Thus the level of change may vary widely between iterations of the search process. TA and GDA generally have a "floor" below which no worse solutions are allowed, and SA can be viewed as having a floor as well, yet there is a chance that more inferior solutions are allowed in a comparable SA technique. Finally, speed is an important factor to those who desire to develop a high volume of good solutions in a reasonable amount of time. While speed is also related to the type of programming language employed, the time required to move through a single iteration of these techniques also varies based on the complexity of the scheduling process. Certainly GA and TS2 techniques can be developed which are fast, but comparable SA, TA, and GDA techniques will be faster. What is not clear is how the size of a planning problem may influence solution times. As problems get quite large, the ability to use parallel processing techniques may greatly increase the efficiency of those techniques which are most suited (TS and GA), since many different potential solutions (in the case of TS, filling out the neighborhood; in the case of GA, developing a population) must be generated before a decision is made.

In addition, the flexibility and complexity of the techniques may prevent or facilitate alternative formulations of the planning problems and interpretation of results. For example, these techniques do not require the generation of a detached coefficient matrix common to linear programming problems – the goals and constraints are embedded in the computer programming code that allows the heuristics to solve the planning problems. Whether the programming code is developed in such a way to allow alternative specifications of the problem is problematic, and not necessarily limited to the use of a subset of the eight heuristics. We have, through our experience, seen both extremes. The development of a flexible heuristic would require an investment in an interface that would allow a variety of variables related to the problem formulation to be modified by the user prior to solving the problem.

While we have provided some insight into the relative differences of eight heuristic techniques, it is possible that each of the eight techniques described here can be modified to produce better solutions than we have illustrated. Our intent was to develop the standard techniques, with some investigation into efficiencies such as using an appropriate tabu tenure in the tabu search techniques, or the appropriate cooling schedule for the SA technique. However, others may find that there are processes that can enhance these search techniques, such as strategic oscillation or a variety of λ-opt moves or a different integration of multiple search techniques. The issue of allow-

ing infeasible solutions to become valid local optima (thus using some sort of penalty function to temper the infeasibilities) in the search for the global optimum is an area of debate among those creating and using heuristic search algorithms. While some may argue that not allowing the search process to examine infeasible solutions restrictive, we welcome others to develop search algorithms that allow these infeasibilities, and subsequently to compare those results with the work we have performed here.

A good example of a technique that may benefit from further developmental work is the GA/TS technique. The GA/TS technique does provide consistently good solutions to the non-spatial problem as well as the minimum patch size problem, and given these results, should be appropriate for problems with clearcut size limitations that utilize adjacency constraints. However, the technique needs more developmental work if it is to be applied to complex planning problems that have goals similar to the complementary patch problem (Problem C). Future developmental work on this hybrid heuristic may explore several areas, such as the use of longer tabu tenures for the two tabu search portions of the technique, or different tenures for each of the two techniques. In addition, an expanded use of the genetic crossover routine may allow a wider search into the solution space, as might the use of mutation processes. An exploration into diverse selection criteria may also prove to result in better solutions for complex spatial problems. Finally, a more integrated use of the different search techniques may be beneficial, since each was used for a considerable amount of time before switching to another. Integrating techniques more frequently (as in the TS2 technique) could lead to better solutions to these complex problems.

As for the GA technique, one of the challenges is in keeping the solutions, after crossover, in the feasible region of the solution space. A GA in its "standard form" may not be a highly efficient model for sequencing and scheduling problems, especially if spatial constraints are included. The generation of a multitude of infeasible offspring will certainly reduce the usefulness of a GA technique, thus developing computer logic to prevent this from occurring may be appropriate. This problem, however, is not unique to GA techniques, as RS, SA, GDA, and TA techniques can also spend a lot of time evaluating solutions that are spatially infeasible, since the management unit and harvest timing are randomly chosen.

The size of the population chosen for a GA technique may also affect the efficiency in which good solutions are generated. A large population allows one to have a wide variety of genes, but slows down the processing time of a GA technique. Appropriate population sizes are generally chosen after several trial and error runs of the algorithm. Again, the appropriate choice of the parameters of a search process is not unique to GA techniques, as all of the others (except RS) require some user interaction to find an appropriate set of parameters to enable them to find good solutions in an efficient manner.

5 Conclusions

As forest management evolves in an increasingly complex regulatory environment, we will likely see more use of spatial restrictions and nonlinear goals in forest plans. As a result, many real world problems are becoming too complex to be solved with classical optimization techniques. In the future, computer software and hardware may progress to the point where classical techniques are once again useful for solving large combinatorial problems, yet in the meantime more reliance may be placed on simulation and optimization with heuristics. The ultimate goal of using heuristic techniques is to produce high quality solutions in short amounts of time to problems with nonlinearities or combinatorial relationships. This research has hopefully provided more insight into the performance differences of a variety of heuristic techniques on increasingly complex planning problems. Readers should not view the results as universal truths, however, and we encourage others to formulate these problems with their most promising techniques and compare the resulting solutions to the solutions presented here.

In the future, the need for a standard set of data and criteria for evaluation of these techniques seems appropriate. We have provided a database and three problems here for others to use as a start. In addition, the need for evaluating the cri-

teria for stopping a search process is noteworthy due to the length of time some of the processes required to arrive at a solution. A discussion of the appropriateness of heuristic techniques is also needed, since their advantages, while clear to some, are unclear to others more enthused with classical optimization techniques. The main concern here is whether the level of effort involved with using heuristics is worthwhile, since some believe that the value associated with implementing a relaxed LP solution may not be much different than the value of implementing a spatially feasible solution generated by a heuristic technique. An evaluation of the time and cost of both alternatives may therefore enhance this debate.

Literature Cited

Arthaud, G.J. & Rose, D. 1996. A methodology for estimating production possibility frontiers for wildlife habitat and timber value at the landscape level. Canadian Journal of Forest Research 26: 2191–2200.

Bettinger, P., Boston, K. & Sessions, J. 1999. Intensifying a heuristic forest harvest scheduling search procedure with 2-opt decision choices. Canadian Journal of Forest Research 29: 1784–1792.

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

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

Boston, K. & Bettinger, P. 1999. An analysis of Monte Carlo integer programming, simulated annealing, and tabu search heuristics for solving spatial harvest scheduling problems. Forest Science 45(2): 292–301.

— & Bettinger, P. 2002. Combining tabu search and genetic algorithm heuristic techniques to solve spatial harvest scheduling problems. Forest Science 48(1): 35–46.

Csuti, B., Polasky, S., Williams, P.H. & others. 1997. A comparison of reserve selection algorithms using data on terrestrial vertebrates in Oregon. Biological Conservation 80: 83–97.

Daust, D.K. & Nelson, J.D. 1993. Spatial reduction factors for strata-based harvest schedules. Forest Science 39(1): 152–165.

Davis, L. 1987. Genetic algorithms and simulated annealing. Pitman, London. 216 p.

Dowsland, K.A. 1993. Simulated annealing. In: Reeves, C.R. (ed.). Modern heuristic techniques for combinatorial problems. p. 20–69. John Wiley & Sons, Inc., New York.

Dueck, G. 1993. New optimization heuristics: The great deluge algorithm and the record-to-record travel. Journal of Computational Physics 104: 86–92.

— & Scheuer, T. 1990. Threshold accepting: A general purpose optimization algorithm appearing superior to simulated annealing. Journal of Computational Physics 90: 161–175.

Falcao, A.O. & Borges, J.G. 2001. Designing an evolution program for solving integer forest management scheduling models: An application in Portugal. Forest Science 47: 158–168.

Glover, F. 1989. Tabu search – Part I. ORSA Journal of Computing 1(3): 190–206.

— 1990. Tabu search – Part II. ORSA Journal of Computing 2(1): 4–32.

— 1996. Finding a best traveling salesman 4-opt move in the same time as a best 2-opt move. Journal of Heuristics 2: 169–179.

— & Laguna, M. 1993. Tabu search. In: Reeves, C.R. (ed.). Modern heuristic techniques for combinatorial problems. p. 70–150. John Wiley & Sons, Inc., New York.

Golden, B.L.& Alt, F.B. 1979. Interval estimation of a global optimum for large combinatorial problems. Naval Research Logistics Quarterly 26(1): 69–77.

Haight, R.G. & Travis, L.E. 1997. Wildlife conservation planning using stochastic optimization and importance sampling. Forest Science 43(1): 129–139.

Hanafi, S. & Freville, A. 1998. An efficient tabu search approach for the 0–1 multidimensional knapsack problem. European Journal of Operational Research 106: 659–675.

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

Hoganson, H.M.& Rose, D. 1984. A simulation approach for optimal timber management scheduling. Forest Science 30(1): 220–238.

Holland, J.H. 1975. Adaptation in natural and artificial systems. Univ. of Michigan Press, Ann Arbor, MI.

Hooker, J.N. 1995. Testing heuristics: We have it all wrong. Journal of Heuristics 1: 33–42.

Johnson, D.H. & O'Neil, T.A. 1999. Wildlife habitats and species associations in Oregon and Washington: Building a common understanding for management. Washington Department of Fish and Wildlife, Olympia, WA.

Kangas, J. & Pukkala, T. 1996. Operationalization of biological diversity as a decision objective in tactical forest planning. Canadian Journal of Forest Research 26: 103–111.

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

Los, M. & Lardinois, C. 1982. Combinatorial programming, statistical optimization and the optimal transportation network problem. Transportation Research 16B: 89–124.

Lu, F. & Eriksson, L.O. 2000. Formulation of harvest units with genetic algorithms. Forest Ecology and Management 130: 57–67.

Metropolis, N., Rosenbluth, A., Rosenbluth, M., Teller, A. & Teller, E. 1953. Equation of state calculations by fast computing machines. Journal of Chemical Physics 21: 1087–101.

Mullen, D.S. & Butler, R.M. 2000. The design of a genetic algorithm based spatially constrained timber harvest scheduling model. In: Proceedings of the Seventh Symposium on Systems Analysis in Forest Resources. USDA Forest Service, North Central Experiment Station, St. Paul, MN. General Technical Report NC-205. p. 57–65.

Murray, A.T. 1999. Spatial restrictions in harvest scheduling. Forest Science 45(1): 45–52.

— & Church, R.L. 1995. Heuristic solution approaches to operational forest planning problems. OR Spektrum [Operations Research] 17: 193–203.

Nelson, J. & Brodie, J.D. 1990. Comparison of a random search algorithm and mixed integer programming for solving area-based forest plans. Canadian Journal of Forest Research 20: 934–942.

O'Hara, A.J., Faaland, B.A. & Bare, B.B. 1989. Spatially constrained timber harvest scheduling. Canadian Journal of Forest Research 19: 715–724.

Öhman, K. & Eriksson, L.O. 1998. The core area concept in forming contiguous areas for long-term forest planning. Canadian Journal of Forest Research 28: 1032–1039.

Palisade Corporation. 1997. BestFit User's Guide. Palisade Corporation, Newfield, NY.

Pressey, R.L., Possingham, H.P. & Day, J.R. 1997. Effectiveness of alternative heuristic algorithms for identifying indicative minimum requirements for conservation reserves. Biological Conservation 80: 207–219.

Pulkki, R. 1984. A spatial database – heuristic programming system for aiding decision-making in long-distance transport of wood. Acta Forestalia Fennica 188. 89 p.

Rojiani, K.B. 1996. Programming in C with numerical methods for engineers. Prentice-Hall, Inc., Upper Saddle River, NJ.

Srinivas, M. & Patnaik, L.M. 1994. Genetic algorithms: A survey. Computer 1994 (June): 17–26.

Valsta, L. 1993. Stand management optimization based on growth simulators. The Finnish Forest Research Institute – Research Papers 453.

Weintraub, A., Jones, G., Magendzo, A., Meacham, M. & Kirby, M. 1994. A heuristic system to solve mixed integer forest planning models. Operations Research 42(6): 1010–1024.

— , Jones, G., Meacham, M., Magendzo, A., Magendzo, A. & Malchuk, D. 1995. Heuristic procedures for solving mixed-integer harvest scheduling – transportation planning models. Canadian Journal of Forest Research 25: 1618–1626.

Total of 44 references