ABSTRACT. In this article, basic concepts of both genetic algorithms and evolution program design are presented. An evolution program is presented to solve a Model I harvest scheduling problem with 0-1 decision variables for the management alternatives for each stand, with annual constraints on harvested volume. An appropriate data structure (i.e., chromosome representation) is presented, as well as modified selection, crossover, and mutation strategies specially designed for application to large forest scheduling problems. Emphasis is on designing an efficient evolution program to address the complexity of large integer problem model solving and to provide both strategic and operational guidance to forest managers. A new stopping criterion for this iterative heuristic based on the asymptotic behavior of the evolution process is further presented. The new evolution program is applied to a two-product timber harvest scheduling problem in Portugal with a temporal horizon extending to seventy 1 yr periods. Results from 50 test computer runs are discussed for application to this large problem encompassing approximately 122,000 binary integer variables and 1,000 constraints. Statistic analysis of the convergence process suggests that the evolution program may seek optimal solutions at reasonable computational cost. Results thus suggest that evolutionary techniques may be used to confront the complexity of integer forest management scheduling model solving. For. Sci. 47(2):158–168.

Key Words: Forest management, heuristics, genetic algorithms, combinatorial optimization, evolution programs.
second group includes heuristics (e.g., genetic algorithms), which seek to approximate an optimal solution at a reasonable computational cost. As Gunn and Rai (1987) pointed out, solutions that are near optimal and near feasible may be adequate and even preferable if they can be produced with a greatly reduced solution effort, given the uncertainty about biological, technical and economic data in most forest management problems.

Several authors used heuristic techniques to approximate optimal solutions to integer forest management scheduling problems. O’Hara et al. (1989), Clements et al. (1990) and Nelson and Brodie (1990) used a random search technique—Monte Carlo Integer Programming (MCIP)—for solving the forest management adjacency problem. MCIP requires large sample sizes when long planning horizons are considered (Murray and Church 1995a). Other techniques were developed to avoid this computational burden. Weintraub et al. (1994) developed a heuristic based on a column generation algorithm to address the adjacency problem. Weintraub et al. (1995) developed a heuristic to solve the transportation problem. Lockwood and Moore (1993), Dahlin and Sallnas (1993), Murray and Church (1995a), and Tarp and Helles (1997) used simulated annealing to solve spatially constrained forest management problems. Murray and Church (1995a), Bettinger et al. (1998), and Boston and Bettinger (1999) further used Tabu search. These techniques were reported to compare favorably to MCIP. Mullen and Buttler (1997) combined a genetic algorithm with a heuristic scheduling algorithm to solve forest management problems with adjacency constraints. Hoganson and Borges (1998) and Borges et al. (1999) developed a heuristic approach based on dynamic programming to solve the adjacency problem and reported that this technique compared favorably to some other solution approaches.

In this article, basic concepts of both genetic algorithms and evolution programs are presented. An evolution program (EP) to solve integer forest management scheduling models is developed based on extensive preliminary testing. An appropriate data structure, i.e., chromosome representation, is presented. Modifications of standard selection, crossover, and mutation strategies are designed for application to large forest problems. Emphasis is on designing an efficient EP that may approximate near-optimal solutions to integer programs and that may provide both strategic and operational guidance to forest managers. This approach extends substantially the number of binary decision variables that may be considered in the IP as compared to some combinatorial optimization heuristic techniques presented in the forestry literature. Contrary to some other approaches, it further enables consideration of the possibility of over one harvest in each stand over the planning horizon. A stopping criterion for this iterative heuristic is further presented.

After describing the EP design, this technique is applied, for illustration purposes, to a two-product timber harvest scheduling problem with a temporal horizon extending to seventy 1 yr periods, in Leiria National Forest (LNF), Portugal. Results from 50 test computer runs are discussed for application to this large problem encompassing approximately 122,000 binary integer variables and 1,000 constraints. It is reported that the statistic analysis of the convergence process suggests that the evolution program can produce near-optimal solutions at reasonable computational cost. Results thus suggest that evolutionary techniques may be used to confront the complexity of integer forest management scheduling model solving.

### Genetic Algorithms and Evolution Programs—Basic Concepts

A genetic algorithm (GA) is a directed random search technique developed by Holland (1975) to seek global optimal solutions in complex multidimensional search spaces (Pham and Karaboga 2000, p. 1). The techniques’ designation originated from the analogy between the representation of a solution by means of a vector of variables and the encoding of a phenotype by means of a sequence of genes in a chromosome (Reeves 1993, p. 152–153). Though the biological parallel is not exact, a sample of solutions in a search space is thus likened to a set of individual chromosomes in a population. The genes in a chromosome represent the variables in a solution vector, and the chromosome fitness value may consist of the value of the objective function for that solution. GA search methods emulate natural evolution by using genetics-inspired operators to manipulate the chromosomes over many generations so that their fitness is gradually improved (i.e., so that the solutions they represent approximate the global optimum). The initial GAs chromosome representation was domain independent and used only binary (0 or 1) genes, but due to failure in many areas (Michalewicz 1996, p. 5), different problem-related data structures were tried with success. Michalewicz (1996, p. 5–9) emphasized that recent experiments confirm the hypothesis that problem-specific knowledge enhances the performance of GA. According to this author, this knowledge may be translated into the selection of appropriate data structures—i.e., chromosome representations, other than traditional GA fixed length binary strings. It may further contribute to developing adequate selection, mutation, and crossover strategies.

Michalewicz (1996, p. 6–9) designated the combination of an adequate data structure with an adequate design of a GA to solve a specific problem as an Evolution Program (EP). EPs are then a more general term that includes GAs as well as other derived heuristics, sharing the same parallel evolutionary principles. Notwithstanding, this nomenclature is not shared by other authors (e.g., Reeves 1993, Pham and Karaboga 2000, in which the term “genetic algorithm” is used independently of the data structure used).

In GAs, each individual is generally comprised of only one chromosome, and both terms are used interchangeably. A GA encompasses the creation of an initial population of chromosomes, the definition of a chromosome fitness evaluation function, and the use of three common genetics-inspired operators: selection, mutation, and crossover (Reeves 1993, p. 153, Michalewicz 1996, p. 21, and Pham and Karaboga 2000, p. 3–7). The initialization process involves the creation of a population of chromosomes (Figure 1). The
value of each gene (allele) in each initial chromosome may be either generated randomly or defined according to a heuristic procedure (Reeves 1993, p.166). The resulting population is then evaluated by assessing the fitness of each chromosome (Figure 1). In optimization problems subject to constraints, the fitness evaluation function may be a mathematical equation, incorporating special penalty functions to assess the quality of solutions represented by chromosomes (Pham and Karaboga 2000, p. 21). It is then necessary to check if any of the individuals in the population satisfy the proposed goal or if it is pointless to try to go any further, by testing the current state of the population with an adequate criterion. If the process continues, each of the genetic operators is applied in turn. Selection aims at reproducing more copies of individuals whose fitness values are higher. Usually, chromosomes to be part of the new generation are selected randomly according to its relative fitness (Michalewicz 1996, p. 34–35). The two remaining operators act on the new generation, thus transforming it. This transformation marks the end of one GA iteration (Figure 1). Mutation consists of changing the values of genes (alleles) in individual chromosomes. It is a monadic operation. Generally, all chromosomes are checked gene by gene, and the corresponding alleles are randomly changed according to a specified rate (Pham and Karaboga 2000, p. 5). Crossover consists of swapping blocks of genes between two chromosomes, thus transforming both. Normally, the two chromosomes are selected randomly, and the blocks are cut at randomly chosen points (Reeves 1993, p. 153–154). Both crossover and mutation are regulated by control parameters that define both the probability of any of these operators being selected and the rate at which they will act on a new generation.

Reeves (1993, p. 189) summarized some attractions of the GA: domain independence, ability to solve nonlinear complex problems, robustness, and ease of modification. Michalewicz (1996, p. 16) further highlighted that, unlike other heuristics, GA may process simultaneously several points of the search space, thus providing information formation and exchange between search directions. Accordingly, there has been an increasing number of applications, and the literature describes a wide variety of different ways of implementing GA (Reeves 1993, Michalewicz 1996, Banzhaf and Reeves 1999, Pham and Karaboga 2000). Reeves (1993, p. 190) and Michalewicz (1996, p. 329) remarked that, in some cases, theoretical work has yet to be done. Nevertheless, both authors emphasize that GA experimental results are encouraging and that there is empirical evidence to support the preference for GA over other classes of algorithms to seek optimal solutions to some problems.

**Evolution Program Design for Solving Integer Forest Management Scheduling Models**

**The Integer Forest Management Scheduling Model**

If we follow a Model I formulation, as defined by Johnson and Scheurman (1977), an integer program that conveys the geographical location of forest activities may be described as:

\[
\text{Max NPV} = \sum_{i=1}^{N} \sum_{j=1}^{M} c_{ij} x_{ij} \tag{1}
\]

subject to

\[
\sum_{j=1}^{M} x_{ij} = 1, \forall i \tag{2}
\]

\[
\sum_{i=1}^{N} \sum_{j=1}^{M} v_{ij} \leq p_{ij} \geq (1 - d_{pi}) V_{p}, \quad p = 1, 2, ..., P, \quad t = 1, 2, ..., T \tag{3}
\]

\[
\sum_{i=1}^{N} \sum_{j=1}^{M} v_{ij} \leq (1 + d_{pi}) V_{p}, \quad p = 1, 2, ..., P, \quad t = 1, 2, ..., T \tag{4}
\]

\[
x_{ij} = 1 \lor x_{ij} = 0, \forall i, \forall j = 1, ..., M_i \tag{5}
\]

where

\[N = \text{the number of management units}\]

\[M_i = \text{the number of alternatives for management unit } i\]
\(P\) = the number of products

\(T\) = the number of planning periods

\(x_{ij}\) = binary variable that is set equal to 1 if alternative \(j\) is chosen for management unit \(i\) and to 0 otherwise.

\(c_{ij}\) = net present value associated with alternative \(j\) for management unit \(i\). It includes the value of the ending inventory

\(v_{ijpt}\) = yield of product \(p\) in period \(t\) that results from assigning alternative \(j\) to management unit \(i\)

\(d_{pt}\) = deviation allowed from target volume level of product \(p\) in period \(t\)

\(V_{pt}\) = target volume level of product \(p\) in period \(t\)

Equation (1) expresses the management objective of maximizing the forest net present value (NPV). Equation (2) ensures that one and only one alternative is assigned to each management unit. Equations (3) and (4) force the volume levels to meet the targets in each period. Equation (5) expresses the binary requirement on the decision variables. In general, this combinatorial optimization problem involves very large numbers of variables and constraints.

**The Design Approach**

The development of an evolution program to solve the integer forest management problem described in Equations (1) to (5) involved testing evaluation fitness functions, genetic operators, and other control parameters. Preliminary test problems encompassed several integer forest programs with the same type of constraints. Forest areas extended from 2,500 to approximately 29,000 ha. These areas were classified into up to 3,200 management units. The number of management alternatives ranged from 30,000 to 240,000 over planning horizons of up to 100 yr. The inventory in each forest management problem involved different stand types [e.g., pure maritime pine (\textit{Pinus pinaster} Ait.) high forest stands and pure eucalypt (\textit{Eucalyptus globulus} Labill) coppice stands]. The preliminary tests variability allowed for progressive refinements not only of the genetic operators but also (in a later phase) of the heuristic parameters. These refinements enhanced the evolution program performance both in terms of accuracy and of computational cost. As Reeves (1993, p. 189) pointed out, it is easy to modify a genetic algorithm to model variations in the original problem, thus generalizing its application.

**Data Structure—Chromosome Representation**

The data structure (representation scheme) used by the heuristic may affect the performance of the EP both in terms of accuracy and computation time (Michalewicz 1996, p. 56 and Pham and Karaboga 2000, p. 2). As Michalewicz (1996, p. 5) pointed out, binary representations may be inefficient when applied to multidimensional numerical problems. Other representations may facilitate the incorporation of problem-specific knowledge, thus contributing to higher heuristic performance. The preliminary tests suggested using a vector of integer parameters rather than a binary string. According to this representation, a chromosome consists of a sequence of genes each representing a management unit (Figure 2). The size of a chromosome is thus defined by the number of forest stands. The value of each gene (allele) identifies the management alternative assigned to the respective management unit (Figure 2). Each chromosome is thus associated with a specific solution to the integer program (Figure 2). The sequencing of genes (management units) in a chromosome is arbitrary, yet it must be the same for all individuals.

**Creation of an Initial Population**

A small number of starting points may under-cover the solution space, while a large number may incur in high computational costs (Reeves 1993, p. 165). The preliminary tests showed that populations with less than 20 individuals soon lose variability converging to local optimal solutions. On the other hand, populations with over 50 chromosomes were associated with slower convergence rates and did not provide better results. Testing suggested

![Figure 2. Data model of the chromosomes. C1 and C2 are chromosomes, with n genes [management units (MU)]. The value of each gene is an index to a specific management alternative. Each chromosome defines a management plan for the whole forest, in a 70 yr time horizon.](image-url)
considering 30 chromosomes representing 30 starting solutions to the forest integer program. The acquisition of knowledge about potential optimal solutions involved computational costs due to the program complexity. Therefore, the value of each gene (allele) in each initial chromosome was generated randomly. According to Pham and Karaboga (2000, p. 3), this method is also preferred for assessing the performance of the heuristic. Reeves (1993, p. 166) further asserted that the random assignment may decrease the chance of premature convergence to local optimal solutions.

**Fitness Evaluation Function**

After a new population is created (either from initialization or from a previous generation), each individual must be evaluated. In unconstrained optimization problems, the fitness evaluation function may correspond to the objective function itself. In this case, chromosome fitness assessment may be based on the respective objective function value. When the problem is constrained, alternative strategies do exist. The evolution program may use modified mutation and crossover operators so that only feasible solutions are produced. In this case, the fitness evaluation function might be the same as in the unconstrained case. Yet the convergence process tends to be slower, as there is a considerable computational burden. Thus the most common approach is the use of special functions that assign higher penalties to infeasibility. The general form of the fitness evaluation function becomes:

\[
Z = \sum_{i=1}^{N} \sum_{j=1}^{M_{i}} c_{ij} x_{ij} - \theta(W_{1t},..., W_{pt},..., W_{pT}, V_{1t},..., V_{pt},..., V_{pT})
\]

(6)

where

\[
\theta(W_{1t},..., W_{pt},..., W_{pT}, V_{1t},..., V_{pt},..., V_{pT}) = \sum_{p=1}^{P} \sum_{t=1}^{T} A_{pt} (W_{pt} - V_{pt})^2
\]

(7)

where

\[
P, T, V_{pt}, and W_{pt}\text{ as in Equation (6).}
\]

\[
A_{pt} = \text{positive parameter that characterizes the shape of the parabola for each product } p \text{ in each period } t \text{ if } |W_{pt} / V_{pt}| \leq d_{pt} \text{, or zero otherwise } [d_{pt} \text{ as in Equations (4) and (5)}].
\]

The values of \(A_{pt}\) affect the convergence process. Larger values emphasize the need to satisfy the constraints. Lower values give more weight to alleles (management alternatives) associated with higher net present values (NPV). Preliminary tests suggested that, in order to avoid scaling problems, user-defined relations between deviations from the target volume levels and unconstrained NPV reductions might contribute to higher convergence rates of the evolution program. The parameter \(A_{pt}\) was thus defined as

\[
A_{pt} = \Delta V_{pt} / (\Delta V_{pt})^2
\]

(8)

where

\[
\Delta V_{pt} = \text{unconstrained NPV reduction considered equivalent to a deviation from the target volume levels } \Delta V_{pt}
\]

For example, consider an integer forest management problem aiming at a volume level of 10,000 m\(^3\) for product \(p\) at period \(t\). Assume that if there are no volume flow constraints the NPV for this problem is 150,000 \(\times 10^3\) PTE. Assume further that a deviation of 5% from the target volume flow for product \(p\) in period \(t\) is considered by the user to be equivalent to a reduction of 1% of the unconstrained NPV. The shape of the penalty parabola would thus be determined by \(A_{pt} = (0.01 \times 150,000) / (0.05 \times 10,000)^2 = 0.006\). This approach contributes to make Equation (6) dimensionally compatible as both the objective function [Equation (1)] and the penalty function [Equation (7)] use the same commensurable units. The preliminary tests showed that optimal \(\Delta V_{pt}\) values ranged from 5 to 30% with \(\Delta NPV\) equal to 1%. As expected, larger \(\Delta V_{pt}\) values emphasize the NPV component in the objective function rather than the volume flow constraints, therefore slowing down the process of reaching a feasible search space. On the contrary, lower values (below 5%) underline the need to satisfy the constraints, but are more prone to drive the solution set to unfeasible local optima. All test problems were solved with the same range of penalty parameters.

**Selection**

Selection aims at reproducing more copies of individuals whose fitness values are higher. It is a critical step in the algorithm as it determines the areas to search thus impacting the convergence process. As Pham and Karaboga (2000, p. 3) pointed out, selection should further avoid the premature loss of diversity in the population in order to increase the probability of a global optimum being reached. Generally, chro-
mosomes to be part of the new generation are selected randomly according to its relative fitness (Michalewicz 1996 p. 34). The probability of selecting one individual to be part of the new population is:

$$P_i = \frac{F_i}{\sum_{j=1}^{C} F_j}$$  (9)

where

- $P_i$ = probability that a chromosome $i$ is selected to be part of the new population.
- $F_i$ = fitness of chromosome $i$, as evaluated by Equation (6).
- $C$ = number of chromosomes.

The preliminary tests showed that in some cases after several iterations most chromosomes in the population tended to be similar. Small differences in the fitness of the individuals increase the probability of selecting less apt chromosomes hence slowing down the convergence process considerably. Thus Equation (9) was modified to reduce more strongly the probability of selecting less fitted chromosomes for the new generation. It became:

$$P_i = (\frac{F_i}{\sum_{j=1}^{C} F_j})^S / (\sum_{k=1}^{C} (\frac{F_k}{\sum_{j=1}^{C} F_j})^S)$$  (10)

where

- $P_i$, $F_i$, and $C$ as in Equation (9)
- $S$ = parameter defining the intensity of selection

Equation (9) is a particular case of Equation (10) where $S = 1$. Preliminary tests showed that larger values of $S$ might accelerate greatly the convergence process. Yet it may also induce premature convergence to local optimaums as a consequence of lower population diversity. The random selection process may further lead to discarding very fit individuals. The tests suggested that the intensity of selection parameter value should be in the interval [2 4]. Higher values did preclude the reaching of feasibility in most test problems. Lower values led to a slowing down of up to 10 times of the convergence process. In order to avoid losing good solutions, this approach incorporated a “reposition of the best” strategy. The best individual in all generations so far is saved, and it substitutes the worst individual in a population with a user-defined regularity. This process proved to be particularly useful when operating with high mutation rates with chromosomes being easily and substantially changed. The test processes demonstrated that this is one of the most important parameters in the whole heuristic. A value of zero (no reposition) led to high chromosome dispersion across generations, thus missing, in most cases, the feasible region. Too small values (less than ten) at the initial stages reduced the variability of the initial populations, narrowing the search space and thus making more difficult the reaching of feasibility.

**Crossover**

Crossover recombines the individuals selected to be part of the new generation. Sections of genes are exchanged between the chromosomes to be transformed. The probability of crossover provides the expected number of chromosomes to recombine (Michalewicz 1996, p. 35). As Pham and Karaboga (2000, p. 7) referred, a low probability may slow down the convergence process in early iterations and too high a probability may lead to saturation around a solution. The selection of the starting gene in the sections to be exchanged is also performed randomly. The literature (e.g., Reeves 1993, Michalewicz 1996, Banzhaf and Reeves 1999, Pham and Karaboga 2000) discussed several crossover strategies (e.g., one-point, two-point, multipoint, and uniform crossovers). The preliminary tests suggested a constant probability of crossover around 0.1, although no substantial differences were found for values between 0.05 and 0.3. Given that the population consisted of 30 chromosomes, the default value chosen led to an average of one or two recombinations in each iteration. They further suggested an adaptation of former methods for exchanging genetic material between chromosomes. In order to implement it, the sequence of genes in each individual randomly selected to be recombined is organized in a set with a fixed number of blocks (B). A set with 20 blocks was used in all test cases (each block comprising 5% of the genes), which provided good results. The crossover procedure is then divided into two phases. First, the number of blocks to be exchanged is randomly determined. Second, the blocks to be swapped between individuals are selected randomly from the set B. The preliminary tests showed that the disposition of the genes in a chromosome is seldom random for it reflects the original sequencing of management units in the inventory database. This block swapping crossover strategy allows the exchange of noncontiguous genes yet maintaining the basic underlying crossover principles.

**Mutation**

Mutation is generally the last operator in a iteration. This operation performs unary transformations (transformations with one operand) upon the selected individuals as it consists of changing alleles in individual chromosomes. In general, the probability of using the mutation operator is fixed throughout all iterations (Reeves 1993, p. 174). Typically, all genes are checked, and the respective alleles are randomly changed according to a constant low (e.g., less than 1%) probability (Reeves 1993, p. 174, Michalewicz 1996, p. 21). This process has shortcomings. It represents a considerable computational burden even if library routines for initializing and generating random numbers are used. As Press et al. (1992, p. 276) pointed out, system-supplied rand () functions are almost always linear congruential generators (a common method for generating pseudorandom numbers). As a consequence, the sequences of random numbers are repeated recurrently, thus biasing this procedure. Moreover, it may be interesting to adapt the mutation rates to solve specific problems. Consider, for example, that the same mutation rate (e.g., 1%) is used in two problems with chromosomes with 100 and 1000 genes, respectively. On average, one management alternative assignment will be changed in the first problem, as
only one gene (management unit) will suffer a mutation. The preliminary tests suggested that this is a good strategy for converging to a global optimum. Yet, in the second problem, ten management alternative assignments will be changed as ten genes (management units) will be mutated, in average, in each iteration. The preliminary tests showed that this last value is too large as it contributes to high diversity, instability, and the slowing of the convergence process. As Pham and Karaboga (2000, p. 7) pointed out, too high mutation rates convert the evolution program into a random search approach. The tests further suggested that the number of mutations might be a better control parameter than the mutation rate.

Several alternative mutation schemes have been discussed in the literature. For example, Reeves (1993, p. 174) pointed out that, in order to decrease the probability of premature convergence to a local optimum, the mutation rate should be inversely proportional to the population diversity. The preliminary tests considered alternative schemes that might both reduce the number of genes’ random evaluations and improve the convergence process. The proposed mutation procedure defines the probability of having one or more mutations in a chromosome according to the value of a real number. The integer part indicates the minimum number of mutations to occur (the minimum number of changes in management alternative assignments) in each chromosome. The decimal part indicates the probability of accepting an additional mutation in the same chromosome. For example, the number 1.2 indicates that each chromosome will suffer at least one mutation and that the probability of a second mutation is equal to 0.2. The preliminary tests suggested that values below 2 are the most appropriate to solve the integer forest management scheduling model described here. They further suggested decreasing the mutation rate when chromosomes represent feasible solutions. Higher values (larger than 2) increase too much the population variability, therefore slowing down the convergence process.

**Stopping Criteria and Proximity to Estimated Optimums**

Assessing the quality of solutions provides information on whether to stop the iterative method or to let the convergence process continue to iterate seeking better solutions. The preliminary tests of this evolution program showed that as the number of iterations increases, the objective function value tends to behave asymptotically, the gains becoming smaller. Therefore, measuring the asymptote proximity might provide an effective stopping rule. In general, the parameters that define the shape of the convergence curve are hard to determine, thus precluding the possibility of using the relative distance to the theoretical asymptote. The approach presented in this article took into account a general property of asymptotic curves: as the curve reaches its horizontal asymptote, the derivatives of the functions that describe the curve tends to zero. The slope of a linear function adjusted to the asymptotic curves: as the curve reaches its horizontal asymptote. The approach preliminary tests demonstrated that the rule was effective, determining when it was useless to proceed with the iterative process.

Checking how close solutions are to the optimum is a critical issue when evaluating a heuristic performance. Reeves (1993, p. 308–313) discussed analytical methods, empirical testing, and statistical inference to assess how close heuristic solutions are to the optimum. The first methods may provide upper bounds to this integer program. If the integer requirement is relaxed, a linear programming solution may provide such an upper bound. In this article, assessing the quality of the evolution program solutions further used statistical inference. Reeves (1993, p.311) referred to the possibility of fitting a Weibull distribution to a sample of heuristic solutions in order to estimate the statistical optimum. Boston and Bettinger (1999) used the Anderson-Darling statistic to test the goodness-of-fit of the samples to a Weibull distribution. As the sample size was large in some of the preliminary test forests, the Kolmogorov-Smirnov (K-S) test was used here to test successfully the goodness-of-fit. In all applicable cases, the proximity of the best solution in the sample to the Weibull distribution location parameter (i.e., to the estimated heuristic optimal solution), suggested that the theoretical optimum obtainable with the heuristic was near and that the stopping rule was effective. Yet the K-S test is conservative, as values used to estimate the Weibull distribution parameters were the same used for the test.

**The Test Forest**

**Description**

Leiria National Forest (LNF) is a public forest located in the Portuguese central region by the Atlantic ocean. It extends over approximately 11,000 ha, of which 8,700 have been allocated to a timber production division. The remaining area is managed for conservation purposes (e.g., wind protection) and was not considered in this study. Even-aged high forest stands of Maritime pine (*Pinus pinaster* Ait.) occupy most of the area allocated for timber production. This forest division was classified into 696 homogeneous management units (MU) or stands according to ecological and productive criteria (Miragaia et al. 2000). Stand area ranges from 1 to 33 ha. The current distribution of stand area by age class is very irregular with over 50% of the area with an age between 10 and 34 yr (Figure 3). The site index ranges from 12 to 24 m (base age of 50 yr). A typical prescription at LNF encompasses natural regeneration and/or plantation leaving

![Figure 3. Age distribution at the Leiria National Forest.](image-url)
about 2,000 seedlings per ha and a noncommercial thinning at 15 yr of age that leaves about 1,500 trees per ha. In the period from 15 to 50 yr of age, pine stands are thinned every 5 yr. Clearcutting may occur in the period from 50 to 90 yr of age, when there are about 200–250 trees/ha (Falcão 1999). The optimal rotation is about 70 yr. The forest supplies two timber products. Logs with larger diameters resulting from clearcuts are a very valuable asset in LNF. Yet there is also demand for material with smaller diameters, resulting mostly from thinnings and from younger rotations.

The prescription writer SAGIFLOR (Falcão et al. 1999, Borges and Falcão 1999) was used to simulate all possible management alternatives for all 696 stands, over a planning horizon of seventy 1 yr periods. Potential clearcut ages ranged from 50 to 90 yr, with a 2 yr interval. The possibility of 3 thinnings regimes (leaving as residual basal area after thinning 18, 20 or 22 m²/ha) was considered. Potential thinning ages ranged from 20 to 50 yr, with a 5 yr interval. The DUNAS model (Falcão 1999) is integrated in the prescription writer and was used for growth and yield projections. The prescription writer assigned over 400 management alternatives to some stands. Yet, on average, it generated about 175 prescriptions to each management unit. Thus, the problem had about 121,800 binary variables. Its cost coefficients (NPVs) in Equation (1) were computed based on economic data collected by the LNF administration. A 5% discount rate was used. Demand levels over the whole planning horizon were assumed to be constant for both products. Target volume flows and acceptable deviations were based on historical volume levels, for each product in each period, to be balanced in the near future. In the case of Product A, acceptable deviations decreased linearly from 10 to 5% from the target flows in the first and the last years of the planning horizon, respectively. The latter decreased with time, as it was assumed that the inventory would be more balanced in the near future. In the case of Product B, acceptable deviations decreased linearly from 10% to 5% from the target flows in the first and the last years of the planning horizon, respectively. In the case of Product B, acceptable deviations decreased linearly from 30% to 15% from the target flows in the first and the last years of the planning horizon, respectively. The number of area and volume flow constraints thus approximated 1,000. These intervals defining the acceptable deviations from the target volume flows for each product affected the evolution process, as solutions with production levels between these two limits were considered feasible and thus not penalized.

**Results**

The test problem was solved with the same machine used for the evolution program design (Pentium II processor operating at 400 MHz and running Ms Windows NT). The whole heuristic was implemented in a user-friendly interface built with Ms Visual Basic 6.0. The evolution program parameters were assigned values according to the preliminary testing (Table 1). Reaching feasibility is a critical step in the convergence process. The loss of population diversity may induce a premature convergence to an infeasible local optimum that is hard to overcome. Yet the evolution program designed circumvented this potential limitation by slowing down the convergence process. For that purpose, until feasibility was reached, the mutations control parameter was higher, the intensity of selection was lower, and the strategy of “reposition of the best” occurred with a lower periodicity (Table 1). Furthermore, the convergence process was designed to restart with a new population if feasibility was not reached before a threshold value of 200,000 iterations.

The procedure used to assign the penalty values assumed as a standard the deviation of 1% from the unconstrained optimum net present value [\(\Delta\text{NPV}\) in Equation (8)]. It further set the equivalent deviations from the target volume levels \(\Delta V_{pt}\) to be user-defined parameters. Therefore, the user just had to select the deviation from the target volume levels, for each product in each period, to be considered equivalent to a deviation of 1% from the unconstrained optimum NPV. In this problem, it was found that the former might be considered constant over the planning horizon for each product, as the target levels did not fluctuate substantially in time. The unconstrained optimum NPV was computed using CPLEX 6.5 (ILOG 1997). It reached about 8,244 x 10⁶ PTE (Table 2).

The evolution program was executed 50 times. The average computational performance of the evolution program reached about 300 iterations per second. The number of iterations needed to reach feasibility ranged from 25,000 to 120,000, the average being 40,000. After feasibility was reached, the number of iterations before meeting the stopping criteria approximated 400,000. Thus the solution process took, on average, less than 25 minutes. The solutions were found to be independent of the number of iterations. The asymptotic behavior of the convergence process (Figure 4) confirmed the results of the preliminary tests. Therefore, a linear function was adjusted continuously to the best solutions provided by the last 100 iterations and the convergence process was stopped when its slope was less or equal to 0.01.

To test for the accuracy of the stopping criterion, a 2,000,000 iteration sample run was used (about 2 hr of processing time). The final solution value showed an increase of 0.15% relative to the point where the stopping criterion was reached (about 650,000 iterations). This value was considered negligible.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Until feasibility</th>
<th>After feasibility</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intensity of selection</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>Mutation control parameter</td>
<td>1.5</td>
<td>0.5</td>
</tr>
<tr>
<td>Crossover probability</td>
<td>10%</td>
<td>10%</td>
</tr>
<tr>
<td>Reposition of the best</td>
<td>50 iterations</td>
<td>5 iterations</td>
</tr>
</tbody>
</table>

**Table 1. Parameters used for the convergence process.**

<table>
<thead>
<tr>
<th>Result</th>
<th>Value (x 10⁹ PTE)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear programming solution</td>
<td>8.2444</td>
</tr>
<tr>
<td>Weibull distribution location parameter (statistical optimum)</td>
<td>7.9158</td>
</tr>
<tr>
<td>Maximum solution found</td>
<td>7.9027</td>
</tr>
<tr>
<td>Minimum solution</td>
<td>7.8154</td>
</tr>
<tr>
<td>Average solution</td>
<td>7.8582</td>
</tr>
</tbody>
</table>

**Table 2. Summary of results from 50 runs (1USD = 220PTE in April 2000).**
when compared to the much larger computational burden required. According to the stopping criterion used in this heuristic approach, it is possible for the user to decrease the slope value that stops the computer run in order to seek better solution values at an additional computational cost.

A 3-parameter Weibull distribution was fitted to the 50-solutions sample produced by the evolution program (Figure 5) using STATISTICA (StatSoft 1997). The Kolmogorov-Smirnov test showed that the data adjustment to the fitted distribution was accurate with a confidence interval of 90%. The value of the best solution found was within 0.2% of the estimated heuristic optimum provided by the Weibull distribution location parameter (Figure 5). In average, solution values were within 0.9% of the value of the location parameter, and all 50 results were within 1.3% of the estimated optimum. Furthermore, all solution values were within a range of 1.0%, thus suggesting that the heuristic is robust. Results thus confirmed the validity of the stopping rule designed in the framework of the preliminary tests. The value of the optimal solution of the linear programming model with no integer requirements was 4.1% higher than the estimated optimum. (Table 2). Yet this solution fragmented 103 management units (about 15% of the total number of units), thus losing required locational specificity.

**Summary and Conclusions**

In this article, an evolution program was designed and proposed to solve integer forest management scheduling models. The program design was based on preliminary tests encompassing a wide range of integer forest programs. Forest areas extended from 2,500 to approximately 29,000 ha. These areas were classified into up to 3,200 management units. The number of management alternatives ranged from 30,000 to 240,000 over planning horizons of up to 100 yr. The inventory in each forest management problem involved different stand types common in Portugal [e.g., pure maritime pine (*Pinus pinaster* Ait.) high forest stands and pure eucalypt (*Eucalyptus globulus* Labill) coppice stands]. These tests contributed to fine-tuning an evolution program by evolving an adequate data structure, an initialization process, and the definition of a chromosome fitness evaluation function and of a selection procedure. They further suggested specific selection, mutation and crossover strategies and a stopping criterion for the convergence process.

This heuristic was applied to a large forest management scheduling problem in Portugal. This test problem was complex as it encompassed an area of about 8,700 ha classified into 696 stands, a planning horizon extending over 70 1 yr periods and target annual volume flows for two products. An average of 175 management alternatives was available for each stand. Model building thus involved the definition of over 121,000 binary variables and about 1,000 constraints. Experimental results from 50 evolution program test runs suggest that the proposed heuristic is both efficient and robust. Coding improvements may lead to better computational performances. Nevertheless, the convergence process took in average less than 25 minutes with a Pentium II processor operating at 400 MHz and running Ms Windows NT. Furthermore, the best solution was within 0.2% of the estimated heuristic optimum provided by the Weibull distribution location parameter. Solutions were further independent of the randomly determined starting point. The convergence process in all 50 cases led to solutions within a range of 1.1%, thus suggesting the robustness of the heuristic.

The forest management scheduling problem addressed here encompassed two-product target volume flows and locational specificity requirements. The evolution program designed conveys the geographical location of forest activities, thus contributing to integrate strategic and operational planning. This approach extends substantially

**Figure 4.** Four sample runs showing the asymptotic behaviour of the convergence process (1USD = 220PTE in April 2000).

**Figure 5.** Empirical distribution (Emp Dist) obtained from the solution values and the adjusted Weibull function.
the number of binary decision variables that may be considered in the framework of an integer program as compared to some combinatorial optimization heuristic techniques presented in the forestry literature. It further enables considering the possibility of over one harvest in each stand over the planning horizon, contrary to some other approaches. This management problem was deliberately over-constrained (e.g., requiring locational specificity of harvest to occur in later periods), thus complex enough to test whether the approach developed might confront very large forest integer problems.

Iteration time in the EP designed has a processing time that is independent of the dimension of the problem, since each new solution is generated randomly from the set of all possible combinations, even though the time needed to meet the stopping criterion may increase linearly with the number of variables in the model. Yet, problems still exist, not the least being the occasional difficulty in attaining feasibility. Several mechanisms can improve the heuristic, conducting the evolving population more effectively towards the bounding region. These are still subject to research, but could include variable mutation rates and crossover probability, or the use of dynamic penalty functions, with parameters self-modified with the convergence process.

EPs are robust algorithms in many ways different from other combinatorial optimization techniques, as the testing of multiple solutions simultaneously establishes complex relations among them, allowing easier escape from local optima. On the downside, EPs will probably perform slower than other approaches; for albeit EPs can carry out a more exhaustive search of the solution space, they do so at a proportional computational cost, mostly due to the intrinsic parallel exploration method. Several code optimizations were made, yet there is still much room for performance improvements. Focus should be mainly on optimizing the internal data storage procedures and on coding the critical evaluation functions in lower level languages. These points could be important factors in optimizing the performance of the current implementation.

Research on the discipline of combinatorial optimization may provide information needed to solve some forest management scheduling problems (e.g., problems that encompass transportation considerations and/or concerns with the spatial arrangement of harvests). The proposed evolution program can solve large-scale problems incorporating multiproduct flow constraints and locational specificity requirements. The literature on combinatorial optimization (e.g., Reeves 1993, Michalewicz 1996) further emphasizes that evolution programming facilitates the modeling of additional constraints to the original problem, contrary to other heuristic approaches. Future research will address the potential of adapting the evolution program to solve integer forest management scheduling models that incorporate additional spatial constraints (e.g., adjacency, edge length, and interior space), provided that topological information for all the management units is available. Using the current penalty functions, it should not be too hard to include spatial products in the formulation, penalizing solutions that do not satisfy the constraints.

### Literature Cited


Borges, J.G., H.M. Hogansson, and D. Rose. 1999 Combining a decomposition strategy with dynamic programming to solve the spatially constrained forest management scheduling problem. For. Sci. 45:201–212.


