# Constraint Technology Applied to Forest Treatment Scheduling

Junas Adhikary[*] Geir Hasle[†] Gunnar Misund[‡]

**Abstract**

Forest treatment planning and scheduling is an important part of forest resource management. It is a complex task requiring expertise and integration of multi-disciplinary fields. In this paper, we outline a real life problem from the ECOPLAN project called the Long Term Forest Treatment Scheduling Problem (LTFTSP). A review of optimization techniques applicable to forest treatment problems in general is presented, and contrasted with our case. The review suggests that long term scheduling is difficult because of the prohibitive size and complexity inherent to the problem. Based on experience from the successful resolution of a simplified problem, we advocate the use of iterative improvement techniques as a solution strategy. Iterative improvement techniques will in general benefit from high quality initial solutions. We show how a Constraint Satisfaction Problem formulation of the LTFTSP can be used to generate initial solutions. A key element to success is the use of a forest simulator for knowledge based definition of variable domains. The initial solution generator will be used as a module in an integrated forest treatment scheduling system which is under development in the ECOPLAN project.

[*]Intelligent Systems Lab, Simon Fraser University, Burnaby, B.C., Canada V5A 1S6, adhikary@cs.sfu.ca, http://www.cs.sfu.ca/cs/research/groups/ISL/

[†]SINTEF Applied Mathematics, Departmernt of Optimisation, Oslo, Norway, gha@math.sintef.no, http://www.oslo.sintef.no/am/

[‡]SINTEF Applied Mathematics, Department of Spatial Modelling, Oslo, Norway, gmi@math.sintef.no, http://www.oslo.sintef.no/am/

## 1 Introduction

The amount of forest area that is managed worldwide is huge. United States Department of Agriculture manages a forest area almost twice the size of Germany. Forest treatment planning and scheduling is an important part of forest resource management. Both official bodies and the public worldwide are demanding sustainable forest harvesting practice that not only deals with economics, but also with preservation of bio-diversity, esthetic values, public recreation areas etc. Long term forest treatment scheduling allows all parties to see if sustainable forestry is actually being practiced. Simplified, the problem is to assign forest treatment actions to treatment units (often called *stands*) over time, in a given forest area. The scheduling horizon is long, covering up to several centuries. An optimization version of the problem is to optimize certain economical and ecological objectives subject to various constraints.

Traditionally, mathematical programming has been extensively used to generate such schedules, and is still the most widely used technique [4, 28]. Artificial Intelligence (AI) techniques such as rule-based and local search methods, have also been applied to forest management [19, 3, 26]. Such techniques have difficulty in modelling the wide variety of spatial, temporal, and visual constraints, and optimizing upon several, usually conflicting criteria that pertain to adequate formulations of the problem. In addition, the typical size of real life LTFTSPs is often beyond the practical capacity of these methods.

Misund *et al* [24] have suggested integrated use of Geographic Information Technology (GIT) and AI techniques (constraint reasoning and local search) to model and solve such problems. They describe their approach applied to a simpler version of the LTFTSP, called the Clear-Cut Scheduling Problem (CCSP), where clear-cutting is the only type of treatment action allowed. They model it as a Constraint Satisfaction Problem (CSP) [34], and the solution is iteratively improved using Tabu search [5]. The study shows that this method can give "good" solutions for CCSP in reasonable time. As an extension of this work, SINTEF Applied Mathematics, in close collaboration with the Norwegian forest owner organizations and the Norwegian Agricultural University, is developing an integrated forest treatment scheduling system called ECOPLAN. Our paper is based on problems and results from the ECOPLAN project.

The paper is organized as follows. As a general backdrop, we briefly outline the ECOPLAN variant of the forest treatment scheduling, called the Long Term Forest Treatment Scheduling Problem (LTFTSP). Next we present a literature review of optimization techniques used in forest treatment scheduling in general. Then we take a closer look at the ECOPLAN LTFTSP, in particular the process of generating initial solutions for the iterative improvement module. We introduce a CSP model for the problem of generating a complete, good quality initial solution, and suggest how constraint reasoning techniques can be applied to it. We then report the design of an algorithm based on a `min-conflicts` heuristic [23], and describe the results from empirical investigation on real-life ECOPLAN data. We conclude with some remarks and suggest a few directions for future research.

## 2 The ECOPLAN Problem

Forest treatment scheduling problems come in variety of flavors and sizes, depending on the purpose of the harvesting, local criteria and constraints, governmental regulations, biological regimes, climate, etc.

To gain the necessary focus, we base our work on a specific problem identified and formalized in the ECOPLAN project. The problem will be referred to as the Long Term Forest Treatment Scheduling Problem (LTFTSP), or, plainly as the ECOPLAN problem.

Sustainable forest management requires that forest treatment actions scheduled for a long period of time must obey a variety of constraints. For management purposes, a forest landscape is divided into basic treatment units, often equivalent to *stands*, which is the forestry term for a forested area considered homogeneous with respect to a selected set of properties.

We will restrict our study to even-aged stands, i.e., stands containing trees of same or almost same age. We will call the time period of the schedule the *scheduling horizon*. It is common practice to divide the scheduling horizon into a number of time periods, each period equivalent to, say, 5 – 10 years.

**Definition 1 (Forest, Stand)** *A stand $S_i$ is a an area which is considered homogeneous with respect to certain properties. The set of $n$ stands comprises a* forest $\mathcal{F}$*, such that:*

1. $\mathcal{F} = \bigcup_{i=1}^{n} S_i$
2. $S_i \bigcap S_j = \emptyset$, *for all* $i, j, \quad i \neq j$

*For each stand $S_i$ we have certain stand specific information*[1]*:*

- *time of most recent treatment*
- *minimum duration between treatments*
- *maximum duration between treatments*
- *optimal time between treatments*
- *time required by trees to grow from 0 to a certain threshold height*
- *area of the stand*

[1]Parts of this information is only available through a stand simulator system.

- *volume that may be harvested after a specified number of years after the last harvest*

**Definition 2 (Treatment)** *A treatment unit (stand) is given a number of management options over the scheduling horizon. We will call an individual management option a* treatment*. A set of treatments $\mathcal{T} = \{T_0, \ldots, T_t\}$ is available for each stand. We define $T_0$ to be the* null treatment *or* "let grow"*, i.e., the stand is left without any treatment. We define $T_1$ to be the clear-cut treatment.*

**Definition 3 (Scheduling Horizon)** *A scheduling horizon $\mathcal{H}$ is a contiguous set of $p$ periods $\{P_1, \ldots, P_p\}$, where each $P_i$ is of a certain length in years.*

**Definition 4 (Long Term Forest Treatment Scheduling Problem)** *Given a forest $\mathcal{F}$ with $n$ stands, $\{S_1 \ldots S_n\}$, find a schedule $\mathcal{S} = \{T_1, \ldots, T_p\}$, $T_i \in \mathcal{T}$, such that for each period in the horizon a corresponding treatment assignment is made. Furthermore, the schedule has to satisfy a set of constraints $\mathcal{C} = \{C_1, \ldots, C_c\}$.*

The Clear-Cut Scheduling Problem (CCSP), as defined and treated in [24], is a special case of LTFTSP. Only one treatment type, clear-cutting, is allowed. Each stand is assigned this treatment only once, i.e., the remaining periods are assigned the trivial treatment. In Figure 1, an example of a forest area divided into stands is shown. Two stands sharing a common border are defined as *adjacent*, or *neighbors*.

**Definition 5 (Clear-Cut Scheduling Problem (CCSP))** *The CCSP is a restricted LTFTSP. Only two treatment types are allowed, $T_0$ and $T_1$ ("let grow" and clear-cut). Clear-cutting is scheduled only once for each stand during the entire scheduling horizon, and period length is* 1 *year.*

In the following, we outline the constraints and objective criteria associated with the ECOPLAN problem.

**Constraints** Constraints can generally be divided into two categories - hard and soft constraints. Hard constraints are defined as those constraints that must be satisfied, whereas soft constraints can be relaxed to satisfy hard constraints or to adjust the objective function value.

**Hard Constraint: X-m constraint** All the neighbors of the stand to be harvested must have an average tree height of at least X meters.

**Soft Constraint: Harvest Time** Bounding times between each harvest, based on forestry knowledge and economical and ecological considerations. A lower and upper threshold is given for each stand, as well as the optimal harvest time relative to the last clear-cut.

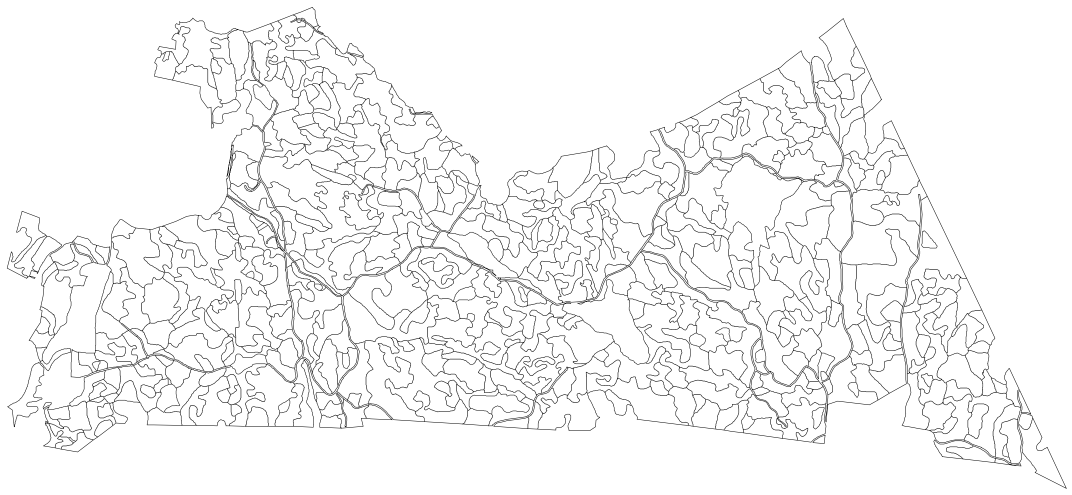

Figure 1: An example of a forest area divided into stands.

**Optimization Criteria** An optimization version of the CCSP arises when various criteria are to be maximized (or minimized) in the solution. Similarly, there are optimization versions of the LTFTSP.

**Optimal Harvest Time** The actual time between harvests should be as close to optimal time as possible.

**Even Consumption** Estimated harvested volume for any period should be as close to a specified consumption volume. In other words, the variance of the harvested volumes should be minimized. In a more refined model, the deviation from a predefined harvesting profile should be minimized.

**Old Forest** A specified minimum area of old forest, i.e., stands with average age above a certain threshold, should be maintained over the schedule horizon.

**Visual Impact** One of the objectives is to minimize the visual impact of clear cutting relative to a set of viewpoints.

For more detailed specifications of the ECOPLAN problem, confer [24].

The X-m constraint is enforced so that large areas are not clear-cut simultaneously causing damage to the regeneration process and wildlife habitats. It is also referred to as the *adjacency constraint* because cutting adjacent stands at the same time is prohibited. The restriction is usually valid for a certain number of specified periods such that the harvested area will have regenerated properly. These periods are commonly referred to as *exclusion periods*. Recently, adjacency constraints and exclusion periods have been given special consideration in most research dealing with forest harvest scheduling. This will also be our main focus when generating initial solutions of the ECOPLAN problem.

## 3 Review on Optimized Forest Harvest Scheduling

There are two major classes of forest harvest scheduling algorithms. One is based on mathematical programming and the other on heuristic techniques (with or without using simulation in both cases). The former is a global procedure attempting to output an optimal solution of the forest management model. The latter is usually a local procedure which iteratively optimizes the model without any guarantee of finding an optimal solution. Mathematical Programming is the technique that is commonly used in practice and consequently a large percentage of research activity is devoted to it. There are some algorithms which do not fall under these two categories which we will mention under miscellaneous techniques.

### 3.1 Mathematical Programming

The general forest planning problem, where harvesting is an important process, has been studied for some time. Early forest management models (Navon 71 [27], Johnson and Crim 86 [10]) were developed using linear programming (LP). Well known LP-based scheduling packages in use are FORPLAN (Johnson and Rose 86 [12]) and MUSYC (Johnson and Jones 79 [11]). Johnson and Scheurman 77 [13] reviewed and analyzed many LP-based forest planning systems. In the paper, the authors group LP models into Model I and Model II, which is frequently used in forest planning. Later, Garcia in his excellent review of LP in forest planning (Garcia 90 [4]) revises the classification. All these models are geared towards LP-based solution strategies.

Since LP models use continuous variables, the solution is non-integral. Largely because of this, spatial relationships are not defined in the model. Consequently, LP-based solutions are not readily acceptable because they are difficult to interpret and may be impossible to implement.

Since LP-based models were not able to express spatial relationship, researchers began to study and apply mixed integer programming (MIP) models (Kirby *et al* 86 [15], Jones *et al* 91 [14]). An integer decision variable is used to express a particular harvesting decision. This allows the model to express spatial relationships in the form of adjacency constraints. Since the MIP model uses a large number of integer variables, it is restricted to small-sized problems.

One technique to keep the MIP model to manageable size is the use of efficient representation of adjacency constraints (Meneghin*et al* 88 [22], Torres-Roho and Brodie 90 [33], Jones *et al* 91 [14], Yoshimoto and Brodie 94 [37]). Researchers have been steadily solving larger and larger MIP problems using improved constraint formulation and various heuristic algorithms. Weintraub *et al* 94 [36] solved a MIP forest harvesting model with adjacency constraints for multiple time periods using column generation technique, linear programming, and cut constraints as heuristic. A LP relaxation of the MIP model is solved and passed on to the MIP solver which attempts to assign integer values using heuristic algorithms.

The LP model is also sometimes used together with simulation models. Growth and yield simulations are used to find appropriate treatment schedule for individual stands. In most cases, each stand is treated independently and the necessary information is provided as the simulation proceeds unlike in LP models where all the information has to be encoded beforehand in the model. Some researchers have tried to combine these two techniques. Hoen's GAYA-LP (1992) [7] and Lappi's JLP (1992) [18] are good examples. In both cases, growth and treatment simulators are used to define allowable treatments to each stand. Then the output is fed to an LP solver which optimizes the net present value of the forest as a whole in every period. Net present value is calculated using input economic and forest data.

## 3.2 Heuristic Optimization

There has been several studies exploring the use of heuristic optimization techniques with or without mathematical programming. One of the approaches that is regarded as successful is the sampling heuristic called Monte Carlo Integer Programming (MCIP) (Nelson and Brodie 90 [28]). It is a biased sampling scheme that generates feasible solution alternatives. The more number of samples, the better the solution. Therefore, optimal or near optimal solutions may only be possible if very large number of samples are generated. Unfortunately, large samples significantly increase the time taken to find a solution.

Lockwood and Moore (1993) [20] use simulated annealing to generate harvest schedules with spatial constraints. Simulated annealing (SA) is a stochastic optimization technique that has been successfully used to solve combinatorial optimization problems (Kirkpatrick 83,84 [16, 17]). The authors report to have solved a large harvest scheduling problem with adjacency constraints.

Kangas and Pukkala (1993) [29] present another heuristic optimization technique. Their method uses a growth simulator as the first step to produce several alternative treatment schedules for each stand. The second step is the actual heuristic optimization where optimal schedules are sought by maximizing the total utility, where utility is calculated by adding the values of the objective function. This method was tested on a data of a small forest area and was found to be successful. It is claimed to be better than LP-based methods because of its ability to express nonlinear objectives. The drawback of this method is that it does not take care of adjacency constraints and therefore the schedules generated may be of poor quality.

A recent study compares three heuristic solution approaches to forest planning problems, harvest scheduling being a part of it (Murray and Church 95 [26]). The authors model the problem as a MIP which allows the representation of adjacency constraints. The only objective used is maximization of net revenue. They develop a method which improves upon the solution produced by Monte Carlo sampling process using Artificial Intelligence heuristic methods such as hill climbing (HC), simulated annealing (SA), and Tabu search (TS). In all three approaches, an initial solution produced by Monte Carlo sampling is locally improved by generating new neighborhood solutions. In HC, only improved solutions are accepted in each step and therefore it is more likely to get stuck in local optima. SA and TS accept worse solutions (with some probability) in hope of escaping local optima. These methods were tested using data from Nelson and Brodie 90 [28]. It was found that TS performed better overall than than SA or HC. However, this does not mean that TS will always produce better solutions than the other two. The authors confirmed this using Friedman analysis (Coonover 1980 [2]) on the solution results. This means that given any initial solution, it is equally likely that a high quality solution is reached by any one of the three processes.

Almost all of the above methods have an underlying mathematical programming model for forest harvesting problem. One interesting study is Misund et al 95 [24], where the problem is modelled as a constraint satisfaction problem (CSP) for the first time (to our knowledge). The authors used a CSP model and Tabu search as iterative improvement technique to solve the CCSP, a restricted version of the LTFTSP.

## 3.3 Miscellaneous

There are a few other studies that do not fall under above categories. One study uses 0-1 integer programming to determine patterns for forest harvesting with adjacency restrictions and forbidden regions modelled as a grid-packing problem (Snyder and ReVelle 95 [31]). Another recent study suggests a method based on Bayesian statistical concepts (Van Deusen 96 [35]) to schedule a large number of stands over a long period of time. This method also allows adjacency constraint to be satisfied. However, the method only uses economic criteria as objectives.

Other techniques worth mentioning here are those based on control theory, non-linear programming, and dynamic programming (Roise 86 [30, 8]). Most of this research also deals with the problem of habitat scheduling along with harvest scheduling. However, the size of problems actually solved is very small because of the enormous size of the model.

## 3.4 Summary

Most of the techniques and methods mentioned in this paper have been tested with very small problems compared to the practical problems that exist. One reason is the use of MIP models which tend to combinatorially explode in size with the number of stands involved. However, some studies have devised and used various heuristics solving larger and larger problems. Furthermore, in most experiments, the scheduling horizon is kept very small to keep the number of constraints under control. However, these methods do not deal with various other constraints and objectives simultaneously, such as the ones we have illustrated in Section 2.

It would be intuitive to use recent advancements in simulation modelling, both growth and yield, and treatment simulation, and geographic information technology to generate and visualize forest harvesting schedules. Many times it is helpful to visualize the solution to see if any improvement can be made, for example in the visual effect of a schedule. It is therefore imperative that models are designed such that these technologies can also be easily incorporated.

## 4 Treatment Scheduling as a CSP Model

The standard Constraint Satisfaction Problem ($CSP$) can be represented as a 3-tuple $(X, D, C)$ where $X = \{x_1, \ldots, x_n\}$ is a set of variables, $D = \{D_1, \ldots, D_n\}$ is a set of associated domains and $C$ is a set such that each member $C_{ij} \subseteq D_i \times D_j$ specifies the consistent values for variables $x_i$ and $x_j$. We can formulate the LTFTSP as a CSP if we consider the adjacency ($X - meter$) constraint. The nodes and edges in the *constraint graph* represent the stands $\{S_1, \ldots, S_n\}$ and the adjacency constraint, respectively. The domain of a node is a combination of treatment types and periods in the scheduling horizon (see Figure 2).

In this formulation, the problem is to assign the nodes as a particular sequence of treatment types and corresponding periods such that the adjacency constraints are satisfied.

It should also be mentioned that we are interested in an optimized solution, not just any feasible solution. In general, the standard CSP definition lacks power to express important aspects of real-life problems such as soft constraints, constraint and tuple priorities, constraint relaxation, and optimization criteria. Recently, the CSP community has turned to richer, non-standard, CSP formulations and corresponding resolution methods in order to address real-life problems more adequately. Examples are the max-CSP, the Maximum Utility Problem, the semiring-based CSP, and the hierarchical CSP formulation [9]. The standard CSP formulation is often adequate for subproblems of real-life problems, e.g., the problem of finding an X-m constraint feasible solution for the LTFTSP.

There are well-studied AI search methods that can be used to solve CSPs [34].

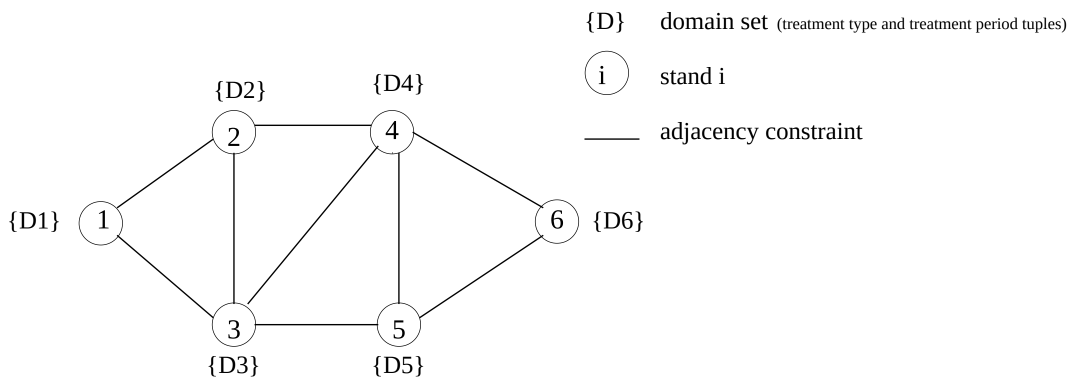

Figure 2: Part of constraint graph associated with forest in Figure 1.

These methods can be classified into constructive and iterative search methods. Constructive search, e.g. standard backtracking [1], starts with a particular ordering of the variables and instantiates them one at a time. Thus it works with a partial solution and tries to extend it to a full solution. This method of search suffers from a phenomenon called *thrashing* whereby the same variable-value pair that leads to no solution is instantiated over and over again. The algorithm has exponential time complexity. The benefit of this search technique is that it is complete, i.e., all possible solutions will be found, and the optimal solution can be identified if optimization criteria are given.

The alternative to constructive search is to start with an initial solution and use *local search* based on operators that perform small modifications to a given solution, in tandem with a *meta-heuristic* to avoid local optima. Examples of meta-heuristics are Tabu search [5] and Simulated Annealing [17]. These socalled Iterative Improvement Techniques (IITs) have proven to be effective for a variety of large and complex search problems. IITs are reasonably fast, and, the best solution so far is available virtually at any time. Fairly large problems can be "solved" from a pragmatic point of view, i.e., the method will be able to return a high quality solution even under strong response requirements. In general, very little can be guaranteed in terms of performance, such as the distance to optimum as a function of the number of iterations. However, this may be due to the inherent complexity of the problem at hand.

## 5 Construction of Initial Solutions

Based upon our research [24], including the previous review of existing methods, we believe that large size LTFTSPs are best handled by IITs, using a meta heuristic selected from the variety of heuristics available [34]. In particular, we performed initial investigations on a 500 stand CCSP using a LP formulation, a CSP formulation using backtracking search with arc consistency techniques, and IIT. IIT clearly outperformed the competition in these studies.

Folklore says that, for IITs to work well, the initial solution has to be reasonably "good". The remainder of this paper focuses on the important subtask of generating a "good quality" initial solution for the LTFTSP. We present a method for generating an initial solution that uses a simple `min-conflicts` heuristic [23] to remove adjacency conflicts.

The method also uses stand simulation to generate the search space for the problem. For real-life cases, the search space is huge. Given our CSP encoding, a scheduling horizon of 200 years, 10 different treatment types the theoretical domain size of each variable will be $10^{200}$. Just a tiny fraction of the domains will consist of local treatment schedules that are realistic from a forestry perspective. With a typical problem size of $5,000$ variables, there is clearly a need to apply forestry knowledge to drastically reduce domain size and focus search towards a high quality initial solution. Our strategy is to let a stand simulator, which is a repository of forestry knowledge, create a reasonably sized domain. Each domain consists of a set of local schedules, i.e., a treatment schedule for a given stand. The local schedules in the domain will thus be consistent and sound according to forestry practice. The task of finding a good quality initial solution to the LTFTSP is hence reduced to finding a near-feasible and optimized combination of local schedules over the total forest area. For large problem instances, this subproblem is still formidable.

## 5.1 Stand Simulation and CSP Domain Values

In the ECOPLAN project, one of the stand simulators to be used is GAYA [7]. GAYA generates a number of alternative local schedules for each stand. Depending upon the scheduling horizon used, the number of schedules can be as large as 1000 per stand [32]. In our CSP encoding, each of these local schedules becomes a value in the domain of the corresponding variable (cf. definition of CSP in Section 4). That is, only the schedules from the stand simulator generated domain can be chosen as a value for the variable. Hence, stand simulation is used to reduce the search space for our problem. The total search space for the LTFTSP increases exponentially with the selected domain size.

One simple algorithm to assign a domain value (a schedule) to a variable (a stand) is as follows:

**Algorithm 1**

1. start instantiating the variables in some static ordering $x_1, \ldots, x_n$. $x_1 \leftarrow v_i \in D_1$.

2. for $i = 2$ to $n$, $x_i \leftarrow d_j \in D_i$ such that adjacency constraints are satisfied as much as possible with $x_1 \ldots x_{i-1}$.

3. if not all consistent, use local repair heuristic, `min-conflicts` [23], or tree search [1] to generate an initial feasible solution

In the general case, it is obvious that it will not be fruitful to include all local schedules generated from stand simulation[2]. A restricted set, i.e., subset of all generated local schedules has to be used. We can then construct a constraint graph and start instantiation (step 1 from above). To take care of the adjacency constraints in an efficient way, each schedule will also have its *exclusion period* available, that is, the time intervals when the stand has an average height below $X - meters$. Instead of using a repair method in step 3, one may use tree search and consistency techniques. This approach needs to be investigated further.

## 5.2 Min-Conflicts Heuristic

It is often the case that while solving a CSP using iterative techniques, an initial trial solution is generated randomly or greedily. This solution in most cases is infeasible, violating various constraints. One way to make this a feasible solution is to repair it iteratively by changing the value assigned to some variable. Which variable to choose for the change is a research issue and often depends upon the problem. One of the methods of choosing the variable to change is the `min-conflicts` heuristic [23]. The heuristic is very simple-minded, yet proven to be effective on large scheduling problems, particularly problems that are loosely constrained. An algorithm using it may be described as follows:

**Algorithm 2**

1. create initial trial solution

2. pick a variable, $x_i$, in conflict

3. assign value $v_j \in D_i$ to $x_i$ such that the conflicts created by it is minimal

4. if no variables in conflict, exit

5. else goto step 2

Step 1 of the algorithm can be implemented either greedily or randomly. We have chosen the greedy approach in our experiments. This step also creates two sets of variables, $VarsDone$ and $VarsLeft$ - the former containing consistent and the latter containing inconsistent variables. Step 2 picks a variable from the set $VarsDone$. In step 3, all the values for variable picked are checked and the

[2]This is depending on characteristics of the particular stand simulator used.

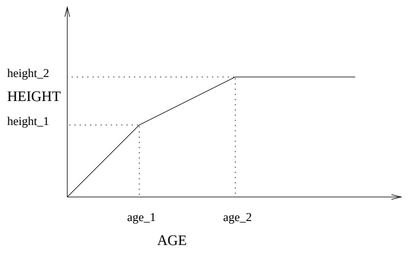

Figure 3: Function used to calculate height

one that minimizes conflicts with other variables is assigned to it. Then, the two sets are updated. Step 4 checks for the terminating condition. The number of iterations can also be included as the terminating condition.

## 5.3 Experimental Results

We have a data generator that takes the following input parameters and produces stand specific data suitable for the routines generating an initial solution. The data file used (first argument) was the ECOPLAN prototype data [24].

In this case, the growth simulation which estimates the average height is based on a simple linear function as shown in Figure 3. In the final versions, a full integration with the stand simulator will be implemented.

**Data Sets** Two basic data sets were used, both generated using the ECOPLAN prototype data. However, these sets are not comparable to the sets used with the CPLEX MIP solver in Chapter 3. The latter does not have treatments other than the clear-cut and are restricted to one treatment per stand. These restrictions were primarily placed due to the inefficiency of general integer programming algorithms. The two data sets generated for this experiment with the `min-conflicts` heuristic have multiple treatments, and a stand may have various treatments during the scheduling horizon.

The difference between the parameters used to generate these two data sets was the value of parameters $MIN\_AGE$ and $MAX\_AGE$, which determined the legal interval (in years) for the clear-cut treatment. For $DataSet_1$, and $Dataset_2$, the intervals were $[35, 90]$ and $[30, 100]$, respectively. From each set, further test sets were generated using different values for the number of alternative schedules, $n$, and the number of scheduling periods, $p$. The domain for $p$ was $\{10, 15, 20\}$, and the domain for $n$ was $\{5, 10, 15, 20\}$.

$Dataset_1$ was tested with $10, 100$, and $500$ iterations. At this point, it was

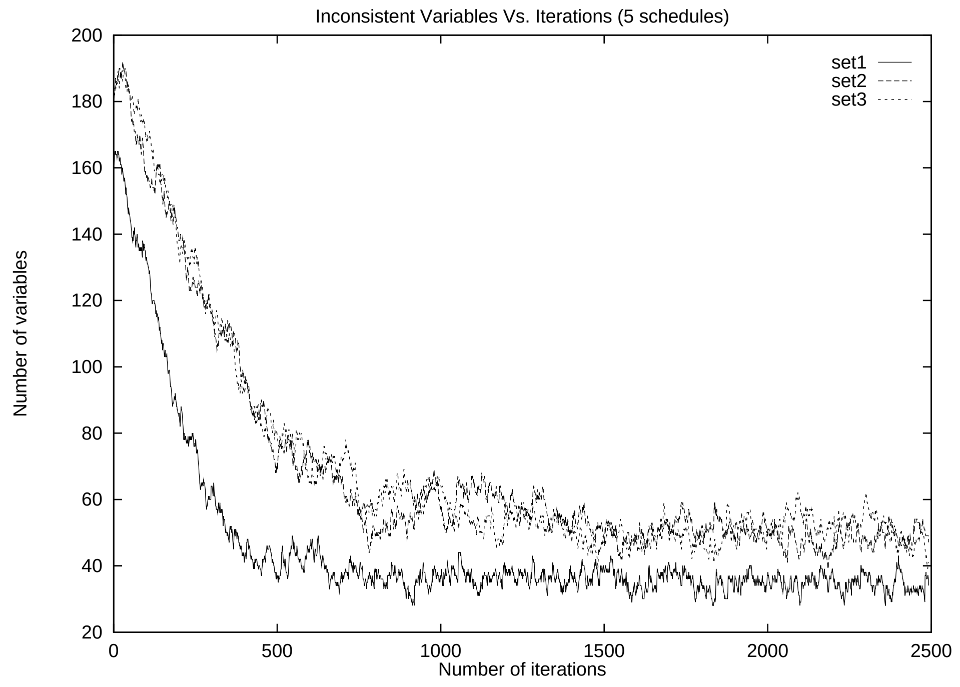

Figure 4: Variables left inconsistent and iterations (each stand has 5 schedules)

observed that some of the consistent assignments were the schedules that had no treatment for all of the periods. This was because of the legal treatment interval which prohibited some stands from having treatments[3]. Since this should not happen when using a simulator, the data generator was modified such that all the schedules generated had at least one valid treatment. This was done by randomly picking one period in which to allow the treatment. $Dataset_2$ was generated in such a way and used for further experiments. For clarity, we divided $Dataset_2$ into 3 sets, $Set_1$, $Set_2$, and $Set_3$, with $p = 10$, $p = 15$, and $p = 20$ with $n$ from the sets $\{5, 10\}$, $\{5, 10, 15\}$, and $\{5, 10, 20\}$, respectively.

Figure 4 shows the result of running our algorithm for instances in which each stand has 5 schedules in its domain, from each of the three sets. Figure 5 is a similar plot but with instances where each stand has 10 schedules in its domain. In Figure 6, results from instances of $Set_2$ and $Set_3$ only are plotted, with each stand having 15 or 20 (only in $Set_3$) schedules in their domains.

## 5.4 Results

Figure 4 shows the result of running our algorithm for instances in which each stand has 5 schedules in its domain, from each of the three sets. Figure 5 is a similar plot but with instances where each stand has 10 schedules in its domain.

[3]For example, there are some stands with current age greater than 100.

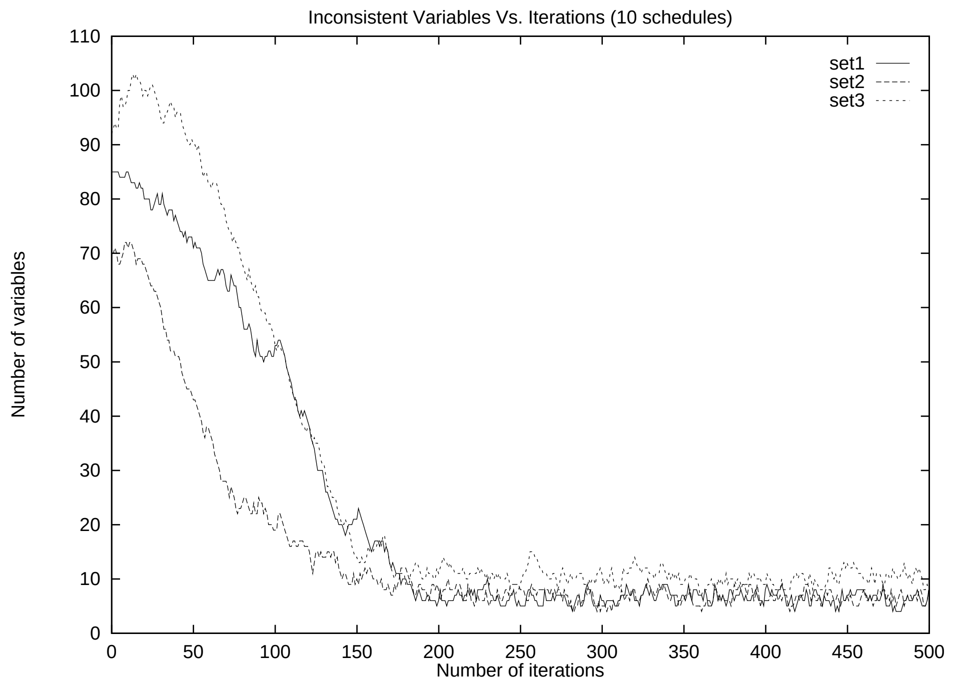

Figure 5: Variables left inconsistent and iterations (each stand has 10 schedules)

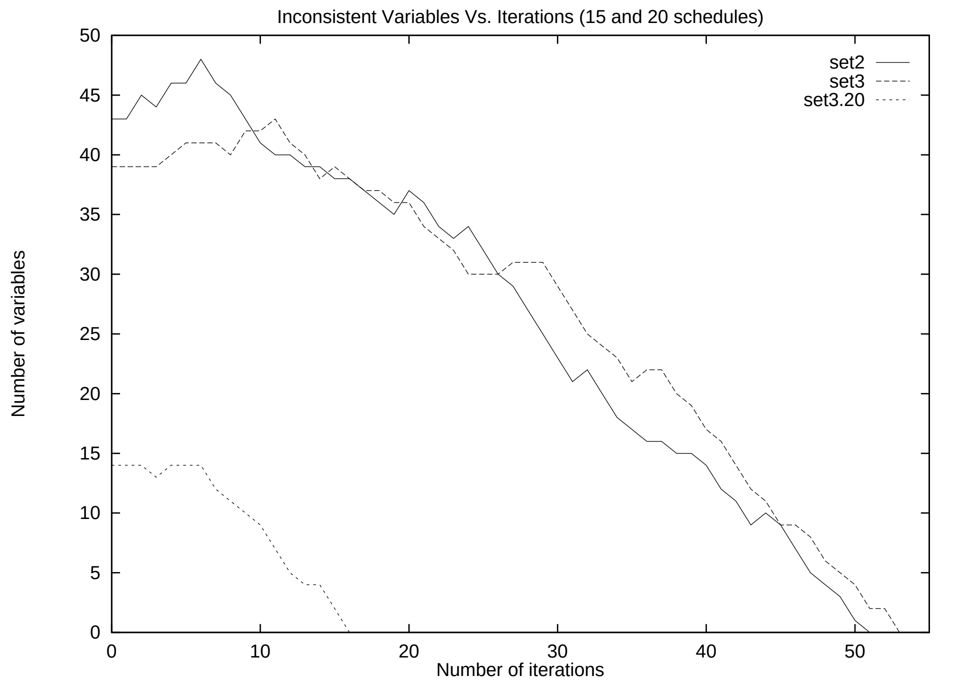

Figure 6: Variables left inconsistent and iterations (each stand has 15 or 20 schedules)

In Figure 6, results from instances of $Set_2$ and $Set_3$ only are plotted, with each stand having 15 or 20 (only in $Set_3$) schedules in their domains.

The figures show that after some iterations the number of inconsistent variables starts to fluctuate within a small interval. The position of this interval depends upon the number of schedules initially available to each stand. In Figure 4 and Figure 5, this interval is [30,60] and [5,10] respectively. In Figure 6, all the variables are consistent within 60 iterations. In fact, this interval is the largest for instances with the least number of schedules as domain. This is as expected - the larger the size of the domain, the greater the chances of finding a consistent solution with a lower number of iterations.

Furthermore, all the instances in Figure 4 and Figure 5 were allowed to run for 10,000 iterations in hope of finding a complete and consistent instantiation. However, the number of inconsistent variables remained within a constant interval throughout the run. Either, our algorithm was trapped in local minima, or there were no better solution to be found.

The results show that this method can be used to find the an initial solution to the LTFTSP. It also suggests that for the ECOPLAN prototype data, if 10 or more local schedules are generated for each stand using a simulator, chances are that the method will converge to a complete or almost complete and consistent instantiation. This is not a very demanding requirement since a stand treatment simulator such as GAYA [7] can be easily modified to output such schedules. However, it is still to be seen whether the schedules generated by such a simulator behave as the randomly generated ones in our experiment.

The repair algorithm took 5–7 minutes of processing time on a SUN Sparc-10 workstation for each of the instances.

## 5.5 Further Improvements and Suggestions

There are many ways to improve the performance of the initial solution generator. The `min-conflicts` heuristic implementation requires the programmer to make certain choices while implementing the trial solution maker, conflict set builder etc. Some of these are outlined below.

1. Right now the constraint graph class is minimally used. However, it can be used to order the variable set using some other heuristic. The rationale for not using it now is that it creates an overhead that is not useful in our test case. If the test case is complicated and large, then it may be beneficial to cluster the nodes of graph according to their degree, neighbors, topology (e.g. cliques) etc. to order the variables before starting repair.

2. The initial trial solution can be constructed randomly instead of greedily if the number of solutions are large. It is so because picking greedily adds overhead that may not be necessary.

3. Introduce some optimization, e.g. area harvested, volume harvested, net present value etc., instead of just performing constraint satisfaction.

## 6 Concluding Remarks

Long term forest treatment scheduling is a very large and complex industrial optimization problem. It typically contains a large number of variables, both hard and soft constraints of various types, and several criteria to be optimised. Earlier work by the authors, including a review of existing approaches for solving forest treatment scheduling problems (e.g., mathematical programming) has lead us to believe that these approaches are not capable of solving adequate models of LTFTSPs of realistic size. A successful systems solution not only requires the application of sophisticated search techniques, but also active use of forestry knowledge and geographical information technology in an integrated decision support solution.

In this paper, a Constraint Satisfaction Problem (CSP) model for the LTFTSP has been presented. We have advocated the use of Iterative Improvement Techniques based on local search and meta-heuristics as a viable approach for solving even large size problem instances. In particular, we have focused on algorithms for building initial LTFTSP solutions to be further improved via IIT.

Results from empirical investigations indicate that real life case problems can be solved efficiently when only hard (adjacency) constraints are considered, i.e., the goal of the initial phase is to generate of an X-m feasible solution. Soft constraints that are disregarded in the first phase may be formulated as objective components to be optimized during the subsequent iterative improvement phase. Our experiments also verified the intuitive fact stating that the size and precise values of the initial domain, i.e., the alternative schedules available for each stand, directly affects speed performance and quality of the initial solution. If the simulator is robust and has the ability to provide a reasonable number of high quality local schedules with some variation for each stand, then the initial solution generator described here will provide good starting points for iterative improvement.

In the near future, we shall develop techniques that will improve an initial solution in an "anytime" fashion. Our initial approach will be iterative improvement based on local search, repair heuristics (such as the `min-conflicts` heuristic), and meta-heuristics (such as tabu search and simulated annealing). We refer to [6, 21] for a discussion of search strategies in the optimization phase. A key to success is to strike the right balance between exploiting the knowledge captured in a stand simulator, and the achievement of maximum search performance. For a treatment of the interface between the search kernel and a stand simulator, we refer to [25]. An integral part of our future work will be to conduct experimental studies on real-life data, taking into account all relevant constraints and objectives. In particular, we shall empirically study how quality of the initial solution affects performance of the improvement phase.

The LTFTSP problem has a significant spatial component. Consider for example, the adjacency constraints. Recent advances in geographical information technology (GIT) should be exploited to handle, analyze, and visualize data as well as results. This is particularly the case when evaluating the visual impact of a schedule, and in general, while handling ecological, recreational, and esthetic constraints. In the ECOPLAN project, we are currently working on the development of a framework that will support efficient and flexible integration of iterative improvement optimization modules, spatial analysis software, and stand simulators [25].

## References

[1] J. R. Bitner and E. Reingold. Backtrack programming techniques. *Communications of ACM*, 18(11):651–656, 1975.

[2] W. Conover. *Practical nonparametric statistics, $2^{nd}$ ed.* Wiley, New York, 1980.

[3] L.O. Eriksson. Two methods for solving stand management problems based on a single tree model. *Forest science*, 40(4):732–758, 1994.

[4] O. Garcia. Linear programming and related approaches in forest planning. *New Zealand Journal of Forestry Science*, 20:307–331, 1990.

[5] Fred Glover. Tabu Search - Part I. *ORSA Journal on Computing*, 1:3, 1989.

[6] G. Hasle and A. Løkketangen. Ecoplan Search Kernel Specification. SINTEF Technical Series, SINTEF Applied Mathematics, 1996.

[7] H. F. Hoen. GAYA-LP: A pc-based long range forest management model. In *EURO XII/TIMS XXXI Joint Intl Conference*, Helsinki, Finland, 1992.

[8] J. Hof and L. Joyce. Spatial optimization for wildlife and timber in managed forest ecosystems. *Forest Science*, 38:489–508, 1992.

[9] W. Hower and Z. Ruttkay. Non-Standard Constraint Processing, August 1996. Working notes of the ECAI-96 workshop W27, 12th European Conference on Artificial Intelligence, Budapest, Hungary.

[10] K. N. Johnson and S. Crim. Forplan version i: Structure and options guide. Technical report, USDA Forest Service Land Management Planning System Section, Washington, DC, 1986.

[11] K. N. Johnson and D. B. Jones. Musyc user's guide and operation manual. Technical report, USDA Forest Service, Washington, DC, 1979.

[12] K. N. Johnson and D. W. Rose. Forplan version 2: an overview. Technical report, USDA Forest Service Land Management Planning System Section, Washington, DC, 1986.

[13] Johnson, K. N. and Scheurman, H. L. Techniques for prescribing optimal timber harvest and investment under different objectives – Discussion and Synthesis. *Forest Science Monograph 18*, 1977.

[14] J. Jones, B. Meneghin, and M. Kirby. Formulating adjacency constraints in linear optimization models for scheduling projects in tactical planning. *Forest Science*, 37(5):1283–1297, 1991.

[15] M. Kirby, W. Hager, and P. Wong. Simulataneous planning of wildland management and transportation alternatives. *TIMS Stud. Management Science*, 1986.

[16] S Kirkpatrick. Optimization by Simulated Annealing. *Science 220*, pages 671–680, 1983.

[17] S. Kirkpatrick. Optimization by simulated annealing: Quantitative studies. *Journal of Stat. Physics*, 34:875–986, 1984.

[18] J. Lappi. A linear programming package for management planning. Research Papers 414, The Finnish Forest Research Institute, Suonenjoki, 1992.

[19] P.E. Linehan and T.J. Corcoran. An expert system for timber harvesting decision making on industrial forest lands. *Forest products journal*, 44(6):65–70, 1994.

[20] C. Lockwood and T. Moore. Harvest scheduling with spatial constraints: a simulated annealing approach. *Canadian journal of forest research*, 23(3):468–478, 1993.

[21] A. Løkketangen and G. Hasle. Solving the Forest Treatment Scheduling Problem using Abstraction and Iterative Improvement Techniques, August 1997. Submitted to 15th Joint Conference on Artificial Intelligence (IJCAI-97), Nagoya, Japan. Not yet accepted.

[22] B. Meneghin, M. Kirby, and J. Jones. An algorithm for writing adjacency constraints efficiently in linear programming models. In *The 1988 Symposium in systems analysis in forest resources.* USDA Forest Service General Tech. Report RM-161, 1988.

[23] Steve Minton, Mark D. Johnston, Andrew B. Philips, and Philip Laird. Minimizing conflicts: a heuristic repair method for constraint satisfaction and scheduling problems. *Artificial Intelligence*, 58:161–205, 1992.

[24] G. Misund, B. S. Johansen, G. Hasle, and J. Haukland. Integration of Geographical Information Technology and Constraint Reasoning - A Promising Approach to Forest Management. In Jan Terje Bjørke, editor, *SCANGIS '95 - Proceedings of the 5th Research Conference on GIS, 12th-14th June 1995, Trondheim, Norway*, May 1995.

[25] G. Misund and S. Stikbakke. A Generic Framework for Multilevel Treatment Schedule Simulation, May 1997. Accepted for the 1997 Symposium on Systems Analysis in Forest Resources, Traverse City, Michigan, USA.

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

[27] Navon, D. I. Timber RAM – A long-range planning methods for commercial timber lands under multiple-use management. Technical report, USDA Forest Service, Research Paper PNW-70, 1971.

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

[29] T. Pukkala and Kangas J. A heuristic optimization method for forest planning and decision making. *Scandanavian Journal of Forest Research*, 8:560–570, 1993.

[30] J. P. Roise. A nonlinear programming approach to stand optimization. *Forest Science*, 32, 1986.

[31] S. Snyder and C. ReVelle. The grid packing problem: Selecting a harvesting pattern ina n area with forbidden regions. *Forest Science*, 42(1):27–34, 1996.

[32] S. Stikbakke. Personal communication. Ph.D. student at NISK, 1996.

[33] J.M. Torres-Rojo and J.D. Brodie. Adjacency constraints in harvest scheduling: an aggregation heuristic. *Canadian journal of forest research*, 20(7):978–986, 1990.

[34] E. Tsang. *Foundations of Constraint Satisfaction.* Harcourt Brace and Co., 1993.

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

[36] A. Wientraub, G. Jones, A. Magendzo, M. Meacham, and M. Kirby. A Heuristic System to Solve Mixed Integer Forest Planning Models. *Operations Research*, 42(6):1010–1024, 1994.

[37] A. Yoshimoto and J. D. Brodie. Comparative analysis of algorithms to generate adjacency constraints. *Canadian Journal of Forest Research*, 24(6):1277–1288, 1994.