Integration of Geographical Information Technology and Constraint Reasoning - A Promising Approach to Forest Management

Gunnar Misund Bjørn Sigurd Johansen Geir Hasle
SINTEF Informatics, P.O. Box 124, Blindern, N-0314 Oslo, Norway
{Gunnar.Misund, Bjorn.Johansen, Geir.Hasle}@si.sintef.no

Jørgen Haukland
Gjøvik College, P.O. Box 191, N-2801 Gjøvik, Norway Haukland@gih.no

Abstract

In this paper we describe synergy effects of combining state-of-the-art Geographic Information Technology (GIT) with novel methods for planning and scheduling from the field of Constraint Reasoning (CR). An integrated problem solving strategy has been developed for a problem arising in forest management. In particular, we have developed a method for solving what we have called the Clear-Cut Scheduling Problem (CCSP), where the task is to assign clear-cutting times to regions in a given forest area over a long term horizon. The schedule must satisfy certain ecological, recreational and economical constraints, and, in addition, optimise on a number of partially conflicting criteria. Our approach is based on the combination of advanced spatial analysis with an Iterative Improvement Technique in tandem with the Tabu Search meta heuristic. We have developed a software prototype called ECOPLAN with functionality for generation, presentation and modification of harvest schedules in an integrated decision-support environment. Empirical experiments have been carried out using a real-life test case consisting of a forest property subdivided in approximately 500 stands.

Keywords: Forest harvesting, Scheduling, GIS, Optimisation, Tabu Search.

1 Introduction

This article presents some results from a research project at SINTEF Informatics carried out in 1994. The main objective was to explore the assumed synergy in combining two separate R&D fields: Geographic Information Technology (GIT), and planning and scheduling based on Constraint Reasoning (CR). Forest management was chosen as an interesting and challenging application area.

GIS and computer aided scheduling and planning

The research project was initiated by the observation that many planning and scheduling activities incorporates spatial information, and vice versa that users of geographical information systems (GIS) often demand advanced analysis functionality to solve problems associated with planning or scheduling problems.

Brief surveys and market analyses, and common experience, showed that existing systems for computer aided planning and scheduling have none or very limited capabilities

to handle problems incorporating a spatial component, and that existing GIS offer very primitive facilities for planning and scheduling, if any at all. A technical report on the field of route planning and fleet management was prepared in the project [Gje94], and the report clearly demonstrates the lack of integration of GIT and planning in this particular problem domain.

New challenges in forest management

To study the effects of integrating GIT and CR, we selected *forest management* as an application area. NORSKOG, a Norwegian association of forest owners, provided the problem formulations and the data needed to design, implement and test a software prototype for decision support in long term forest scheduling.

During the past decade, the forest trade has faced a set of new challenges, both in Norway and in other parts of the world. Authorities and market segments demand accomplishment and documentation of sustainable forest harvesting. In addition to reaching economical objectives, the trade also has to take care of ecological concerns, such as wildlife preservation and biological diversity, and occasionally also have to achieve recreational objectives.

To document that external restrictions are being followed, and to actually carry through a sustainable forest harvest, long term treatment schedules are considered one of the main vehicles. Based on Norwegian legislation in general, and, in particular restrictions on forest harvest in the near-city areas of Oslo, a case example was constructed in collaboration with NORSKOG:

Given a forest subdivided into a set of stands, *i.e. regions that are considered homogeneous with respect to certain parameters such as soil fertility, wood species, average age and so forth, the goal is to compute a schedule that for each year in a the planning period specifies a set of stands to be clear-cut. The schedule should satisfy certain constraints and be satisfactory according to a set of associated criteria:*

- ***2–m constraint:*** *All neighbour sites of a clear-cut stand have to be higher than 2–m.*
- ***Even Consumption:*** *The total volume of harvested forest should be as even as possible from one year to another.*
- ***Old Forest:*** *The total area of "old" forest should be kept above a given level over the entire period.*
- ***Optimal Harvest Time:*** *The harvested forest should be as "ripe" as possible, not too old, nor too young.*
- ***Visual Impact:*** *Given a set of predefined viewpoints in the landscape, the spatial distribution of the clearings should minimise the reduction of the aesthetically quality of the sceneries.*

The planning period is typically 100 years, and the forest properties may consist of from 50 to 5,000 individual stands.

Long term forest harvesting schedules are commonly used in the management of about 70% of the total productive forest area in Norway. Currently, there is no software available to support scheduling under the new constraints and criteria imposed by the authorities and the market.

The remainder of the paper is organised as follows: In section 2 we define and discuss the CCSP from a scheduling point of view. We outline some of the traditional approaches and describe the chosen CR-strategy, the *Iterative Improvement Technique (IIT)*. The design and implementation of an experimental prototype called *ECOPLAN* is described in section

3. A specific test case is presented in section 4, within which we describe the experiments that have been carried out in order to study the behaviour and performance of ECOPLAN. In section 5 we discuss briefly some of the results and suggest further research in the field.

2 The Clear-Cut Scheduling Problem (CCSP)

We define the *Clear-Cut Scheduling Problem (CCSP)* as the assignment of clear-cut a year to individual stands in a forest area. Given a forest area and its subdivision in stands, the task in CCSP is to generate a harvesting schedule for a horizon of, say, 100 years which a) satisfies a number of constraints, and, b) strikes a careful balance between several criteria. Adequate models of the CCSP will represent a complex combinatorial optimisation problem.

In this section, we shall discuss the successful solution of an instance of the CCSP, which has a specific set of constraints and optimisation criteria.

2.1 A Semi–Formal Definition of the CCSP

Our semi–formal definition of the CCSP consists of a topological description with a partitioning of a forest area into regions (stands), individual parameters for regions, a description of the problem variables, a description of the hard and soft constraints[1], a description of the criteria that define the quality of a harvest schedule.

The goal of the CCSP is to find a *complete*, *consistent*, and *optimised* clear-cut harvesting schedule s. By *complete* we understand that each region must be assigned a time for future treatment. By *consistent* we mean that the schedule s must not violate any hard constraints. By *optimised*, we understand that s must optimise certain criteria, e.g., by minimising or maximising the value of a defined objective function, and, s must satisfy all soft constraints to as large degree as possible. Time granularity is 1 year and there is only one type of forest treatment.

Topological description. The considered forest area F contains n non-overlapping regions $R_1, \ldots, R_n$ in such a way that, when regarding the forest area and the regions as point sets in the plane, $R_1 \cup \ldots \cup R_n \subseteq F$ and $\forall i, j \in [1, n],\ i \neq j : R_i \cap R_n = \emptyset$. The underlying assumption is that each region is homogeneous with respect to the forest properties that are relevant to harvest scheduling (i.e., regions are stands). For every region $R_i,\ i \in [1, n]$ a set of neighbours $N(R_i)$ is defined. $N(R_i) \subset \{R_1, \ldots, R_n\}$. In a similar way, we define $I(i) \subset [1, n]$ as the index set for the neighbours of R_i.

Individual parameters/functions. For every region $R \in \{R_1, \ldots, R_n\}$, several parameters/functions are given:
$LH(R)$ denotes the time of the most recent harvesting.
$EARLY(R)$ denotes the minimum duration between harvests.
$LATE(R)$ denotes the maximum duration between harvests.
$OPT(R)$ denotes the optimal time between harvests.
$TM(R, H)$ denotes the time it takes for trees to grow from 0 to a certain height H.
$AREA(R)$ simply denotes the area of R.
$GROWTH(R, y)$ is the volume that may be harvested y years after the last harvesting.
The CCSP variables. Let h_i denote the scheduled harvesting year for region R_i. $\{h_i\},\ i =$

[1] A hard constraint is a relation which must be satisfied. A soft constraint may be relaxed.

$1, \ldots, n$ are the problem variables. We shall also use $h_i(s)$ to denote the scheduled harvesting year for region R_i relative to a particular schedule s. The variables have domains that represent a part of the time line. We have selected an integer interval representation: $D_i = [0, M]$, where 0 represents current year, and M is typically the scheduling horizon.
Hard constraints. Before a region may be harvested, it is required that every neighbouring region has an average tree height of say at least 2 meters. Thus, for a feasible schedule s, the following must hold:

$$\forall i \in [1, n] : \forall j \in I(i) : (h_i(s) < h_j(s)) \vee (h_i(s) \geq h_j(s) + TM(R_j)).$$

We shall denote these hard constraints the *2–m constraints*.
Soft constraints. For economic and quality reasons, there are bounds on times between harvesting: $\forall i \in [1, n] : EARLY(R_i) \leq h_i - LH(R_i) \leq LATE(R_i)$. These constraints may be relaxed in order to fulfill the 2–m constraint.
Criteria for an optimised schedule s**.** Below we shall describe and suggest formal definitions for the four major optimisation criteria identified by forestry experts for the CCSP.
C_1*: Optimal Harvesting Time.* For every region R_i, the harvesting time should be as close as possible to its optimal harvesting time. E.g., the value $\sum_{i=1}^{n}(\mid OPT(R_i) - (h_i - LH(R_i)) \mid)$ should be minimised.
C_2*: Even Consumption.* The estimated harvesting volumes for each year $EHV_y(s)$, $y \in [0, M]$ should be as close as possible to the average harvested volume $AVH(s)$. E.g., the value $\sum_{y=0}^{M}(\mid EHV_y(s) - AVH(s) \mid)$ should be minimised.
C_3*: Old Forest.* It is desired that the schedule maintains a minimum area of old forest AOF over the schedule horizon. Let $OLD(y, s)$ denote the area of old forest in year $y \in [0, M]$ resulting from the schedule s. We then want to minimise for instance $\sum_{y=0}^{M} \max(0, AOF - OLD(y, s))$.
C_4*: Visual Impact.* The schedule should minimise visual damage relative to a given set of viewpoints. For instance, we may want to minimise the maximum sum of projections of clear-cuts from a given set of critical viewpoints over all years. Assuming one viewpoint V, and denoting a quantification of the visual impact from region R in year y relative to viewpoint V as $VIS(R, y, V)$, the schedule should minimise: $\max_{y=0}^{M}(\sum_{i=1}^{n} VIS(R_i, y, V))$.

2.2 Properties of the CCSP

Focusing on the hard 2–m constraints, the CCSP may be regarded as a *Constraint Satisfaction Problem (CSP)* [Tsa93]. In particular, there are strong similarities between the CCSP and the *Graph Colouring Problem (GCP)*. Informally, the task in the GCP is to assign colours (from a given set of colours) to the nodes in a graph in such a way that no neighbours is given the same colour. The GCP belongs to the class of **NP**–complete problems [GJ79], for which there probably does not exist any efficient (polynomial) algorithm. In practical terms, this means that we cannot expect to find the optimal solution for large problem instances as this would require too much computation due to combinatorial explosion.

Although there are additional constraints and objectives, our conjecture is that the CCSP is **NP**–hard[2]. We must therefore lower our expectations and concentrate on finding high–quality solutions in limited time. To illustrate the size of the search space, assuming an

[2] A proof is beyond the scope of this paper.

effective domain size of 50 for a typical number of 500 regions, the total number of schedules (counting both feasible and infeasible solutions) is $50^{500} \approx 10^{850}$. A current estimate for the total number of atoms in the Universe is only 10^{120}. Returning to the optimisation aspects, it is clear from the above description that the CCSP involves the balancing of several potentially conflicting criteria. This problem characteristic also contributes to the complexity of the CCSP, a complexity which calls for efficiency, robustness and flexibility in optimisation techniques to be selected for CCSP decision–support tools.

2.3 Problem Solving Techniques for the CCSP

For constrained combinatorial optimisation problems (such as the CCSP), alternative problem solving techniques exist. Our selection of technique has been motivated by the requirements for end user forest management systems outlined above.

Mathematical Programming. *Linear Programming*, in particular, *Mixed Integer Programming (MIP)* and *Goal Programming (GP)* has been applied to the CCSP and similar problems[WMMK94]. Our initial attempts to formulate the CCSP as a MIP has lead us to conclude that this approach is not well suited, for the following reasons:

- their lack of flexibility in expressing constraints and objective criteria
- their lack of support for mixed-initiative problem solving
- their lack of repositories for heuristics to guide combinatorial search

Systematic Tree Search (STS). Taking a Constraint Satisfaction Problem perspective on the CCSP, several backtracking tree search and consistency techniques are viable. *Standard Backtracking (SB)* may be seen as the basis for these techniques. SB will iteratively construct a solution by successively assigning values to the problem variables (i.e., assign a harvesting year to one region, then to another, and so on) while checking whether constraints are satisfied. If no value is possible for the current variable due to constraint violations, the algorithms will backtrack and try a new value for the previously instantiated variable. STS is complete, i.e., it effectively enumerates all solutions.

Due to its exponential time complexity, STS is not an alternative for the CCSP without the addition of strong *heuristics*, i.e., knowledge, general or problem specific, that may rapidly guide search towards a solution. Examples are variable ordering and value ordering heuristics that will decide the sequences of variables and values in the instantiation process. In addition, consistency techniques may be used to prune the search space by removing values that may not be included in a solution. Except for questions about adequate response performance, STS suits the above requirements for forest management decision–support.

If the more "intelligent" variants of STS fail, one may have to sacrifice completeness[3] for response time and employ more powerful heuristics in a non-systematic search regime.

We have evaluated STS with several variable and value ordering heuristics in the context of finding a 2–m feasible solution. In initial empirical investigations it was not possible to obtain a solution within acceptable response limits, even for small CCSP problem instances. More recent investigations using alternative variants of STS have shown promising results, but more work remains to be done.

Iterative Improvement Techniques (IIT). The initial failure of systematic search techniques pointed us to IIT, which, over the past few years, have shown remarkable performance in providing high quality solutions to scheduling problems in limited time [Dor95]. The basic idea is *neighbourhood search*, i.e., given a complete (but possibly non-feasible

[3] I.e., the guarantee that the algorithm will find a solution if there is one.

and sub-optimal) solution, generate a neighbourhood for this solution by applying a set of modification operators. The search for a better (i.e., more optimal and more feasible) solution then proceeds iteratively by selecting the best neighbour as the new current solution. In this basic form, IIT is a hill-climbing algorithm which tends to get stuck in local optima. To remedy this, so-called *meta-heuristics* may be employed, e.g., *Simulated Annealing (SA)* [Kir83] or *Tabu Search (TS)* [Glo90]. IIT may at first glance seem most appropriate for problems where a feasible solution is known to exist (and is relatively easy to find). However, one may apply a strategy similar to Goal Programming to add robustness and automatic constraint relaxation facilities. An initial solution may be generated randomly, by greedy heuristics, or even by STS.

A particularly nice feature of IIT in the context of decision–support is their *anytime* characteristic. The iterative problem solving process may be interrupted at any time, and the best solution so far is available for presentation. Moreover, a candidate solution may be modified by the user and used as the seed for a new iterative improvement process. IIT has earlier been applied to forest management problems, e.g., to solve the afforestation problem[MJTVV92].

2.4 IIT - The Selected Search Strategy

Our selected search strategy for the CCSP is IIT with the Tabu–Search (TS) meta–heuristic [Glo90]. TS is composed of a neighbourhood operator, an evaluation function for neighbours, a tabu criterion, and an aspiration criterion. A method for generating an initial candidate solution is needed to initiate the iterative improvement process. Below, we discuss these elements of TS in the context of the CCSP and our concrete implementation.

The Evaluation Function assigns goodness values to candidate schedules. The evaluation function is critical as it is used to guide search. We have selected a relatively simple approach where the evaluation function is a weighted sum of the 4 optimisation criteria components described above. In addition, a penalty function for violations of the 2–m constraints is introduced as an evaluation function component[4]. Hence, the selected evaluation function may be expressed as: $\sum_{k=0}^{4} w_i C_i$. For decision–support our approach is attractive because it allows end users to interactively experiment with different weights and thus overcome the rigidity of static optimisation functions. However, the selection of appropriate weights may be non-trivial.

Neighbourhood Operator. We have selected a neighbourhood operator which simply generates the neighbourhood by modifying exactly one harvesting year. Let $\Gamma(s)$ denote the neighbourhood of schedule s. The operator is defined by:

$$\Gamma(s) = \{s' : \exists' i \in [1,n] : (h_i(s) \neq h_i(s') \wedge (\forall j \in [i,n],\ j \neq i : h_j(s) = h_j(s'))\}.$$

(We have taken the liberty to use a modified existence quantifier $\exists'$ which has the meaning "there exists one and only one"). Moreover, we have chosen to let Γ only generate local-feasible harvesting years to the selected modified region, i.e., we have added the condition: $EARLY(R_i) \leq h_i(s') - LH(R_i) \leq LATE(R_i)$[5]. The size of the neighbourhood is: $\mid \Gamma(s) \mid = \sum_{i=1}^{n} \mid LATE(R_i) - EARLY(R_i) \mid$. Hence, we have a simple operator which generates a large neighbourhood.

[4] The approach is similar to Goal Programming.

[5] This condition may be relaxed in order to increase the probability of satisfying the 2–meter constraints.

Initial Schedule. We have selected a greedy algorithm for generating the initial schedule s_0. It simply consists of assigning the local–optimal harvesting time to every region, where possible. More precisely, we have: $\forall i \in [1,n] : h_i(s_0) = \max(0, LH(R_i) + OPT(R_i))$.

The Tabu Criterion specifies moves that are tabu and thus will not be executed. In TS, the iterative improvement basically consists of movement to the neighbour with the best value of the evaluation function. To escape from local optima, neighbours with certain defined properties are defined as tabu. Currently, we use a simple criterion, stating that we are not allowed to move to a neighbour which had its harvesting year changed within a certain number of iterations.

The Aspiration Criterion. In TS, a move which is defined as tabu may (on second thoughts) be performed if allowed by an aspiration criterion. Our current choice of aspiration criterion simply checks for global improvement. If a move is deemed tabu, but will result in the best schedule encountered so far, the move will be performed anyway.

3 Design and Implementation of ECOPLAN

As suggested by Pukkala and Kangas [PK93], there are several requirements for forest management planning systems to be accepted by end users. They mention user friendliness, optimisation capabilities, and sufficient level of detail. On our own account, and, particularly relevant to CCSP tools, we would like to add representational adequacy (i.e. that the underlying problem model is adequate), heuristic adequacy (i.e. that high-quality solutions will be presented in reasonable time), and flexibility towards modification of data, constraints, and criteria. These requirements, as well as the overall CCSP characteristics, strongly suggest a decision-support approach. Moreover, tight integration with GIS, as well as an excellent user interface for controlling the scheduling process, are needed. Previous attempts to solve the CCSP have generally failed to address the full set of requirements.

The ECOPLAN prototype is designed as a synthesis of four modules. The core of the system is a close integration of a scheduling engine called the *IIT Kernel* and a set of *GIT Services*. These two modules are embedded in an interface environment to facilitate interaction with the operator and communication with data sources, as illustrated in figure 1.

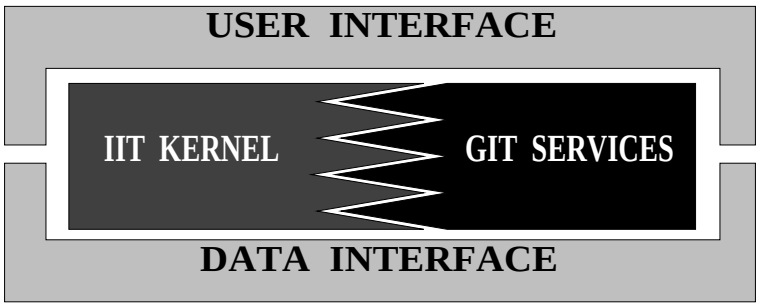

Figure 1: Conceptual view of ECOPLAN architecture.

The functionality of the four components is briefly outlined below.

User Interface: Information on and access to input data. Control of the optimisation process (parameter settings, manual interruptions). Modification of existing constraints/criteria and addition of new ones. Selections of output presentations.

Data Interface: Communication with external data sources. Conversion of input data.

GIT Services: Generation of customised terrain models from scattered data, such as elevation contours, 3D data on road and stream networks and geodetic points. Generation of consistent topological models from spaghetti data of site polygons. Spatial calculations of for example area, perimeter and distance. Visual viewpoint analysis. Preparation of colour coded digital maps. Preparation of data for 3D visualisation.

IIT Kernel: Modeling and management of all information relevant to the scheduling optimisation. Methods and algorithms for iterative schedule improvement.

Please note that the list refers to the design of a full-blown version of ECOPLAN. The current prototype is developed with emphasis on the IIT Kernel and the GIT Services. Thus, the user and data interfaces have been only rudimentary implemented.

The IIT Kernel was implemented from scratch in the object-oriented language C++. Special attention was paid to achieve a flexible structure, easy to modify, enhance and extend. The interfaces to user, the data sources and the GIT Services module were implemented with well defined parameter descriptions. The modularity of the Kernel makes it easy to plug in new iterative improvements methods (see section 2.3). The constraints and criteria are implemented as specialisations of a generic class, thus it is straightforward to define and implement new ones.

The GIT services is a package composed of a diversity of public domain tools and a suite of spatial analysis methods designed and developed particularly for the ECOPLAN module. The terrain generation and analysis, e.g. the Visual Impact analysis, is implemented by extensive use of the SISCAT library [sis95], [ADH95]. SISCAT is a comprehensive C++ based toolkit for construction of surfaces from various kinds of scattered data.

Currently, the user interface is implemented exclusively with development and evaluation of the IIT Kernel and the GIT Services in mind. The user may control the behaviour of the ECOPLAN module by means of an input file with a set of well documented parameters. The results from the schedule optimisation may be presented in various ways (See section 4 for some examples of different ECOPLAN outputs):

- Alpha-numerical tables with statistics and other key figures.
- Colour coded maps for each year in the optimised schedule.
- Map animation (mpeg) of all years in the plan.
- 3D vista rendering from a given viewpoint.
- 3D vista animation (mpeg) of all years.
- Graphs of goodness development of constraints and criteria over the iterative improvement sequence.
- Animated graphs of some of the key figures from the consecutively improving schedules.

Input of data, e.g. digital maps of stands (geometry) and site-specific information such as average age, soil fertility and growth functions, are currently only performed by means of flat ASCII files.

4 A Case and Some Results

To enable the investigation of behaviour and performance of the ECOPLAN prototype, we were provided data on a forest area in the South-Eastern part of Norway. The area is about 16km^2, of which roughly speaking 85% is considered productive. The forest is subdivided

into approximately 500 stands, i.e. homogeneous units with respect to wood species, age classes and site quality. The average size of a stand is $28,000\text{m}^2$, varying from 200m^2 to $148,000\text{m}^2$. A map of the geometry of the individual stands is presented in figure 2.

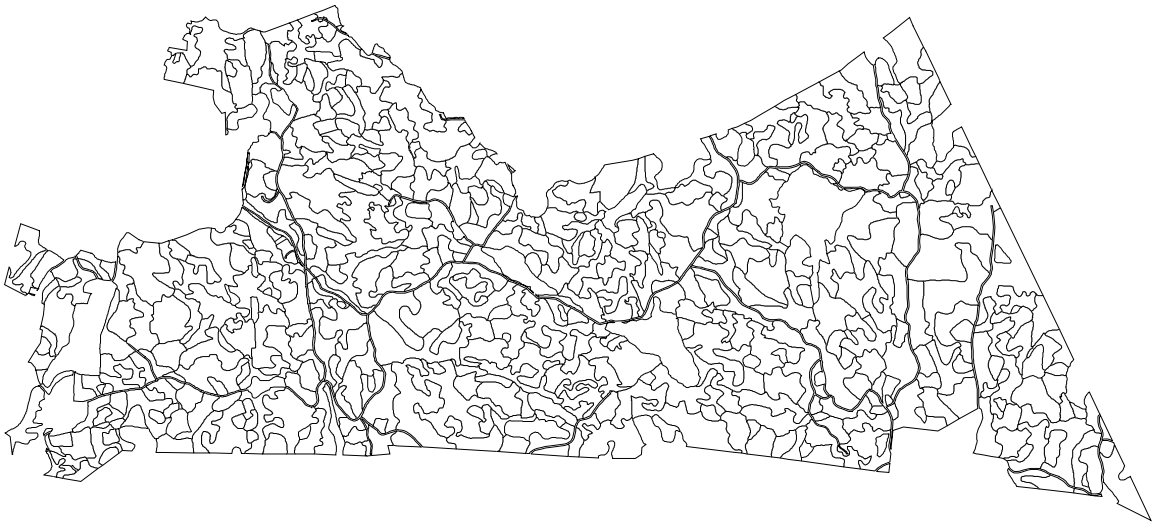

Figure 2: Geometry of the stands.

The ECOPLAN prototype handles site-specific information, but still some simplifications have been made:

- All stands consist of only one single wood species.
- All stands are considered equal with respect to site quality.
- The volume growth function is designed as simply as possible, i.e. as linear growth up to a given threshold age and stagnation thereafter.
- The only treatment considered is clear-cutting.
- "Old" forest is defined to be older than 60 years for all stands.

The forest is relatively young, about 60% of the total area consists of forest which is less than 30 years old. A part of the forest is presented in figure 3, where we have divided the stands into five age categories.

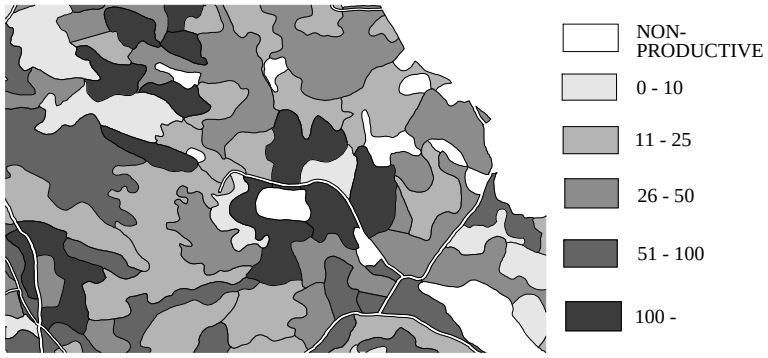

Figure 3: Initial age state of case forest.

The scheduling horizon is 100 years, and for each of these years we want to assign a set of stands to be treated. In the next sections we present some of the results from a series of

experimental runs of ECOPLAN with different parameter settings. We focus on one single constraint/criterion at the time. However, the examples are generated with all constraints simultaneously active (i.e. that all weights in the objective function are non-zero).

4.1 2–m constraint

The 2–m constraint implies that all neighbour stands of a clear cut region must be higher or equal to 2 meters. Experiments showed that the neighbour constraint is totally satisfied after a relatively low number of iterations. The number of violated neighbour relations typically dramatically decreases during the first hundred rounds, and it takes equally many iterations to resolve the last few conflicts.

Figure 4 shows a typically development of the goodness of the 2–m constraint. For each iteration we have plotted the accumulated sum of broken neighbourhood relations, weighted with how "grave" the violations are, i.e. how large (in years) the difference is between the clearing and its illegal neighbours.

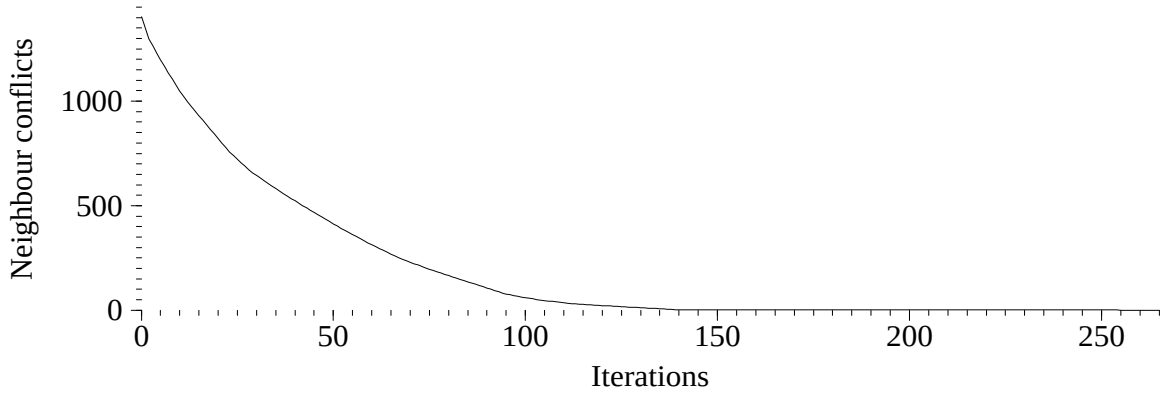

Figure 4: Iterative improvement of 2–m constraint.

In figure 5 we see how a neighbourhood conflict is resolved (see also figure 3). The left map focus on the illegal situation, which is taken from the first year in the schedule devised in iteration no. 254. The right map is from the optimised plan from the 255th improvement, where the 2–m constraint is satisfied.

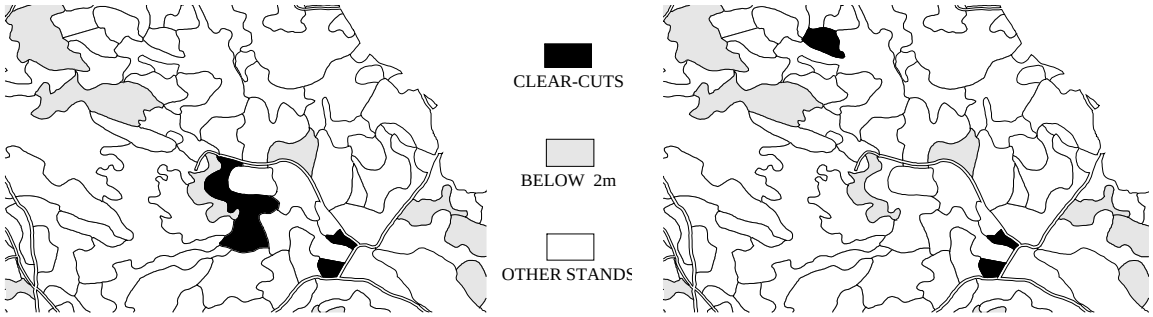

Figure 5: First year in optimised schedule, 254 vs. 255 iterations.

4.2 Optimal Harvest Time

The initial solution is defined as a schedule where the Optimal Harvest Time criterion is maximised, i.e. that every stand is clear-cut exactly in the year when the forest is considered to be as "ripe" as possible, not unripe, nor overripe. In this context, the criterion may be regarded as a soft constraint. However, the optimisation schema prevents harvesting of regions below or above certain threshold ages (see section 2.1). Thus, the harvest time also acts as a hard constraint.

In figure 6 we have plotted the number of clear-cut stands according to how many years their harvest date deviate from the local optimum[6]. We observe that in the first schedule a large number of regions are harvested at optimal time. Since the initial state imply that we have a number of very old stands, these must be harvested in the first year in addition to the optimal regions.

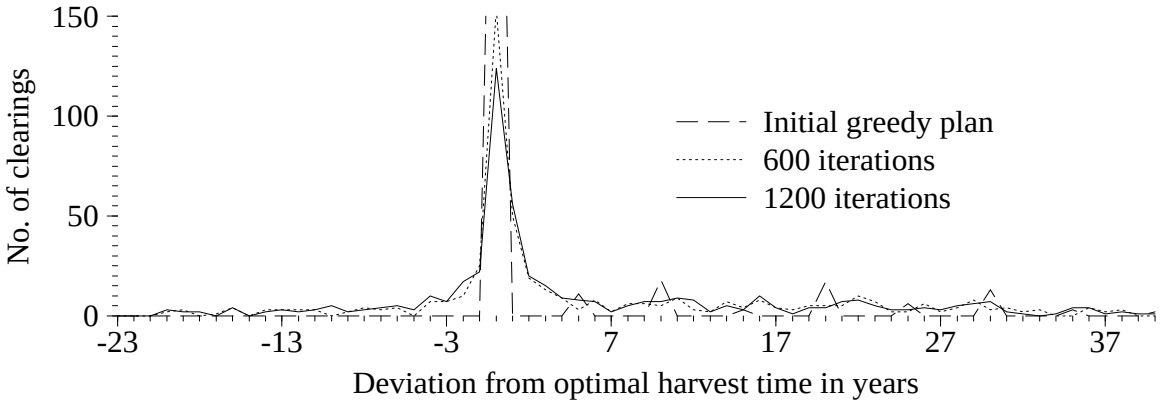

Figure 6: Development of the Optimal Harvest Time criterion.

In contrast with the other criteria, the Harvest Time goodness decreases during the optimisation process when all constraints/criteria are active. The negative development is an unavoidable consequence of the fact that the initial solution is optimal with respect to this criterion, but produces very low goodness values for all the other constraints and criteria. However, we observe that after 1200 iterations about 25% of the regions are still optimally harvested, many stands are clear-cut at near-optimal time, and relatively few sites are distributed in the sub-optimal part of the legal interval.

4.3 Even Consumption

The Even Consumption criterion implies that the optimisation process seeks to distribute harvesting volume over the scheduling period in an even manner. The total volume of consumed forest should not vary to much from one year to another.

In figure 7, we show how this criterion is gradually improving by plotting the harvested volume each year for three different schedule, the initial greedy solution and the schedules resulting from 600 respectively 1200 improvement iterations.

[6]The y-axis interval is truncated in order to focus on the interesting parts of the graphs. This also applies to figure 7.

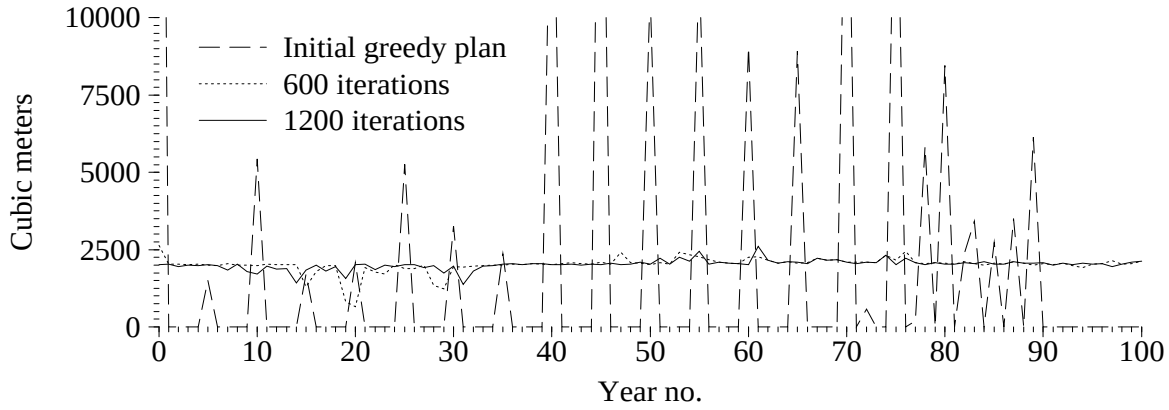

Figure 7: Development of the Even Consumption criterion.

The initial solution is, as described in section 2.4 and 4.2, generated by cutting the forest at the local optimal harvest time. Due to the structure of the input data (the stands are classified according to five year intervals), we get large consumed volumes every 5th year and no activity in the intermediate years. The schedule is considerably improved after 600 iterations, and after 1200 rounds the yearly consumption seems to stabilise at an approximately even level.

4.4 Old Forest

Due to ecological considerations, the areal percentage of old forest should be kept above a certain threshold level. In our case, we define old forest to be 60 years or older, and we want 40% or more of the total area to be occupied be stands of this age class.

In figure 8, we see how the greedy strategy of the initial schedule diminishes the area of old forest in the first years. This is due to the initial state of the forest, with a relatively large amount of young stands, and that regions with higher age than optimal will be cut in the first year. Then the forest is allowed to gradually age, before we enter a period of intensive harvesting which drastically lowers the number of old regions.

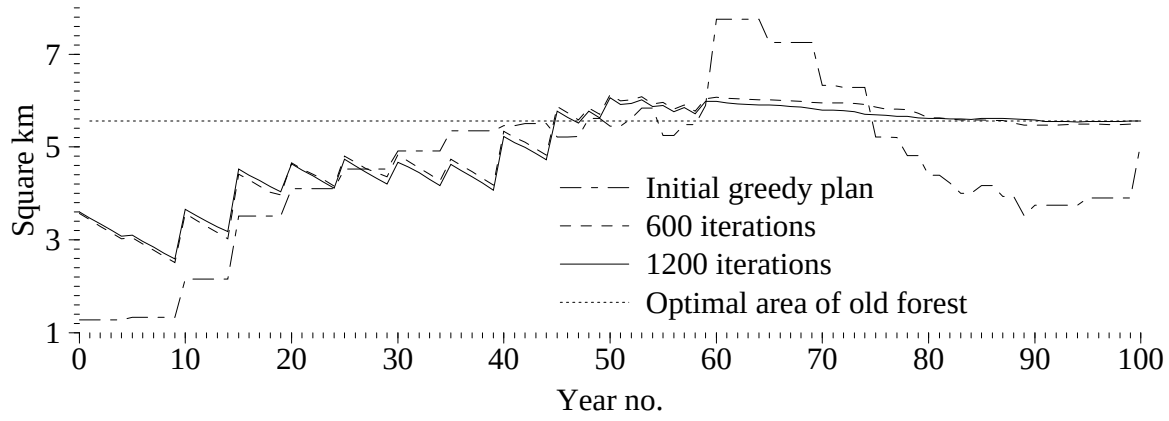

Figure 8: Development of the Old Forest criterion.

This development is considerably improved during the optimisation process. We notice that we reach a stable situation after about 70 years of harvesting, were the wanted level of

old forest is reached. Note that it would not be possible to reach equilibrium at an earlier stage of the plan, due to the history of harvesting expressed by the initial state.

4.5 Visual Impact

In figure 9, we have simulated the visual impact from a given viewpoint in a certain year of a good respectively bad schedule. In the 3D views, clearings and young stands are rendered light grey, while older forest is darker. The left vista present a landscape which may be characterised as totally demolished, while the right one is more aesthetically satisfactory.

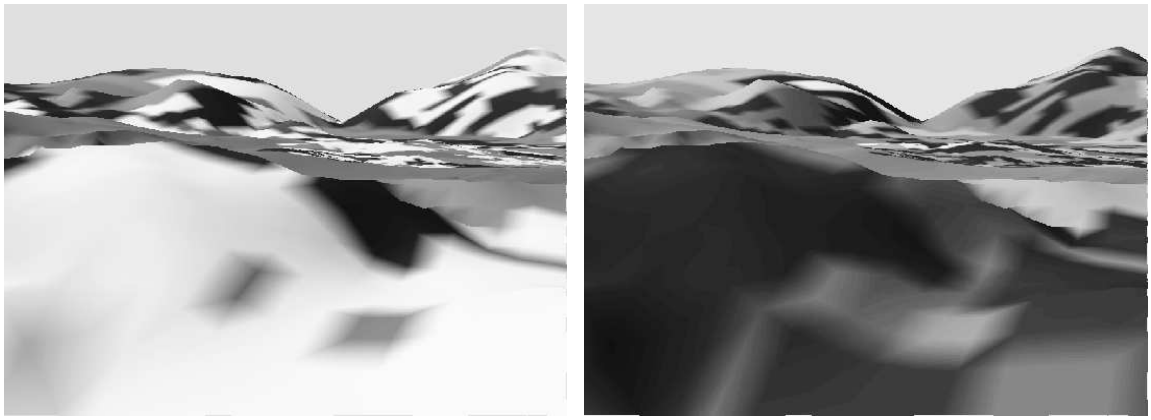

Figure 9: Visual Impact criterion, bad and good vistas.

The optimisation procedure will favour a schedule were many of the vistas are visually pleasant, and punish heavily solutions which contain visually unpleasant harvest patterns.

4.6 Performance

To facilitate interaction between the the ECOPLAN module and the operator(s), the performance of the system in terms of speed is crucial. Our prototype is implemented in a UNIX environment, and the tests have been conducted on standard mid-range work-stations.

To establish the values of the various parameters, an optimisation session typically started with test-runs with a low number of iterations, and perhaps with only one or two active constraints. Finally, an optimisation process involving a high number of iterations has been executed to gain a satisfactory solution.

One single pass involves a preprocessing phase (see section 3) and a number of improvement steps. The preprocessing typically lasts from 20 to 30 seconds up to a couple of minutes, depending on the input data. The ECOPLAN prototype generates from ten to hundred new schedules per minute. Quite intuitively, the performance has shown to be relative to the number of active constraints.

The ECOPLAN prototype has been implemented with little attention to CPU performance requirements. However, the obtained performance has proven quite sufficient for experimental use with problems of size and complexity equal to the case described in the last sections. Clearly, there is a significant potential in optimising the implementation.

5 Discussion and Further Work

As mentioned in section 1, the primary objective in the research project was to explore the integration of GIT and scheduling in general. Secondarily, we wanted to specifically study the case of long term forest treatment scheduling under multiple objectives and constraints.

The ECOPLAN prototype

As documented in section 4, the scheduling strategy for the CCSP reported in this paper has shown excellent performance on real data. The 2–m constraint was initially expected to be hard to satisfy, but the conflicts were relatively easily resolved. The yearly harvest volume provided by the optimised schedules turned out to be remarkably even. The visual impact was considerably reduced, and the deviation from optimal harvest time was kept inside acceptable limits. In addition, the overall performance of the prototype in terms of speed proved to be satisfactory for experimental use.

However, more work is needed to further improve performance, both in terms of speed and schedule quality. Apart from further refinements of TS components, we intend to perform comparative experiments with alternative IIT meta–heuristics as well as "intelligent" backtracking search algorithms. These experiments will be conducted with the purpose of developing search strategies that are able to handle CCSPs with a number of regions which is an order of magnitude larger than in the case data described above.

In addition, further work is required to refine the underlying model from a forestry point of view. More realistic growth models are crucial for reliable results, and it would be desirable to include other types of treatment than clear-cutting, for example thinning and sparse cutting. Naturally, it is important to allow for a mixture of wood species in each stand. However, the most important challenge would probably be to provide mechanisms for modification of existing constraints and criteria and introduction of new ones.

To enhance the ECOPLAN prototype into an operational tool for forest managers, special attention has to be paid to develop a simple, intuitive, and efficient user interface. Finally, an "industrial strength" version should include flexible and efficient methods for integration with external information repositories and possible external services such as advanced growth simulation.

GIT and scheduling

For generalised versions of the CCSP, one might use a similar approach. Examples are area planning in local government, agricultural management, campaign planning in marketing and advertising etc. These application domains may be characterised by the term *Spatio-Temporal Decision Support*. This category encompasses problems which involve discrete sets of: Spatial objects, moments in time, and possible actions. In general, the task is to decide *when* to assign *what* kind of action to *which* spatial entity. The broadness and significance of applications of Spatio-Temporal Decision Support makes this field of research and development highly interesting and attractive, and it certainly deserves more attention in the years to come.

The extra efforts needed to establish and maintain the multi-disciplinary profile of the project proved to be highly rewarding, and a necessary condition to achieve our goals. The design and implementation of ECOPLAN strengthened our initial synergy thesis. The supply of well-known methods and techniques for management and analysis of spatial information is large and varying, and this is also the fact in constraint based planning. However, by

combining the technologies, we achieved results beyond expectations at a relatively modest cost.

6 Acknowledgements

We would like to thank NORSKOG for their supply of the Clear-Cut Scheduling Problem and access to real-life case data. We also are in debt to our fellow researchers Trond Vidar Stensby and Per Kristian Nilsen for their competent contributions in the implementation process of the ECOPLAN prototype. Finally we want to thank Torgrim Johan Castberg for permitting the use of the forest case data in the SINTEF research project, and Arne Løkketangen at Molde College for introducing us to Tabu Search and providing us with valuable comments and suggestions for improvement.

This work has been partially supported by the Norwegian Research Council, project MOI.31386, and SINTEF Informatics, project 3391 1000.

References

[ADH95] Erlend Arge, Morten Dæhlen, and Øyvind Hjelle. Mathematical Software for Terrain Modelling. In *this volume*, 1995.

[Dor95] Jürgen Dorn. Iterative Improvement Methods for Knowledge-Based Scheduling. *AI Communications*, 8(1):20 – 34, 1995.

[GJ79] M R Garey and D S Johnson. *Computers and Intractability.* Freeman, 79.

[Gje94] Jan Carsten Gjerløw. Transportplanlegging og GIT: Datasystemer for flåtestyring og ruteplanlegging. Technical report, SINTEF Informatics, 1994.

[Glo90] F Glover. Artificial Intelligence, Heuristic Frameworks and Tabu Search. *Managerial and Decision Economics*, 11:365–375, 90.

[Kir83] S Kirkpatrick. Optimization by Simulated Annealing. *Science 220*, 83.

[MJTVV92] R Mørk Jørgensen, H R Thomsen, and R V Valqui Vidal. The Afforestation Problem: A Heuristic Method Based on Simulated Annealing. *European Journal of Operational Research 56*, pages 184–191, 92.

[PK93] T Pukkala and J Kangas. A Heuristic Optimization Method for Forest Planning and Decision Making. *Scandinavian Journal of Forest Research*, 8:560–570, 93.

[sis95] SISCAT - The SINTEF Scattered Data Library (version 2.1). Technical report, SINTEF Informatics, Oslo, 1995. Reference manual.

[Tsa93] E Tsang. *Foundations of Constraint Satisfaction.* Harcourt Brace & Co, 93.

[WMMK94] G Weintraub, A Jones, A Magendzo, M Meacham, and M Kirkby. A Heuristic System to Solve Mixed Integer Forest Planning Models. *Operations Research*, 42(6):1010–1023, 11 94.