Universidade de Évora - Escola de Ciências e Tecnologia

Mestrado em Engenharia Informática

Dissertação

Models and Methods for the Modelling of Forest Managing

Eduardo Jorge Oliveira Eloy

Orientador(es) | Salvador Abreu

Vladimir Alekseevich Bushenkov

Évora 2023

Universidade de Évora - Escola de Ciências e Tecnologia

Mestrado em Engenharia Informática

Dissertação

Models and Methods for the Modelling of Forest Managing

Eduardo Jorge Oliveira Eloy

Orientador(es) | Salvador Abreu

Vladimir Alekseevich Bushenkov

Évora 2023

A dissertação foi objeto de apreciação e discussão pública pelo seguinte júri nomeado pelo Diretor da Escola de Ciências e Tecnologia:

Presidente	|	Lígia Maria Ferreira (Universidade de Évora)
Vogais	|	Rogério Ventura Lages dos Santos Reis (Universidade do Porto) (Arguente) Salvador Abreu (Universidade de Évora) (Orientador)

Évora 2023

Acknowledgements

This master's thesis was only possible due to important support coming from both people and institutions for which I must give some acknowledgements.

To Professor Salvador Abreu for being co-supervisor of this project as well as for sharing his knowledge of the technical aspects of Constraint Programming, for the criticisms, suggestions and recommendations as to the implementation and for the words of support throughout these 2 years.

To Professor Vladimir Bushenkov also for being co-supervisor of this project, for the availability and time whenever I had questions regarding the datasets, the overall direction of the project and the milestones achieved and for offering the material to review in this paper.

Fundação para a Ciência e a Tecnologia (FCT) is acknowledged for supporting the development of the system presented in this thesis, mostly through the BIOECOSYS and MODFIRE projects.

Sumário

A gestão florestal é uma actividade de grande valor económico e importância ecológica. As áreas florestais geridas podem abranger regiões muito grandes e a sua gestão adequada é muito importante para um desenvolvimento eficaz, tanto em termos de planeamento económico como de recursos naturais, e gerir uma floresta implica tipicamente a aplicação de escolhas políticas em diferentes parcelas de terra, aqui referidas como Stands ou Management Units (Unidades de Gestão).

Este documento analisa vários métodos de gestão florestal, juntamente com as suas variações disponíveis na literatura ao longo de 4 capítulos, sendo esses métodos o Unit Restriction Model and Area Restriction Model por Alan T. Murray, the Area Restriction Model with Stand-Clear-Cut variables por Constantino et al, the Path Algorithm and the Generalized Management Unit formulation por McDill et al and the Full Adjacent Unit formulation por Gharbi et al.

Os resultados apresentados nos artigos originais são discutidos nas conclusões de cada capítulo.

Os 2 últimos capítulos apresentam uma formulação de Constraint Programming do problema e a sua implementação utilizando a biblioteca Choco da linguagem de programação Java e apresentam também os resultados, um capítulo relacionado com a primeira implementação que trata apenas da optimização do Madeira Total Obtida e o outro alargando o problema, juntamente com um novo conjunto de dados, para lidar com a optimização multicritério. Para o fazer, os princípios de Constraint Programming são primeiro enumerados juntamente com uma breve história da tecnologia de Constraint Programming.

Finalmente, outros possíveis desenvolvimentos são discutidos numa secção de Trabalho Futuro.

Palavras chave: Constraints, Forest, MU, Stand, Solution

Abstract

Forest management is an activity of prime economic and ecological importance. Managed forest areas can span very large regions and their proper management is paramount to an effective development, in terms both of economic and natural resources planning and managing a forest typically implies applying policy choices to different patches of land, here referred to as Stands or Management Units.

This paper reviews several methods of forest management alongside their variations available in the literature throughout 4 chapters, those methods being the Unit Restriction Model and Area Restriction Model by Alan T. Murray, the Area Restriction Model with Stand-Clear-Cut variables by Constantino et al, the Path Algorithm and the Generalized Management Unit formulation by McDill et al and the Full Adjacent Unit formulation by Gharbi et al.

The results as presented in the original papers are discussed in the conclusions of each chapter.

The final 2 chapters present a Constraint Programming formulation of the problem and its implementation using the Choco framework of the Java programming language and showcases the results, one chapter relating to the first implementation that deals only with the optimization of total Wood Yield and the other broadening the problem, alongside a new dataset, to deal with multi-criteria optimization. In order to do this the principles of Constraint Programming are first enumerated along with a short history of Constraint Programming technology.

Finally further possible developments are discussed in a Future Work section.

Keywords: Constraints, Forest, MU, Stand, Solution

x

1

Introduction and Problem Statement

Forest management is a multi-faceted resource management and planning problem domain, one in which aspects such as the interests of multiple stakeholders, a diversity of underlying biological and physical models, economic performance and the impact of climate change in more than one sense, all combine to create complex combinatorial optimisation situations.

Traditional Operations Research techniques, relying on Mixed Integer Linear Programming (MILP), while effective at solving a problem once it has been modeled, remain very difficult from a technical point of view. Constraint Programming (CP) [RvBW06, Apt03] provides a paradigm whereby one may directly model a problem in terms of the entities which are pertinent thereto, together with the relevant relations which they must observe, in a fairly abstract and generic fashion. This specification is understood to be *executable*, in that it is sufficient for a *constraint solver* to efficiently search for a solution.

The thesis showcases different methods and models of forest management available in the literature as well as implementations developed by the author.

In 2020 with support and funding from the BIOECOSYS project the authors began developing a Constraint Programming implementation for the problem of Forest Management for the Vale de Sousa forest subject to area constraints, differing from previous approaches where the constraints are related to the

closure of adjacencies between management units (MUs), this approach constrains the transitive closure of adjacencies.

In this effort the authors were provided with all necessary information about the forest and the possible actions of management. An early version of the implementation consisting of mainly the model was showcased at the Dynamic Control and Optimization conference in February 3rd 2021, the complete and functional implementation was the subject of a paper delivered in time for the final BIOECOSYS conference during which the authors presented the results of the project[Elo].

1.1 Different Models and Methods

The use of forest resources is traditionally multifaceted. To ensure that these resources remain available for use by present and future generations, sustainable management practices are essential to balance the diverse and often competing demands of forest management. One way of planning management to minimise the impact of forestry activities is to include spatial constraints (or adjacency constraints) in the analysis of harvest schedules [BBSG17]. The objective function may be to maximise profit, net present value or other alternatives.

Forest management authorities often place restrictions on the size of harvest openings. At present, legal limits on clear-cut sizes in Portugal are set to 50 ha. Such restrictions can dramatically complicate forest planning.

Many researchers used Integer Programming or Mixed-Integer Programming models to model spatial forest planning problems (see for example [HJ93], [MRBB02], [GR05], [GRBC19]) where forests are represented as graphs, with MUs representing nodes and adjacencies between them representing edges. Decision variables usually correspond to harvesting blocks at a particular period of time. The Unit Restriction Model (URM) and Area Restriction Model (ARM) are the two main approaches to deal with adjacency in harvest scheduling models [Mur99]. In the URM approach, the boundaries of each potential cutting block are predefined; simultaneous harvesting is prohibited in two adjacent units. However, the ARM models allow for simultaneous harvesting of adjacent units, provided their combined area does not exceed the maximum allowable cut size [Mur99]. In the latter approach, harvest block boundaries are not predefined; instead, they are defined through models that determine all the potential harvesting blocks that satisfy a maximum allowable cut ([MRBB02],[GMVW09]). Formulating and solving ARM models is significantly more difficult than formulating and solving URM models [BK05]. Unlike URM models, ARM models are very flexible and generate more useful possibilities of better-performing harvesting plans. The first ARM formulations encompass an exponential number of variables or constraints. One integer programming ARM with a polynomial number of variables and constraints – called Area Restriction Model with Stand-Clear-Cut variables (ARMSC) – was proposed in [CMB08].

These models are typically implemented as very large mixed integer linear programming (MILP) problems that are difficult to solve. The branch and bound algorithm (BBA) is a general method of obtaining exact solutions to MILP problems [MB01, CMB08, GRBC19]. Until recently, however, only relatively small or medium problems could be solved with this algorithm. Considerable effort has therefore gone into developing alternative methods of solving harvest scheduling models with adjacency constraints, including heuristics, such as Monte Carlo integer programming [BB99], simulated annealing [BEB14], genetic algorithm [BB02], and dynamic programming [BHR99].

1.2 Spatial Restrictions in Harvest Scheduling

In a 1999 paper[Mur99], Alan T. Murray proposed 2 different model forms to solve the problem of forest management, the Unit Restriction Model and the Area Restriction Model.

Unit Restriction Model

The former, which might require an a priori treatment of the management area to delineate spatial units, assumes spatial units that are smaller than the maximum area restriction so that at any point harvesting 2 adjacent units would break the maximum area restriction, meaning the formulation is based on harvesting each unit only once throughout the time horizon, never harvesting 2 adjacent units at the same time, while maximizing some criteria and keeping another criteria inside an upper and lower bound. The following formulation uses the symbols in Appendix A.1 and A.2

$$max \sum_t \sum_i p_i^t x_i^t. \tag{1}$$

$$subject\ to \sum_t x_i^t \leq 1 \quad \forall i. \tag{2}$$

$$\sum_i C_i^t x_i^t \geq LB_t \quad \forall i. \tag{3}$$

$$\sum_i C_i^t x_i^t \leq UB_t \quad \forall i. \tag{4}$$

$$x_i^t + x_j^t \leq 1 \quad \forall i, t, j \in N_i. \tag{5}$$

$$x_i^t = (0, 1) \quad \forall i, t. \tag{6}$$

Area Restriction Model

The latter, which forms the basis of various models in this paper, does not assume spatial units that are smaller than the maximum area restriction, in this formulation harvesting adjacent units are feasible when the total contiguous area of harvested units is below the maximum area restriction. The formulation is as follows and uses the symbols in Appendix A.1 and A.2:

$$max \sum_t \sum_i p_i^t x_i^t. \tag{7}$$

$$subject\ to \quad (2) - (4), (6)$$

$$f(x_i^t) \leq A_{max} \quad \forall i, t. \tag{8}$$

While similar to the URM this formulation replaces expression (5) with a recursive function which ensures that, starting from unit "i", the contiguous area harvested does not exceed the limit. The f function takes the following form:

$$x_i^t \sum_{j \in N_i \cup S_i} a_i\ x_j^t \leq A_{max}$$

Where S_i is a subset of harvested units that contains all of unit i's adjacent units which are being harvested at period t, and all units adjacent to those units which are also being harvested and so on, resulting in a group of units which form a cluster that is being harvested at the same time so the area constraint can be enforced. As this is enforced for each management unit i in each period t there occurs a lot of constraint redundancy.

Of note is that under certain conditions the URM and ARM specify and solve the same problem, that is when none of the spatial units have an area larger than the maximum limit and the smallest spatial unit is larger than half of the maximum limit.

Allowing for constraint violations

In the paper an extension is introduced in order to relax some of the constraints, that is to allow for the maximum area constraint to be violated but for these occasions to be minimized. The formulation replaces expressions (7) and (8) with the following

$$max \sum_t \sum_i p_i^t x_i^t - \sum_i O_i. \tag{10}$$

$$f(x_i^t) - O_i \leq A \quad \forall i, t. \tag{11}$$

Where O_i is the amount by which the contiguous harvest that starts at unit i violates the maximum area limit, so if the limit is 50ha and the contiguous harvested area is 60 O_i would be 10.

1.3 Structure of this Document

The thesis is structured as follows: in chapters 2 through 4 we cover the current state of the art, including Mixed Integer Linear Programming and other approaches to solving the forest planning problem. Chapter 5 discusses our constraint-based formulation of the problem and the new propagator. In chapter 6 we discuss the issue of having multi-criteria optimization over the same base technology. Finally, in chapter 7 we evaluate the work and point to possible lines for further development.

2

Mixed Integer Linear Programming Model

In this chapter we briefly recap the work presented in [CMB08].

2.1 Introduction

In a 2008 paper [CMB08] Constantino et al presented a new Mixed-Integer Programming formulation for the problem of forest management and scheduling along with 2 other formulations available in the literature. This formulation differed from previous approaches as it didn't require the user to define a priori regions or clusters of Management Units and instead left it to the model to choose which MUs to group up for an action. The problem of forest management in this context had the goal of maximizing the timber net present value in a long term planning horizon where each MU could only be harvested once, being subjected, of course, to maximum harvest area constraints. Each of these formulations is considered an Area Restriction Model as per defined in Alan T. Murray's Spatial Restrictions in Harvest Scheduling which is the subject of the previous chapter.

2.2 Area Restriction Model Formulations

The following formulations use the symbols present in Appendix A.1 as well as the following:

- v^t is the volume of timber harvest in period t.
- Constraint A1 states that the area of each clear-cut cannot exceed A_{max}
- Constraint A2 states that each stand can only be harvested once in T
- Constraint A3 states that periodic fluctuations cannot exceed a percent value of v^t, $|v^{t+1} - v^t| \leq \Delta v^t$, this constraint will only be applied to the new formulation presented in this paper.

2.2.1 Formulation with Stand Variables and Explicit Area Restriction Constraints

This formulation proposes that management options be represented by binary variables x_i^t which take the value of 1 if MU i is to be harvested during period t and 0 otherwise. The user then has to specify $\mathscr{R}$, which is all the possible clusters with a total size greater than A_{max} that don't contain a similar cluster. The formulation is as follows:

$$max \sum_{t \in T} \sum_{i \in V} p_i^t x_i^t \tag{1}$$

$$subject\ to \sum_{t \in T} x_i^t \leq 1 \quad \forall i \in V, \tag{2}$$

$$\sum_{i \in R} x_i^t \leq |R| - 1 \quad \forall R \in \mathscr{R}\ and\ t \in T, \tag{3}$$

$$x_i^t \in \{0, 1\} \quad \forall i \in V\ and\ t \in T. \tag{4}$$

Expression (1) maximizes the timber yield, expression (2) ensures the A2 constraint while (3) ensures the A1 constraint. Expression (4) ensures the binary value of variables.

The main drawback of this formulation is number of constraints, which grows exponentially with the number of MUs.

2.2.2 Formulation with Cluster Variables

In this formulation the user has to specify all the possible clusters with area not exceeding A_{max}, represented as $\mathscr{F}$. The decision variables represent feasible clear-cuts, defined as z_F^t taking the value of 1 if cluster $F \in \mathscr{F}$ is harvested in period t and 0 otherwise. The formulation is as follows:

$$max \sum_{t \in T} \sum_{F \in \mathscr{F}} p_F^t z_F^t \tag{5}$$

$$subject\ to \sum_{t} \sum_{F : i \in F} z_F^t \leq 1 \quad \forall i \in V, \tag{6}$$

$$\sum_{F \cap \{i,j\} \neq \emptyset} z_F^t \leq 1 \quad \forall \{i, j\} \in E\ and\ t \in T, \tag{7}$$

$$z_F^t \in \{0, 1\} \quad \forall F \in \mathscr{F}\ and\ t \in T. \tag{8}$$

Similar to the previous formulation the 4 expressions represent the timber yield, the A2 constraint, the A1 constraint and the binary requirement of variables. The formulation can be strengthened by replacing expression (7) and using "cliques", as this is also done in the next model it will be explained there.

The main drawback of this model is the large number of variables which grows exponentially with the number of MUs.

2.2.3 Formulation with Stand-Clear-Cut Variables

In this new formulation the number of variables and constraints are bounded polynomially to the number of total MUs. The clear-cuts are grouped by the model into sets $C = \{\mathscr{C}_1....\mathscr{C}_{NS}\}$, where some sets $\mathscr{C}_i$ can be empty, so the user doesn't have to define any regions a priori. The decision variables y_j^{it} take the value of 1 if MU j belongs to set $\mathscr{C}_i$ in period t and 0 otherwise. To express the area restriction constraint the w_e^{it} variables are introduced, which take the value of 1 if at least one of the MUs in edge e belongs to set $\mathscr{C}_i$ in period t and 0 otherwise.

$$max \sum_{t\in T}\sum_{j\in V} p_j^t \sum_{i\in V} y_j^{it} \tag{9}$$

$$subject\,to\;\; y_j^{it} - w_e^{it} \leq 0 \quad \forall e \in E,\; j \in e,\; i \in V,\; and\; i \in T, \tag{10}$$

$$\sum_{i\in V} w_e^{it} \leq 1 \quad \forall e \in E\; and\; t \in T, \tag{11}$$

$$\sum_{t\in T}\sum_{j\in V} a_j y_j^{it} \leq A_{max} \quad \forall i \in V, \tag{12}$$

$$\sum_{t\in T}\sum_{i\in V} y_j^{it} \leq 1 \quad \forall j \in V, \tag{13}$$

$$y_j^{it} \in \{0,1\} \quad \forall j \in V,\; i \in V,\; and\; t \in T, \tag{14}$$

$$w_e^{it} \geq 0 \quad \forall e \in E,\; i \in V,\; and\; t \in T. \tag{15}$$

Expression (9) states that the timber harvest should be maximized. Expression (10) expresses how whenever a MU belonging to a set is being harvested there's a corresponding edge where at least one of the MUs belonging to it, which belong to that same set, are being harvested. Expressions (11) and (12) enforce the A1 constraint by stating that in each period every 2 adjacent MUs are in 1 clear-cut at most and that any clear-cut does not exceed A_{max}. Expression (13) enforces the A2 constraint. This is the first step of the new proposed formulation.

However this results in a large number of equivalent solutions because there isn't a constraint enforcing the connectivity of sets $\mathscr{C}_i$, so any solution with a disconnected harvest set, say a hypothetical MU 1 and 7 belonging to the same set $\mathscr{C}_1$ = 1,7, is equivalent to the solution where $\mathscr{C}_i$ is replaced by its connected component, say $\mathscr{C}_1 = 1$ and $\mathscr{C}_7$=7. To solve this, the authors further develop the formulation so that any set $\mathscr{C}_i$ which contains MU i does not contain MU j where j<i, for this expression (10) is removed. Furthermore because $y_j^{it} = 0$ whenever $y_i^{it} = 0$ and due to the presence of expression (13), expression (12) can be replaced by (12').

$$\sum_{j \geq i} a_j y_j^{it} \leq A_{max} \times y_i^{it} \quad \forall i \in V, \; and \, t \in T, \tag{12'}$$

$$y_j^{it} - y_i^{it} \leq 0 \quad \forall i, \; j \in V, \; j \geq i, \; and \, t \in T, \tag{16}$$

$$w_e^{it} - y_i^{it} \leq 0 \quad \forall i \in V, \; e = \{j_1, j_2\} \in E, \; with \; i \leq j_1 \; or \; i \leq j_2 \; and \, t \in T. \tag{17}$$

This formulation composed by expressions (9)-(11), (12') and (13)-(17) is referred to as ARMSC, it has $O(NS^2{\times}NT)$ variables and constraints since the graphs representing forests are nearly always planar, that is they can be drawn without any edges intersecting, for non-planar graph forests the number of variables and constraints is $O(NS{\times}NE{\times}NT)$.

To ensure that in each period the volume of timber harvested is within a percent of the timber harvested in the previous period the following expression is introduced:

$$(1-\Delta) \sum_{j \in V} V_j^{t-1} \left(\sum_{i \leq j} y_j^{i,t-1} \right) \leq \sum_{j \in V} V_j^{t} \left(\sum_{i \leq j} y_j^{it} \right) \leq (1+\Delta) \sum_{j \in V} V_j^{t-1} \left(\sum_{i \leq j} y_j^{i,t-1} \right) \quad \forall t = 2, ..., NT. \tag{18}$$

The ARMSC formulation can be further modified to remove unnecessary variables.

ARMSC with Cliques

A large number of y_j^{it} variables will take the value of 0 in any valid solution, this happens for any clear-cut which meets the following conditions:

- Clear-cut is composed of $\{i_1 ....i_p\}$ while $i_1 = i$ and $i_p = j$,
- $\{i_k, i_{k+1}\} \in$ E
- $i_k > i$ with $1 < k \leq p$
- sum of the areas in the shortest chain between i_1 and i_p is larger than A_{max}

Therefore these $y_j^{it} = 0$ variables may be removed for all the planning horizon. In the case that $y_j^{it} = 0$ then the corresponding w_e^{it} variables may also be removed as they represent edges with no nodes in the set $\mathscr{C}_i$.

This is done by introducing cliques, in graph theory a clique is a complete subgraph of G, meaning a subgraph of G where every pair of distinct vertices is connected by a unique edge. The set Q is introduced which represents the maximal cliques of G, with a maximal clique being a clique that cannot be extended just by adding an adjacent node. With this the w_e^{it} variables are replaced by W_P^{it}, which take the value of 1 if at least one of the MUs in clique P belong to $\mathscr{C}_i$ in period t and 0 otherwise.

So the expressions (10) and (11) are replaced by:

$$y_j^{it} - W_P^{it} \leq 0 \quad \forall P \in Q,\ j \in P,\ i \leq j\ and\ t \in T, \tag{10'}$$

$$\sum_{i \in V} W_P^{it} \leq 1 \quad \forall P \in Q\ and\ t \in T. \tag{11'}$$

2.3 Experimental Evaluation

CPLEX 9.0 was used, with default parameters, as both a linear and integer programming solver and the branch-and-bound (BaB) algorithm was allowed to run for a maximum of 2 hours. The tests were run on a desktop computer with an Intel Pentium M-1.6 GHz processor and with 512 MB RAM.

The objectives were to assess variations in the results due to variations in the model and to obtain solutions within less than 1% deviation from the best upper bound found by the branch-and-bound algorithm which is given by CPLEX, the integer solutions are also compared to their linear relaxations to further assess the strength of the formulation.

2.3.1 Branch and Bound Algorithm

The BaB algorithm was designed to solve discrete and combinatorial optimization problems, in this project it was used by CPLEX to find a solution which maximizes wood yield while subjected to the area constraints, since BaB technically is only able to minimize the value of an objective function CPLEX converts this to a minimization problem. Essentially the algorithm splits, or branches, the search space into smaller spaces that minimize the objective function, which will then be tested for feasible solutions. Throughout the process the algorithm keeps track of the minimum upper bound (for the objective function) seen among all examined solutions, and if the solution being searched is determined to have a value greater than the bound value it's removed from the search space.

2.3.2 Results

For this both real forests and hypothetical forests were used, each with different defining characteristics and parameters such as number and size of MUs, age class, minimum cutting age, Δ (for the A3 constraint) and others. The ratio between A_{max} and the average area of the MUs in the forest, $\bar{a}$, affects the size of the formulation, values between 3 and 6 were considered.

The authors presented the results for the tests done using the new model and it's modified versions: ARMSC (no volume flow constraints, no clique variables); ARMSC-C (with clique variables W_P^{it} instead of edge variables w_e^{it}); ARMSCV-C (with volume flow constraints and clique variables); and ARMSCV-C with constraints (12) replacing constraints (12').

For the ARMSC, solutions were found within a reasonable time frame for most of the forest except 3 of the largest ones,the ARMSC-C formulation was effective in 2 of these cases where the corresponding linear relaxation is tighter than the linear relaxation of the ARMSC as was the case with other instances. In the case of the ARMSCV-C, the algorithm was able to find a solution for all instances tested before. The optimum linear relaxation solution is very close to the best integer solution obtained for most instances. In some instances where the value of A_{max} was increased the algorithm was still able to find a solution within or slightly above 1% of the optimum.

The authors state that using either (12) or (12') likely isn't significant to the results.

Overall the results show that the model finds solutions within 1% of the optimal solution in all cases where the average number of MUs per clear-cut ranges in the interval [3, 4]

2.3.3 Conclusions

In the paper's conclusions the authors state that their model presented a step forward in expanding the concept of adjacency in forest management spatial objectives but computational constraints remain an obstacle to solve complex ARM problems. Its polynomial number of variables and constraints are also an efficiency improvement on current models which have either an exponential number of variables or constraints.

3

Path algorithm and GMU

3.1 Introduction

In a 2002 paper[MRBB02] McDill et al proposed two new formulations of MILP ARMs to solve harvest scheduling problems. The first is called the Path Algorithm which only imposes adjacency restrictions to prevent the area limit being broken and the second is the Generalized Management Units (GMU) Formulation which uses variables representing combinations of adjacent management units whose combined areas don't break the area limit, this approach allows for harvesting adjacent units together when it's known in advance that there's some kind of benefit to doing so.

3.2 Path Algorithm

This method essentially works by creating "paths", meaning groups of adjacent MUs that break the area limit when combined and applying a constraint which determines that if there are N number of MUs in that group then only N-1 MUs can be harvested at the same time, this is called the Path constraint. For

example, using some symbols that have already been established, if the MUs A, B and C have a combined area above the area limit this constraint would be applied: $x_A^t + x_B^t + x_C^t \leq 2$

Put more formally the algorithm is as follows:

1. Start with an arbitrary pair of adjacent MUs, if their combined area exceeds the limit then impose a Path constraint if not then this is now the current cluster of processed MUs
2. Select an MU adjacent to the current cluster and add it to the cluster.
3. Based on the current cluster define a network, with nodes corresponding to each MU and arcs representing adjacencies. Identify each possible path originating from the node just added to the cluster and terminate paths when either the accumulated area of the MUs in it exceed the limit or when there aren't any more MUs in the cluster that are adjacent to the path but not already part of it. Paths can follow multiple branches.
4. Identify if the newly created path is redundant, meaning if the set of MUs in any previously identified path are a subset of the set of MUs in the newly created one, if the path is not redundant add a set of Path constraints, for each period, corresponding to the set of MUs defined in this path. Removing redundant paths serve to keep area limit from being broken.
5. Stop if the cluster contains all the MUs in the forest, if not then return to step 2.

The following formulation uses the symbols presented in Appendix A.1 and A.3

$$max \sum_{i=1}^{V} \sum_{t=1}^{T} p_i^t a_i x_i^t \tag{1}$$

$$subject\ to \sum_{t=1}^{T} x_i^t \leq 1 \quad for\ \ i = 1, 2, ..., M \tag{2}$$

$$\sum_{i=1}^{V} v_i^t x_i^t - v^t = 0 \quad for\ \ t = 1, 2, ..., T \tag{3}$$

$$b_{lt} v^t - v^{t+1} \leq 0 \tag{4}$$

$$-b_{ht} v^t + v^{t+1} \leq 0 \tag{5}$$

$$\sum_{j \in P_i} x_j^t \leq N_{Pi} - 1 for\ \ P_i\ \ and\ \ t = 1, 2..., T \tag{6}$$

$$\sum_{i=1}^{V} \sum_{t=1}^{T} (Age_i^t - Age^T) a_i x_i^t \tag{7}$$

$$x_i^t \in 0, 1 for\ \ i = 1, 2, ...V and\ \ t = 0, 1, 2, ...T. \tag{8}$$

Expression (1) maximizes the objective function of the problem (timber yield), expression (2) states that MUs can at most be harvested once, expression (3) sums up the harvest volume for each period and assigns the value to the variables v^t, expressions (4) and (5) are flow constraints, they enforce an upper and lower bound on the rate of harvest. Expression (6) represents the adjacency constraints generated with the Path algorithm, expression (7) enforces the average age of the forest area at the end of the planning horizon to be at least Age^T which prevents the model from overharvesting the forest. Finally expression (8) enforces the binary value of constraint variables x_i^t.

3.3 The Generalized Management Units Approach

This method introduces Generalized Management Units, these units correspond to feasible combinations of adjacent MUs, with the condition being that the sum of the areas of the MUs inside a GMU don't exceed the area limit, so actions taken upon a GMU would be applied to all the MUs composing it. These GMUs could be defined a priori but the idea is that it should be the model deciding when to create these combinations.

To formulate the ARM with the GMU approach the previously defined expressions (2) and (6) must be replaced with the following expressions:

- V_G is the set of all GMUs.
- G_i is the set of GMUs in V_G that include the MU i.
- C_j is the set of indexes corresponding to the jth pair of adjacent management units.

$$\sum_{u \in G_m} \sum_{t=0}^{T} x_u^t \leq 1 \quad for \;\; m = 1, 2, ..., V_G \tag{9}$$

$$\sum_{m \in C_i} x_m^t \leq 1 \quad \forall C_j and t = 1, 2, ..., T. \tag{10}$$

This approach can produce ARM formulations prohibitively large to solve although some factors moderate this complexity such as the fact that it deals with clique constraints, meaning lists of variables where it's stated that only 1 can take a non-zero value, which reduces the complexity of these models.

According to the authors an optimal solution to this ARM "will always be at least as good as the optimal solution to the (equivalent) URM", therefore a sub-optimal solution achieved in less time through terminating the branch and bound algorithm early could be as good if not better than the equivalent URM.

The authors state that relative to the Path formulation this approach is inefficient however it's still important for providing an alternative formulation for ARM problems and may prove useful in specific applications, for example when it's known in advance that there are direct cost savings in managing certain MUs jointly instead of separately.

3.4 Experimental Evaluation

The tests were carried out on 2 example forests created using Makeland[MJ00], one with 50 MUs and the other with 80 MUs, in both forest the average area of a MU is 20ha with the largest and smallest MUs in the smaller forest having 32.6ha and 10.5ha respectively, and for the larger forest, analogously, 39.1ha and 10.7ha. MUs are assumed to be of the same forest type but of different ages. All solutions where obtained using CPLEX V6.5 on a 500 MHz Pentium III computer with 384 Mb of RAM.

To eliminate solution time disparities each problem was solved 5 times and the averaged solution time for each problem was presented.

For testing each formulation 3 key parameters were considered:

- Maximum harvest area: Problems were created using maximum harvest areas of 40ha, since this is larger than the largest MUs in both forests, and 48.6ha which is "the maximum harvest area specified by the AFPA's Sustainable Forestry Initiative (AF&PA 2000)". The larger this parameter the more possible combinations of MUs that can be harvested concurrently, therefore the models will be larger, this can also be said about smaller MUs.
- Maximum age-class difference among contiguous stands: The closer MUs are in age the more likely they are to be harvested concurrently, increasing the number of possible combinations and therefore the complexity. The authors varied this parameter from 0 to 4 to test for potential loses in objective function values or time savings for limiting the age differences (0 means all MUs are of the same age-class).
- Maximum number of MUs that can be harvested simultaneously: This corresponds to the maximum number of MUs allowed in a Path, if this values exceeds 4 the authors state that implementing the algorithm would be considerably more complex. One of the ways to deal with this is, for example, not to define MUs in the forest with sizes larger than one fourth of the total area limit and setting the maximum number of MUs in a path to 4, however this is very restrictive. It also corresponds to the maximum number of MUs that can compose a GMU, a value of 4 would allow a maximum o 3 stands in a GMU, and a higher value means a larger sized problem.

In the smaller forest there were no groups of 3 contiguous MUs with a combined area less than 40ha and in the larger forest there were only a few. With these parameters in mind, 3 test cases considered for each forest.

- Case A: 40ha maximum harvest area and maximum of 2 adjacent MUs harvested concurrently.
- Case B: 48.6ha maximum harvest area and maximum of 2 adjacent MUs harvested concurrently.
- Case C: 48.6ha maximum harvest area and maximum of 3 adjacent MUs harvested concurrently.

As stated before, the maximum age-class difference ranged between zero and four for each case.

3.4.1 Results

The solutions obtained were the same for the GMU method and the Path method, even when the Path algorithm was started with 10 different initial pairs of MUs, however model size was different and affected by the different parameters.

The number of variables in the Path formulation aren't affected by the parameters but the number of constraints is and tends to increase in the manner already described. Compared to the Path algorithm the GMU formulations always resulted in substantially larger problems in number of variables and constraints, this was reflected in solution times which increase with the number of variables and constraints. Solution times often decreased as maximum age difference was increased although this lead to larger models.

The ARM models obtained better solutions in terms of objective function values, using case A (40ha area limit) with the smaller and larger forests ARM solutions were respectively 3.2% and 1.2% better than the equivalent URM model solutions, with a 48.6ha area limit the improvement was 4.8% and 2.7% respectively. These gains are significant when taking into account the reduction in the cost of adjacency constraints (or cost per hectare) when compared to the URM formulation, these reductions range from 11.5% to 74.7%.

The results show that having large age differences tended to increase the model size while not providing relevant benefits, the largest improvement when manipulating this parameter was the 0.06% improvement gained from allowing a maximum age-class difference of 2 instead of 1.

With the examples tested there was little to be gained from allowing a maximum of 3 contiguous MUs to be harvested simultaneously, as stated previously there weren't many of these groups with areas less than 40ha so this parameter was only relevant to the larger forest using case C, even so these groups were not scheduled to be harvested simultaneously in the best solution obtained.

The authors point to a potential drawback to the ARM approach where they posit that smaller MUs will tend to be disregarded as they will often just be harvested alongside larger adjacent MUs, reducing MU size diversity. A potential solution to this is to add "constraints requiring minimum numbers of harvests in different ranges of size".

As stated earlier the GMU formulation can recognize direct cost savings obtained from combining MUs for higher timber sales, tests were carried out on GMU and Path models which included a fixed cost of 1,500$ associated with a timber sale so groups of adjacent MUs harvested simultaneously only incur this cost once, the models were based on the small forest using case B.

The Path formulation as it stands can't recognize cost savings from combining MUs so the solution obtained from this model was the same as the previous test, although the objective function value is different due to the fixed costs incurred from timber sales, it scheduled 8 pairs of adjacent MUs to be harvested simultaneously. The GMU formulation does recognize these costs a, it scheduled 11 pairs of adjacent MUs to be harvested simultaneously, even so the objective value function was only 0.04% better than the Path formulation, saving 1,140$ over a 60 year planning horizon. The authors state that despite the improvement being small in this case the GMU formulation may be useful in other situations.

3.4.2 Conclusions

In the paper's conclusions the authors reiterate some of the points already stated about the results, such as the effects of the parameters on the ARM formulations. The GMU formulation is proposed have more potential than what's immediately obvious from the results since it offers solutions that more closely correspond to real world problems and constraints. It also may prove useful as a base for heuristic solutions algorithms or for modeling specific cases such as interior space or forest patch size distributions.

It's also possible to define a priori constraints for which MUs should be treated jointly instead of separately and the results suggest that considering the simultaneous harvest of MUs that are more than 2 age-classes apart is generally negligible.

4

Full Adjacent Unit Formulation

4.1 Introduction

In a 2019 paper[GRBC19] Gharbi et al presented a new Mixed Integer Programming (MIP) formulation for the problem of forest management, the Full Adjacent Unit formulation. Taking into account the relevant literature at the time this new formulation has both a URM based form and an ARM based form, in either form it essentially works by prohibiting the simultaneous harvest of an MU and its adjacent MUs when these are not contained in the same "harvesting unit", so it focuses mainly on satisfying adjacency constraints between MUs, this is defined as the FAU constraint. The authors state that this formulation is simpler than the previously discussed models as, for its ARM form, it only requires an a priori enumeration of feasible clusters of MUs and not other, more complex, sophisticated sets of MUs, unlike other ARM models discussed in this paper the maximum area constraint is enforced by the a priori enumeration and not by any expression in the formulation. The URM form doesn't require any a priori enumeration of feasible clusters.

When analyzing the results of applying this formulation to an example forest particular focus is drawn to aspects such as structure, number of MUs, mean area of stands and so on, so tests were carried out on both large and small forest cases.

4.2 FAU Formulation

While there's a key difference in the formulation for the URM and ARM approaches the general model is defined as:

$$max \sum_{t\in T}\sum_{k\in K} p_k^t x_k^t \tag{1}$$

$$subject\ to \sum_{t\in T}\sum_{k\in K} x_i^t \leq 1 \quad \forall k \in K, \tag{2}$$

$$\sum_{k\in K_i} x_k^t + (\frac{1}{B_i}) \sum_{k´\in K_{i´},\, k´\notin K_i,\, i´\in N_i} x_k^t \leq 1 \quad \forall i \in V, \forall t \in T \tag{3}$$

$$x_k^t \in \{0,1\} \quad \forall k \in K, \forall t \in T \tag{4}$$

This formulation uses the symbols in Appendix A.1 and A.4.

As in previous models expression (1) maximizes the value and (2) guarantees that each MU is harvested at most once. Expression (3) is the FAU constraint which ensures that no two adjacent MUs or clusters of MUs are harvested simultaneously ($x * \frac{1}{B_i}$ will always result in a positive non-zero number if any MU adjacent to i is being harvested). (4) ensures the binary value of variables.

Some additional constraints may be implemented.

- L_t is the ratio of lower bound harvesting volume (timber) in period t.
- U_t is the ratio of upper bound harvesting volume (timber) in period t.
- m is the number of years in a period.
- g_i is the initial age of MU i.
- G^- is the mean age for the forest.

$$L_t \sum_{k\in K} v_k^t x_k^t \leq \sum_{k\in K} v_k^{t+1} x_k^{t+1} \leq U_t \sum_{kinK} v_k^t x_k^t \quad \forall t = 1..(nt-1) \tag{5}$$

$$\sum_{i\in V} a_i \left[g_i + ntm - \sum_{t\in T}(tm + g_i) \sum_{k\in K_i} x_k^t \right] \geq G^- \sum_{i\in V} a_i \tag{6}$$

Expression (5) enforces the ratio of the harvested volume to be between a lower bound and an upper bound and (6) enforces a mean ending age for the MUs. This is the FAU_1 formulation, however, it may be strengthened.

When the forest in question has many MUs tightly packed together, the number of adjacent MUs of each MU, B_i, becomes large. If B_i represented instead the number of maximal cliques containing MU i the formulation could become tighter while the adjacency constraint for MU i still holds. So the following expression (3´) replaces expression (3), this is the FAU_2 formulation.

- $B´_i$ is the number of maximal cliques which include MU i.

$$\sum_{k \in K_i} x_k^t + (\frac{1}{B^{'}{}_i}) \sum_{k^{'} \in K_{i^{'}},\, k^{'} \notin K_i,\, i^{'} \in N_i} x_{k^{'}}^t \leq 1 \quad \forall i \in V,\, \forall t \in T \tag{3'}$$

Another possibility to further strengthen the formulation is if B_i represented the maximum number of MUs adjacent to i which are not mutually adjacent since this number might be smaller than the number of maximal cliques containing i. So expression (3´´) replaces (3), this is called FAU_3

- $B^{''}{}_i$ is the maximum number of MUs adjacent to i which are not mutually adjacent

$$\sum_{k \in K_i} x_k^t + (\frac{1}{B^{''}{}_i}) \sum_{k^{'} \in K_{i^{'}},\, k^{'} \notin K_i,\, i^{'} \in N_i} x_{k^{'}}^t \leq 1 \quad \forall i \in V, \forall t \in T \tag{3"}$$

The authors emphasize the linear relation between the number of constraints and the number of MUs in this model and the chances of the problem being computationally tractable provided the number of MUs in a feasible cluster isn't too large.

4.3 Experimental Evaluation

The tests were carried out on both small and large cases of forest management as well as real and hypothetical forests, URM tests where run on an Intel Xeon processor at 3.07 GHz frequency with 8 GB of RAM using Windows 7 while ARM tests using large cases were run on an Intel Xeon processor at 2.6 GHz frequency with 43 GB of RAM using Windows 7. CPLEX 12.5 was used with mostly default parameters and all programs were written in A Mathematical Programming Language (AMPL). The purpose of the tests was to evaluate the preprocessing effort of generating feasible clusters, the quality of the formulation with attention payed to the number of constraints and the quality of solution within a time limit of 10000s. For comparisons with other formulations available in the literature the authors also ran tests using the Maximal Clique-Cluster formulation(MCC), which is based on clusters of adjacent MUs that can't be enlarged without breaking the area limit when harvested at the same time, and the already discussed Path formulation, which required the generation of maximal cliques and minimal infeasible clusters (clusters that slightly exceed the area limit) respectively. The generation of feasible clusters was done using a custom algorithm and was generally a quick endeavor while generating minimal infeasible clusters required several hours for the large cases.

4.4 Results

4.4.1 Comparing problem size

Generally, in both real forests and hypothetical square-shaped forests, the FAU formulations result in the smallest number of constraints, followed by the MCC formulations. In both formulations the number of constraints grows linearly with the number of MUs, however the MCC formulation is sensitive to the structure and size of the forest so it can result in a large number of constraints in certain cases, thus the FAU is judged to have more advantages in regards to problem size. The Path formulation for its part results in the smallest number of variables but a far larger number of constraints which grows exponentially.

When testing the effects of different values for the maximum area restriction on the three formulations the

authors find that while the number of variables will increase the number of constraints won't be heavily affected.

4.4.2 Comparing solution quality

For evaluating solution quality the authors use the time it took to find the optimal solution, if the best feasible solution found is not the optimal solution the optimality gap (computed as $\frac{|best\ node - best\ integer|}{|best\ integer|} * 100$) given by CPLEX is used. 3 values for the planning horizon were used during the tests, those values were 1, 3 and 5.

For testing the small cases 3 variants for the problem were considered, variant 1 assumed the adjacency and single harvest constraint, variant 2 adds the volume constraints, expression (5), and variant 3 adds the mean ending age constraint, expression (6). The large cases were only tested with variant 1.

Small cases

The ARM based approach was used.

In several cases optimal solutions were found with the FAU models, especially FAU3, and in cases where the solutions found were not optimal the optimality gap was within 1%

For the two smallest forest the authors recommend the use of the MCC formulations as this one achieved better results for all variants, however generally the performance of the MCC and FAU formulations was close. The Path formulation also performed well.

Large cases

The large cases were tested for the MCC and FAU formulations in both their URM and ARM approaches on a set of 8 large forests in Quebec, Canada.

In the URM approach both formulations found optimal solutions when the planning horizon was 1 (T=1) for all large cases, however with the available memory the MCC formulation did not find feasible solutions for some of the large cases when 3 or 5 time periods were considered. Optimality gaps for the FAU ranged from 0.1% for some of the smaller cases to 13% for the larger cases where the planned horizon was 3 periods. For the MCC optimality gaps were shorter, reaching a maximum of 2% however these were a bit slower to achieve than for the FAU formulation.

In the ARM approach problems were much larger, the FAU formulation never found optimal solutions in any case and in the case of the 3 largest cases when considering 5 periods the solver returned a "ran out of memory" status, feasible solutions achieved optimality gaps ranging from 1% to 17% (with one anomaly reaching 39 919%). The MCC initially seems to fare better since it achieved optimal solutions in most cases where a single period was considered and in the other ones achieved feasible solutions, however half of these optimal solutions had optimality gaps ranging from 9 000% to 41 000% making the FAU formulation more convincing. For the largest forest considering 5 period the solver ran out of memory.

Table 4.1: Total time to solve the URM problem with one period

Case	No. of stand	FAU time (s)	MCC time (s)
04251_1	10 282	4.16	333.92
04251_2	20 736	8.10	849.82
04251_3	31 514	14.56	2 920.71
04251_4	42 374	22.44	6 698.74
04251_5	52 819	32.32	10 425.37
04251_6	63 275	40.90	15 292.30
04251_7	73 782	101.10	17 697.90
04251_8	83 826	139.00	28 009.60

4.4.3 Conclusions

The authors conclude that the proposed FAU formulation performs well against the MCC and Path formulations in several aspects, one of which is time and simplicity, as shown in table 4.1 , the FAU formulation achieves the smallest computational time. Another advantage is that not needing to generate complicated sets of MUs and relying simply on feasible clusters cuts down on time in a major way as well as possible inconsistencies, minimal infeasible clusters took a significantly amount of time to build in the large cases (10 hours for the smallest large case).

The FAU formulation is also robust against fluctuations in forest size and structure, usually being more sensitive to the number of MUs instead while the MCC formulation ended up with a far larger number of constraints as these factors increased.

While no method consistently outperformed the other at solving integer problems in most cases the FAU formulation achieved optimal solutions or feasible solutions with low optimally gaps while the MCC formulation left large gaps in certain cases.

5

Choco-Solver implementation for Vale de Sousa

Most of this chapter was published as [EBA22] in the Dynamic Control and Optimization conference. The work was carried out in the context of the FCT grant PTDC/ASP-SIL/30391/2017 (BIO-ECOSYS.)

5.1 Modeling Forest Management

A Forest Management problem over a geographical map consists of:

- A set of *management units*, or *stands*, which represent contiguous pieces of land. A stand has fixed and time-dependent attributes. In the former case we include surface area and adjacency information (w.r.t. other stands.) The latter includes the species which is planted and the management action which is planned at a given time.
- A set of *prescriptions* which apply to the stands. These are actions to be carried out in the context of each stand, in a given point in time, and they include *harvesting* (removing all the trees), *thinning* (cutting off some wood but leaving the trees in place) or just doing nothing.

The problem we are aiming to solve entails selecting a prescription for each stand in the map, over the entire planning period. This goal is further qualified by a criterion which is to be optimised for, for instance maximum total wood output.

We are given as input a set P of *prescription definitions* which describe a sequence of time-indexed actions to be applied to each management unit (u_i.) Each prescription p is a tuple of the form: $(\mathit{uid}, \mathit{id}, wt, wh, c, y, s)$ where:

uid is a management unit identifier: an integer value over the finite set of unit identifiers $\mathcal{U} \subset \mathbb{N}$.[1]

id is an integer which uniquely identifies the prescription. It ranges over the finite set of prescription identifiers $\mathcal{P} \subset \mathbb{N}$.

wt is the wood reward for thinning.

wh is the wood reward for harvesting.

c is the cork reward.

y is the year this operation applies to (as an offset from the starting year for the simulation).

s is the *species* of tree which is planted on this management unit, and therefore available for the next period.

Prescriptions represent a strategy for each management unit, which will apply for the duration of the simulation. There may be more than one possible prescription for each management unit, with the understanding that they are mutually exclusive, i.e. one must choose which prescription to apply to each management unit. In the data we are working with, the simulation is carried out over a period of 90 years, so the domain of the y variables is the range $\{1 \cdots 90\}$.

Moreover, we have a set U of management units, each of which is a tuple of the form $u = (\mathit{uid}, s, a)$, where:

- $\mathit{uid} \in \mathcal{U}$ is the management unit identifier, as above.
- $s \in \mathbb{R}^+$ is a surface area (obtained from a geographical outline), expressed in hectares.
- a is a set of adjacent management units, qualified with the length of the common border, i.e. a set of pairs (uid_a, l_a) where uid_a is another management unit identifier and l_a is the length (in meters) of the shared border between *uid* and uid_a.

We define the function $\mathit{adj} : \mathcal{U} \to 2^{\mathcal{U}}$, which maps a management unit identifier to the set of its adjacent units. We also introduce an auxiliary collection of parameters, A_{ij} which are defined as $A_{ij} =$ true if $j \in \mathit{adj}(i)$ and false otherwise.

We also introduce the function $\mathit{prescr} : \mathcal{U} \to 2^{\mathcal{P}}$, which associates a unit identifier with the set of prescription identifiers which may apply to it. Conversely, we define the function $\mathit{units} : \mathcal{P} \to 2^{\mathcal{U}}$ which maps a prescription identifier to the set of unit identifiers it may apply to.

We model the problem as a Constraint Satisfaction Problem (CSP) over Finite Domains (FD).

[1] Note that both the management unit identifiers and the prescription identifiers are remapped from the original external arbitrary string representation, which is more complicated than what we have here.

Let P_i be the prescription assigned to management unit $i \in \mathcal{U}$, P_i will range over the set of identifiers for all prescriptions $\mathcal{P}$. We say S is a *well-formed solution* to the planning problem if it is a complete assignment to P_i, $\forall i \in \mathcal{U}$ which also satisfies the constraint:

$$\forall i \in \mathcal{U}, P_i \in prescr(i) \tag{5.1.1}$$

Besides stating what constitutes a well-formed solution, we want to express further restrictions on admissible solutions. A case in point is the total contiguous harvested area limit constraint, which may be formulated as:

> *For all management units, whenever there is a time in which, under the selected prescription, the management unit is to be totally harvested, then the sum of the areas of the adjacent management units (and closure thereof) which are also bring totally harvested, must be less than a preset limit ω.*

To help in meeting this goal, we introduce a function which captures the concept, which we call *glade* : $\mathcal{U} \times \mathbb{N} \to 2^{\mathcal{U}}$. This function is defined in terms of an auxiliary function g:

$$\mathit{glade}(i,t) = g(i,t,\varnothing)$$

The auxiliary function g is of type $g : \mathcal{U} \times \mathbb{N} \times 2^{\mathcal{U}} \to 2^{\mathcal{U}}$ and is recursively defined as follows:

$$g(i,t,I) = \begin{cases} I, & \text{if } i \in I \vee \mathit{wh}_{i,P_i,t} = 0 \\ \bigcup\limits_{j \in \mathit{adj}(i)} g(j,t,I \cup \{i\}), & \text{otherwise} \end{cases}$$

Intuitively, $glade(u,t)$ is the set of management units which are being harvested, contiguous with u.

With these definitions, we may specify the *harvested area limit* constraint:

$$\forall i \in \mathcal{U}, \forall t \in \mathit{times}(i), \left(\sum_{j \in \mathit{glade}(i,t)} \mathit{area}(j) \right) < \omega \tag{5.1.2}$$

With constraints (5.1.1) and (5.1.2) we are guaranteed to produce only solutions which respect the limit on contiguous area harvesting.

5.2 Constraint Programming State of the Art

5.2.1 History

Constraint Programming is a programming paradigm best used to solve combinatorial optimization problems, it's main attraction is usually stated to be the "automatic" nature of its problem solving functions, in that a problem is declared and specified by the user and the computer attempts to solve it, find a solution, automatically. In many situations when using CP the programmer doesn't need to design complex algorithms because of its automatic solving capabilities which saves them time and resources that can be spent elsewhere.

The origins of CP can be traced back to 1963 when Ivan Sutherland of the Massachusetts Institute of Technology proposed in his PhD thesis a new system called Sketchpad[Bar11], also known as Robot Draftsman. The pioneering project is considered a breakthrough in several fields of computer science such as Computer Graphics, Graphical User Interfaces and, for the purpose of this document, Constraint Programming. The way to draw using this system was using a "pen" to touch and draw geometric shapes on a small screen, but internally it worked by declaring relations between the geometric shapes that the user drew, say that a line must be connected (constrained) to another line at a specific point, this allowed the user to manipulate these shapes in a way that the constraints specified would always hold. The idea here is that instead of writing code on how the shapes can be manipulated the constraints are declared by the user and the system automatically maintains them as the shapes are manipulated. Since the 70s many of the developments in CP technology were often achieved concurrently with developments in the Prolog logic programming language, the process of "unification" in Prolog, where the system tries to match the term on the left-hand side of an expression with the term on the right-hand side, and it's capabilities for "backtracking", the process through which the system goes back to a stage in the execution where it can alter it's path, essentially worked like constraints, allowing for Constraint Logic Programming to be developed and CP extensions to be made for the Prolog language. Throughout the 90s several CP conferences and other community events occurred to further the growth of CP and CP adoption and in 2005 the Association for Constraint Programming was founded which now governs the CP community. Nowadays there are several CP languages and CP extensions for other programming languages like Minizinc, a constraint modeling language compatible with various solversm, and Choco, a Java library with its own custom solver.

5.2.2 Constraint Programming Principles

A good way to understand CP is that it stands as a synthesis of Mathematical Programming, where the programmer specifies the problem and the system attempts to find a solution on its own, and Computer Programming, where the programmer has to instruct the system on how to go about finding a solution, in CP the programmer specifies the problem as a set of relations that must hold (not be broken) but they also specify how the system goes about finding a solution.

An application may be formulated as a *Constraint Satisfaction Problem* (CSP) $\mathcal{P}$, which consists of a triple (V, D, C) where V is a set of *variables*, D is a set of *domains* for the elements of V and C is a set of *constraints*, i.e. relations over $\mathcal{P}(\mathcal{D})$ which must hold. The nature of the domains for the variables (*Finite Domains*, Booleans, Sets, Real numbers, Graphs, etc.), together with the specific relations (i.e. the *Constraints*) greatly influence the class of problems and application areas for which Constraint Programming form a good match. The vocabulary of *built-in constraints* combined with *composition operators* results in a very feature-rich and expressive formalism, which is arguably closer to most application domains than traditional O.R. formulations.

A *Constraint Optimisation Problem* (COP) is like a CSP but we are also interested in minimizing or maximizing an *objective function*. To achieve this, one may equate the objective function to the value of a particular variable. It is then possible to solve a COP by iteratively solving interrelated CSPs, involving the addition of a constraint which establishes an inequation between the analytical definition of the objective function and the previously found value.

The model for an application problem may be declaratively formulated as a CSP, which will form the specification for a *constraint solver* to find a solution thereto. Many successful approaches have been followed to solve CSPs, namely systematic search, in which variables see their domain progressively restricted and each such step triggers the reduction of the domains of related variables, as dictated by the *consistency*

policy – these are in general designated as propagation-based constraint solvers and there are several ones, some being presented as libraries for use within a general-purpose programming language, such as Gecode [STL19] or Choco [PFL16]. Others offer a domain-specific language (DSL) which may be used to model a problem and provide it as input to different solvers; such is the case for instance for MiniZinc [NSB+07] or PyCSP3 [LS20].

Another approach entails selecting an initial solution candidate and working a path towards an actual solution by means of an *iterative repair* algorithm. The latter forms the basis for several *local search* techniques, which may be generalised to related methods called *metaheuristics*. Solvers which derive the strategy used to guide the search from the specification of a CSP are called *constraint-based local search* solvers [MH18] and combine the convenience of a declarative problem formulation with the (relative) efficiency of an *anytime algorithm*.

Solvers exist for both propagation-based search and metaheuristic search, which exhibit high performance and the capacity to make use of parallel hardware to attain yet better performance, e.g. as discussed in [CMDA18, RM18].

Constraint modeling allows one to express a problem by means of both simple arithmetic and logic relations, but also resorting to *global constraints*. These are instance-independent yet problem-class-specific relations, for which particular dedicated algorithms can be devised and encapsulated in a reusable specification component. Intuitively, a global constraint expresses a useful and generic higher-level concept, for which there is an efficient (possibly black-box) implementation. For instance, the `AllDifferent` constraint applies to a set of variables and requires them to take pairwise distinct values. It may be internally implemented in a naïve way by saying that each distinct pair of variables in the list must be different, or it may resort to a more specialized algorithm to achieve the same result more efficiently. The application programmer will benefit from the performance gain with no additional effort.

Global constraints have proved to be a fertile ground for effective research, over the years. A limited common set of global constraints has been presented in the XCSP^3-core document [BLAP20], which lists 20 such frequently used and generally useful constraints. This forms the basic vocabulary of XCSP^3-core, an intermediate representation for CSPs designed with the purpose of interfacing different high-level modeling tools with distinct specific constraint solvers.

5.2.3 Example

An example of a CP problem is the following, the programmer declares the constraint variables A, B and C and they are declared to have the domain of {1, 2, 3},{4, 5, 6, 12} and {10, 11, 12} respectively, then the programmer declares the following constraints:

C module 2 is 0 (so C is an even number).

C is equal to *A* * *B*.

The constraint model is ready and the programmer calls on the Solver to find combinatorial solutions to the problem, most solvers do this by choosing a variable and one of the values in that variables domain and "labeling" that value (the way in which the variables and values are chosen is called the Search Strategy), meaning they assume the variable will take that value and move on to another value which will also be labeled, at some stage the Solver will check if the values it attributed to these variables have broken any of the constraints and if they did it backtracks to a point where the constraints weren't broken and labels the variables with another one of their possible values until all variables are instantiated with values that

don't break the constraints. In this example the solver should return the results: $\{A = 1, B = 12, C = 12\}, \{A = 2, B = 5, C = 10\}, \{A = 2, B = 6, C = 12\} and \{A = 3, B = 4, C = 12\}$

If the problem was to become a Constraint Optimization Problem the programmer could also declare that:

The value of *C* should be maximized.

The value of *A* should be minimized.

And the solver would, after satisfying the previous constraints, satisfy these 2 new ones and return only the result: $\{A = 1, B = 12, C = 12\}$

5.3 Implementation

Our initial implementation was carried out using the `Choco` Constraint Solver framework [PFL16] and written in the Java programming language.

The process of implementing the problem entails firstly setting up 2 data structures, one is an array containing all the information of each MU obtained from the input called `Nodes`, and the other is an array of possible values to represent the possible prescriptions to apply to each MU called `MUS`. Note that `MUS` is a constraint variable array so the domain of possible values for each variable will be reduced until a solution is found. Another constraint variable array is called `WoodYields`, similar to `MUS` each index of the array represents something about the solution, in this case they represent the wood yielded by harvesting/thinning that MU when the corresponding prescription found in MUS is applied (this process involves using the "Element" constraint from the Choco-Solver framework), this is done so that at the end the contents of this array can be summed up to obtain the total amount of wood yielded by the solution. Once the setup is done we iterate through the main loop, as shown in Listing 5.1.

Listing 5.1: Main Loop

```
for (Var node in Nodes) {
    CreateConstraint (node, MUS, MAXLIMIT);
}
SumOfWood = sumConstraint(WoodYields, range=[-999999, 999999]);
```

This loop iterates through every `MU` in the input and imposes all valid constraints pertaining to it and its possible prescriptions. These constraints are implemented as a global constraint, via a custom propagator, as shown in Listing 5.2.

The propagator essentially iterates through the MU's possible prescriptions and checks if a prescription value can be applied by recursively checking its neighbouring MUs and their possible prescriptions. If at any one point the total sum of contiguous forest area cut down exceeds the given limit, the propagator fails and another value will be chosen. The propagator calls a recursive function which verifies that a given MU is valid, w.r.t. the maximum cut area requirement, as shown in Listing 5.3.

Listing 5.2: Custom Propagator

```
void propagate (){
    for (int year in node.yearsWithCuts) {
        try {gladePropagate(node, year, 0)}
        catch (Exception limitSurpassed) { fails() }
    }
```

```
}
```

Listing 5.3: Propagator Helper

```
int gladePropagate (node, year, sum)
  if (node.hasCut() & node.isValid()) {
    sum = sum + node.area;
    if (sum > MAXLIMIT) { throw Exception }
    for (neighbourNode in node.neighbours())
        sum = gladePropagate (neighbourNode, year, sum);
  }
  return sum;
}
```

After leaving the main loop, the model has been fully setup. The Choco Solver framework is then told to set the objective of the Solver to be one of *Maximisation*, and the target to be maximised is a constraint variable with the aforementioned sum of the contents in the `WoodYields` array.

Ultimately, the solver is activated and, if a solution is found then the resulting MU/prescription pairs are written to an output file.

5.4 Experimental Evaluation

The data for computational experiments is the same as in article [MBLB21] for the region of Vale de Sousa in the north of Portugal. The testing was done on a laptop running Ubuntu 20.04.3 LTS, with 4GB of available ram available and 4 cores. The code was compiled using java 8.

It should be noted that no solution for the full problem could be found in a reasonable time frame: the complete problem includes 1373 management units and the program ran for an entire day and a solution was not found. Consequently we opted for an approximation to the problem.

5.4.1 Limiting by distance to initial MU

By limiting the input and therefore the problem size it is possible to more closely observe the growth of the execution time.

To this end we decided to limit the number of MUs taken into account by initially choosing an "initial MU" and then only working with MUs within a maximum distance to that MU. So in the implementation the third argument is the Id of the "initial MU" and the fourth specifies a maximum distance, meaning only MUs within that distance to the "initial MU" are taken as valid input, the rest are ignored.

With this setup, while choosing the MU with Id `0` (which should be located near the Penafiel-PaivaNorte border) as the "initial MU" and gradually increasing the maximum distance by 1km at a time, solutions are found until the 5km distance mark, where the program finishes in around 4 minutes but does not find a solution (meaning that it is not possible to satisfy the constraints.) Increasing the distance to 6km results in the same outcome and by the 7km distance mark the program takes about 1.30-2h to finish execution and figure 5.1 shows the valid MUs in this context.

Figure 5.1: MUs within a 7km distance from the "initial MU"

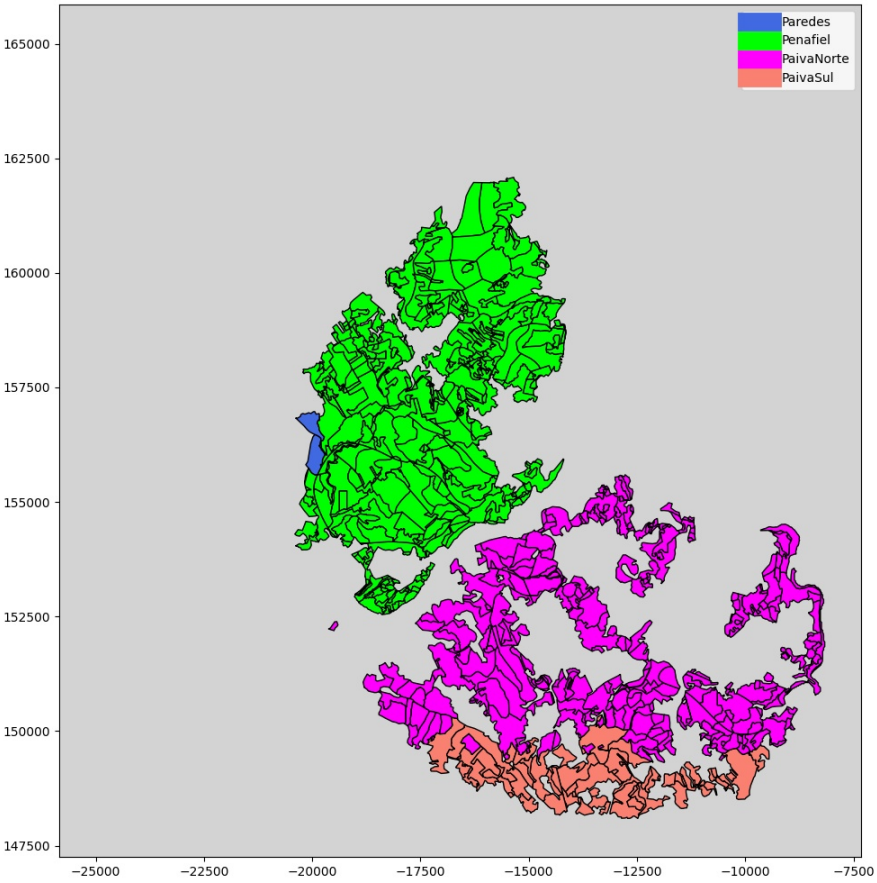


Table 5.1: Number of MUs, Runtime and Wood Yield for each step

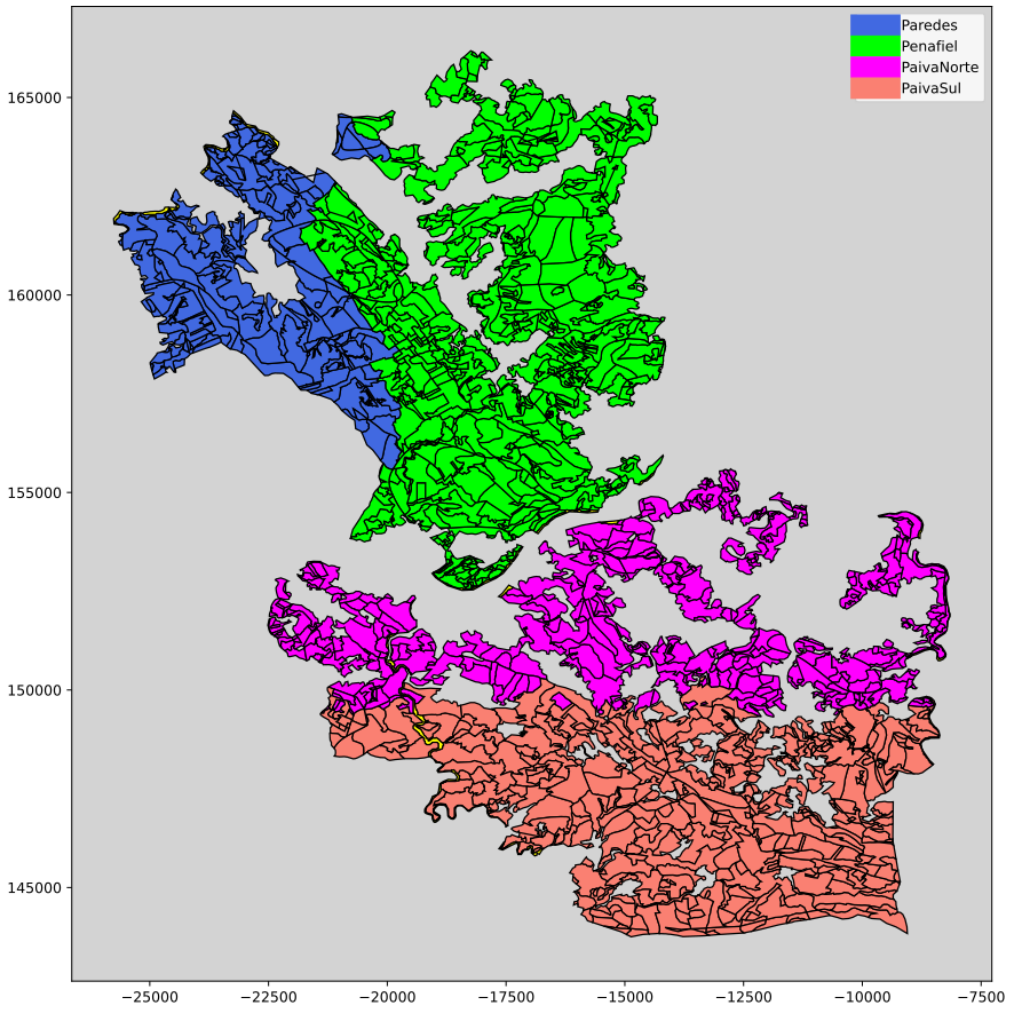
Distance(km)	Number of MUs	Time(min)	Wood Yield(kg)
1	21	0.162	58540
2	47	0.188	187724
3	103	0.253	557720
4	178	0.310	1042205
5	319	4.220	Solution not found
6	478	13.490	Solution not found
7	642	100.650	Solution not found
8	810	—-	Solution not found

Once the 8km mark is reached we are dealing with 810 MUs, however this problem proves to be too complex and a solution is not found within a reasonable time frame. The median of 3 tests is shown in table 5.1.

5.4.2 Restricting to a Sub-Region

Another way to limit the size of the problem is by selecting which of the 4 sub-regions in Vale de Sousa should be managed, these are "Paredes", "Penafiel", "PaivaSul" and "PaivaNorte" and are shown in figure 5.2.

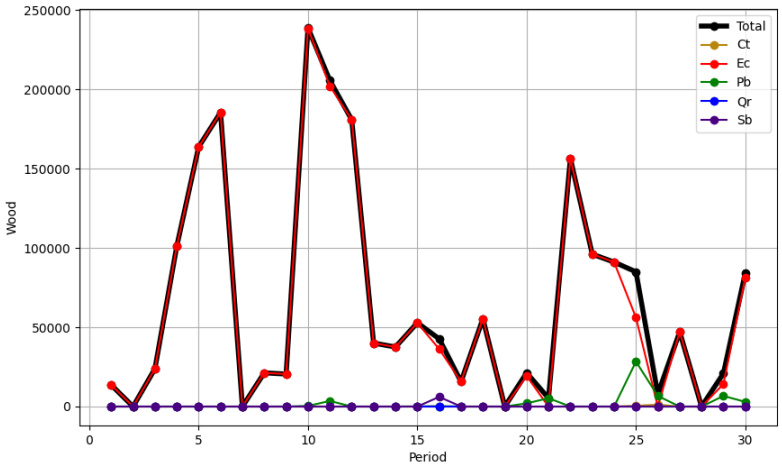
Figure 5.2: Sub-Regions in Vale De Sousa



Paredes(Blue)

The Paredes sub-region of Vale De Sousa has 159 MUs, of which 18 have an area of over 40ha, one of those being 49ha. There is a total of 1998 possible prescriptions with 222 pertaining to the aforementioned 18 largest MUs. With the maximum limit set to 50ha the optimal solution is found in under a minute, this solution yield 1356 tonnes of wood.

Figure 5.3: Wood yield of a possible solution when applied to the Paredes sub-region



In the previous tests where the limit is based on maximum distance to the starting MU with internal Id 0, it's observable that the Paredes sub-region doesn't factor in before the problem becomes unsolvable so it's possible to conclude that this sub-region by itself is not problematic nor too complex for the implementation to handle.

PaivaNorte(Pink)

This sub-region has 351 MUs with only 8 of them being over 40ha and a total of 7362 possible prescriptions. Predictably, if the assumption that the number of large MUs is a meaningful factor, the optimal solution within the 50ha limit is found in a reasonable time frame, yielding 2021 tonnes of wood. A large section of this sub-region is always within the distance limits to the MU with Id 0 since that MU is in PaivaNorte. Figure 5.5 provides an example of a solution outputted by the implementation, in this case the actions to take in PaivaNorte during period 25.

Figure 5.4: Wood yield of a possible solution when applied to the PaivaNorte sub-region

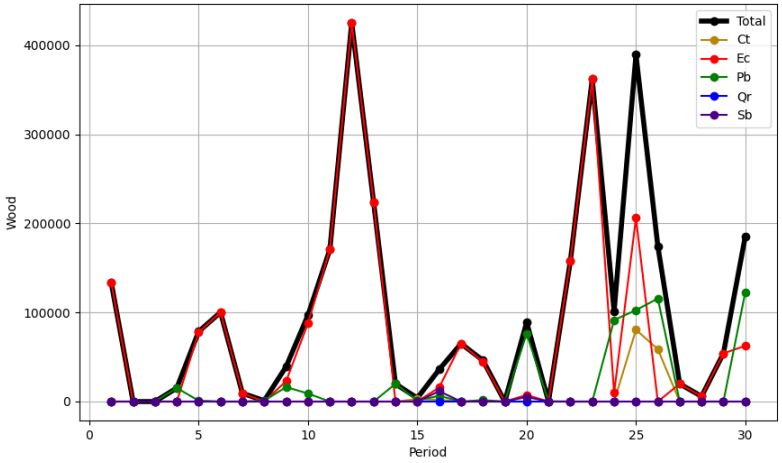
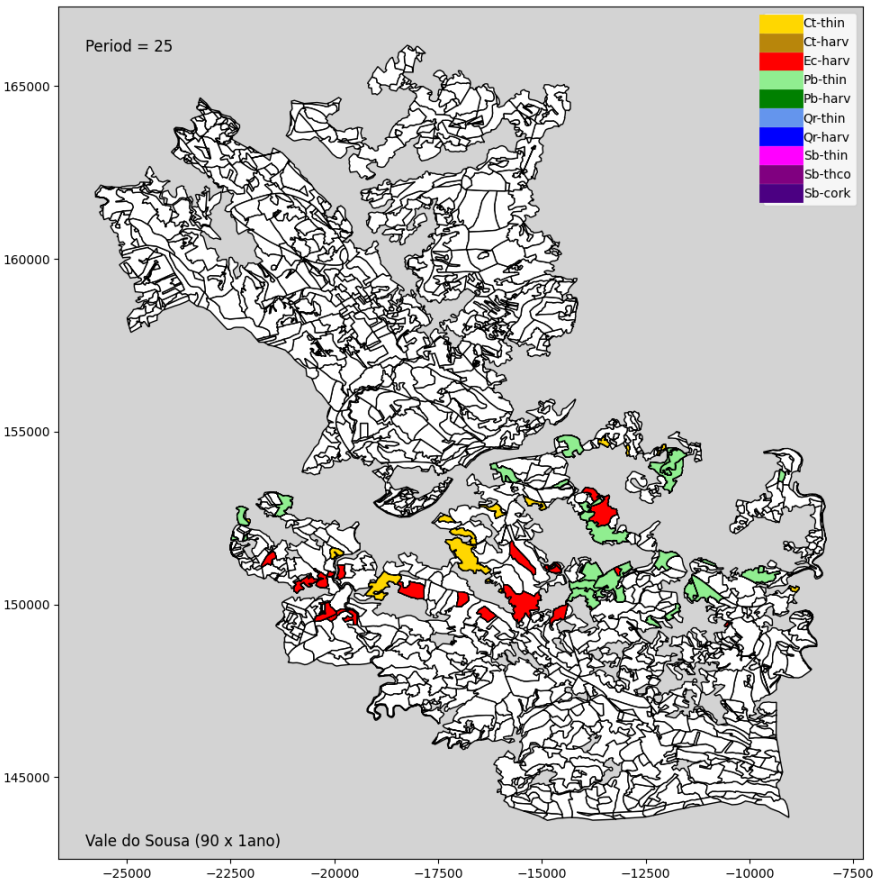


Figure 5.5: Example of a solution provided by the implementation



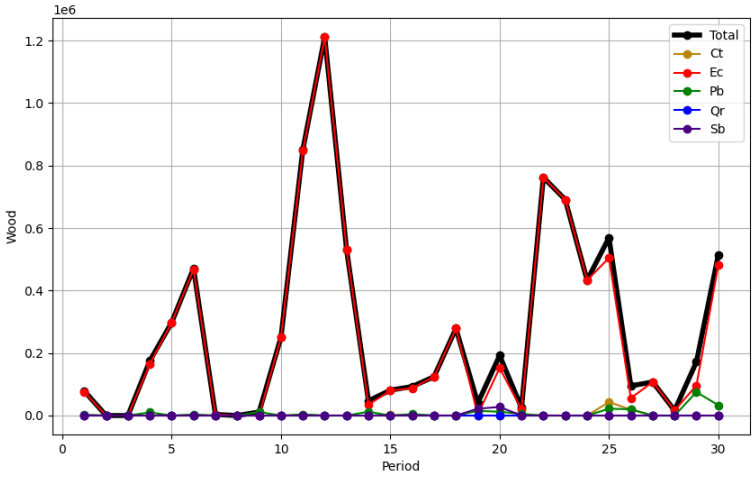
PaivaSul(Light Red)

This sub-region has 350 MUs, of which 30 have an area of over 40ha and there are 17 instances where these large MUs are adjacent to each other, there is a total of 6998 possible prescriptions. The MU with internal Id 1133 belonging to this sub-region is adjacent to 3 other large MUs, which would force the solver to find solutions where these MUs are never harvested in the same year.

If the limit is set to 50ha then valid solutions which satisfy the constraints can be found but the optimal solution is not found in a reasonable time frame, by increasing this limit gradually the value with which the solver finds the optimal solution in a similar time frame as the previous tests is 81ha, which yields 3811 tonnes of wood.

The upper section of the PaivaSul sub-region is within the 7km distance limit to the MU with Id 0, however when that distance limit is increased, some of the lower section of the sub-region where more large MUs are located becomes valid input which adds to the complexity of the problem.

Figure 5.6: Wood yield of a possible solution when applied to the PaivaSul sub-region



Testing Penafiel and Paredes+PaivaNorte+PaivaSul separately

By searching for solutions to the problem for the Penafiel sub-region and the rest of Vale de Sousa separately some interesting results are obtained. The prototype was first run without the 455 MUs located in Penafiel, so only searching the 860 remaining MUs, and with the area limit set 50ha. The optimal solution was found in 789 minutes (around 13 hours), obviously a large amount of time but when running the implementation with all the sub-regions (including Penafiel) and the limit set to 50ha a solution is not found within this time frame. This gives us a solution that maximises wood yield to the original problem but only for the Paredes, PaivaNorte and PaivaSul sub-regions. Of note is that, since the Paredes sub-region is separated from both Paivas, the optimal solution found is the same as the one obtained when searching only in the Paredes sub-region, however the solution for the Paiva sub-regions differs from the one found when searching them separately.

The next step was finding a solution only for the Penafiel sub-region.

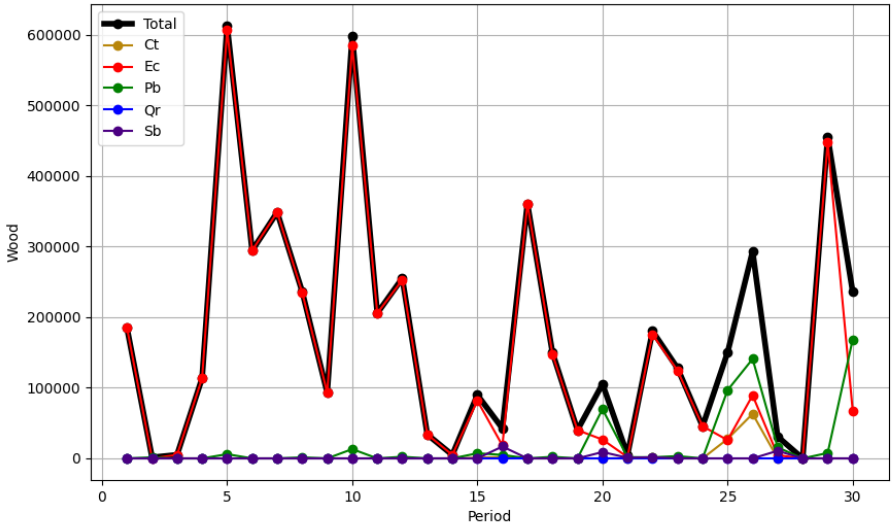
Penafiel(Green)

This sub-region has 455 MUs, of which 26 have an area of over 40ha, 4 of which have 49ha, and a total of 8593 possible prescriptions.

There are 13 instances of these large MUs being adjacent to each other, with adjacencies that seem to imply a clustering of around 9 of these large MUs. These factors add up to this sub-region being the most problematic and possibly responsible for the dramatic increase in complexity when attempting to find a solution for the whole forest.

The limit had to be raised to 96ha in order to find a solution in a reasonable time frame, which yields 3626 tonnes of wood.

Figure 5.7: Wood yield of a possible solution (not respecting the 50ha limit) when applied to the Penafiel sub-region



Increasing the limit of the area constraint

Another useful test to analyze this implementation is to check for variations in total wood yield in the solutions found, when the maximum limit for contiguous forest area harvested in a year is increased. The sub-regions picked for this test were the Paredes and PaivaNorte sub-regions because they both have solutions when the area limit constraint is set to 50ha.

Table 5.2: Area limit and Wood Yield for each step, Paredes sub-region

Limit(ha)	Wood Yield(kg)
50	1356534
55	1363613
60	1366454
65	1377433
70	1393014
75	1412062
80	1411094
85	1415788
90	1418554
95	1417703

Table 5.3: Area limit and Wood Yield for each step, PaivaNorte sub-region

Limit(ha)	Wood Yield(kg)
50	2021722
55	2035704
60	2042593
65	2044444
70	2049264
75	2049392
80	2073522
85	2083581
90	2082840
95	2086786

Predictably, as shown in tables 5.2 and 5.3, the wood yield tends to increase when the area limit constraint is increased because the solver attempts to find solutions that maximize that output but at some point the amount of wood yielded tends to plateau.

5.5 Conclusions

In this project we proved that the Forest Management problem can be conveniently described as a Constraint Optimisation Problem, where all the selection criteria and optimisation objectives may be integrated without need for further changes or additions to the base model.

We contributed a new global constraint, which ensures a "flood filled" area surrounding a given point observes specific conditions, e.g. being under a general limit.

The implementation, although functional, is currently limited in its reach when applied to the full-scale problem, but it compares well to the MILP solver presented in this paper. It also has the merit of having already gone through peer-review and been published in a scientific journal.

6

Multi-Criteria Optimization

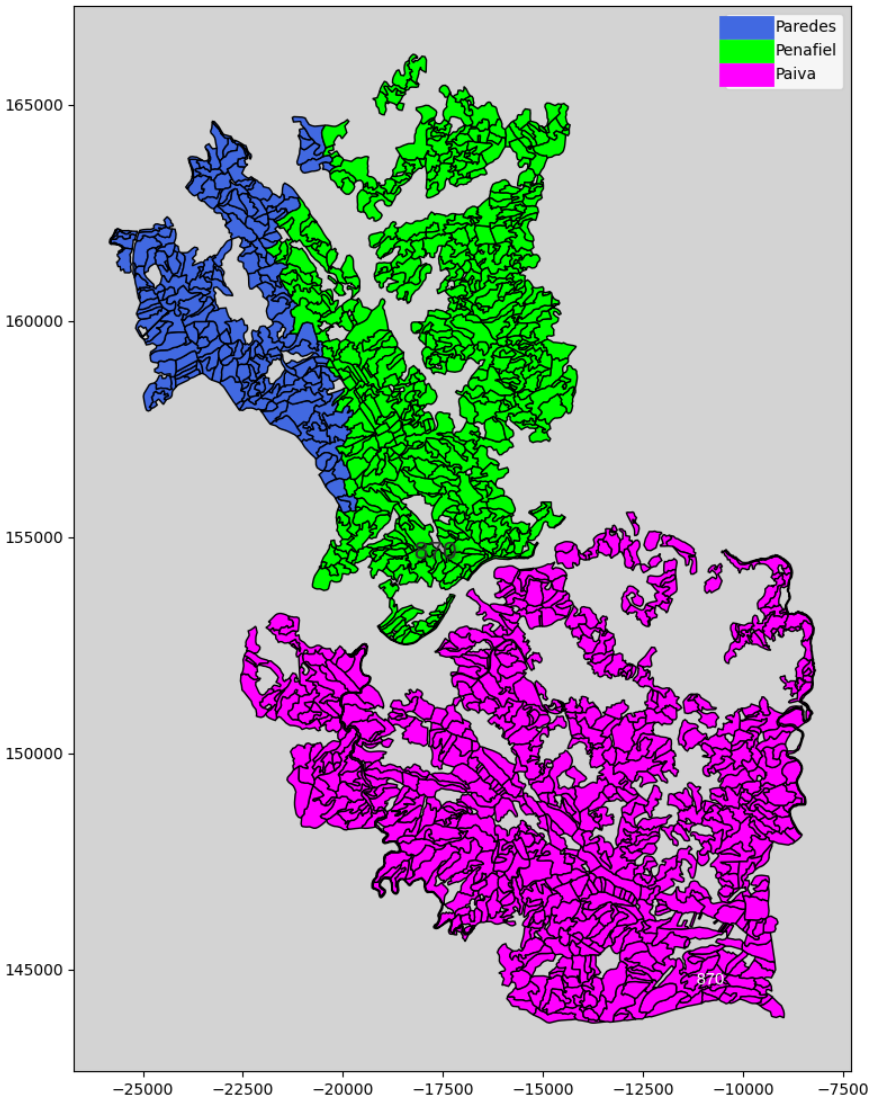
6.1 Introduction

Once the BIOECOSYS project was completed we sought out the opportunity to continue working on constraint management approaches to forest management through a new project and FCT grant named "A multiple criteria approach to integrate wildfire behavior in forest management planning (MODFIRE)". The subject of this project was still the Vale de Sousa forest but this time the focus was on integrating more criteria besides Wood Yield into the constraint programming model, for this we were given new data regarding prescriptions, optimizable criteria and forest structure.

6.1.1 New Data

Following an analysis of the former structure of the Vale de Sousa forest a new structure was proposed.

Figure 6.1: New representation of Vale de Sousa Forest



The main changes to the forest structure were the breaking up of the large MUs on the northeast of the Penafiel sub-region and the removal of separation between the Paiva sub-regions.

The planning horizon was reduced to 50 years, starting in 2020 and ending in 2070.

The new prescriptions now contain several rewards/effects associated with their application, those being the following: Wood Yield, Soil Loss, 3 different measures of Fire Risk Protection, Net Present Value, Biodiversity, Soil Loss, Cash Flow, Carbon Stock and Biomass.

Some criteria such as biodiversity have 3 variations relating to the level of "prep-work" done on the forest, such as cleanup, these tests were done with the variation that assumes no prep-work was done.

6.1.2 Implementation

The implementation presented in the previous chapter was modified to work with the new data. It's worth explaining how multi-criteria optimization works in Choco, much like single-criteria optimization we tell the model which criteria to optimize, but while previously the solver would try to find 1 solution that maximizes/minimizes the single criteria chosen, for multi-criteria optimization the solver will run in a loop finding all possible valid solutions for the problem and once the loop terminates it calculates the coordinates of the Pareto Efficient solutions, these coordinates, as well as the non-efficient coordinates, are then used to plot the Pareto Frontier outside the implementation. Of note is that Choco requires that all criteria have to be either maximized or minimized so in the implementation minimization problems (such as reducing Soil Loss) were converted to maximization problems.

Usually the the Solver will take too long to find all possible solutions when dealing with a large number of MUs so we set a timer the Solver loop so that it exits the loop after a set amount of time and the Pareto Frontier is calculated using all the solutions found until then, the loop can also exit after an Out of Memory error.

These tests were carried out on a version of the implementation which integrates 3 of the optimizable criteria, those being Wood Yield, Soil Loss and Fire Risk Protection (measured through the Fire Risk Protection Percentile), the more optimizable criteria are included in the implementation the more constraints are applied resulting in more complex problems (for example the PaivaWest sub-region starts taking too long to solve once 6 criteria are integrated into the implementation) so for now these 3 criteria were chosen.

Another modification was the setting of a minimum border length for an adjacency between 2 MUs to be taken into account, this was necessary because the software which determined the adjacencies between MUs gave us adjacencies with borders as short as 0.001 meters which we deemed to be unnecessary, so the minimum border length for an adjacency to count was set to 50 meters.

6.2 Results

6.2.1 Valid Solutions and Single Criteria Optimization

When it comes to finding valid solutions (without optimizing criteria) the Solver accomplishes this goal for the Paredes sub-region, the Penafiel sub-region and Paredes and Penafiel put together, however it does not find valid solutions for the Paiva sub-region. When it comes to single-criteria optimization the Solver finds optimal solutions for most criteria for the Paredes sub-region and the Penafiel sub-region, but not the 2 sub-regions at the same time nor the Paiva sub-region. So we can conclude that the new forest structure is favorable in that the Penafiel sub-region can be solved by itself, even with added complexity due to more optimizable criteria, while with the previous structure it couldn't.

Table 6.1: Results of single criteria optimization

Sub-Region	WoodYield(kg)	Soil Loss(ha)	FireRisk Protection
Paredes	892871	-335789	7339
Penafiel	2349476	-1212446	18619

6.2.2 Multi-Criteria Optimization for Paredes and Penafiel

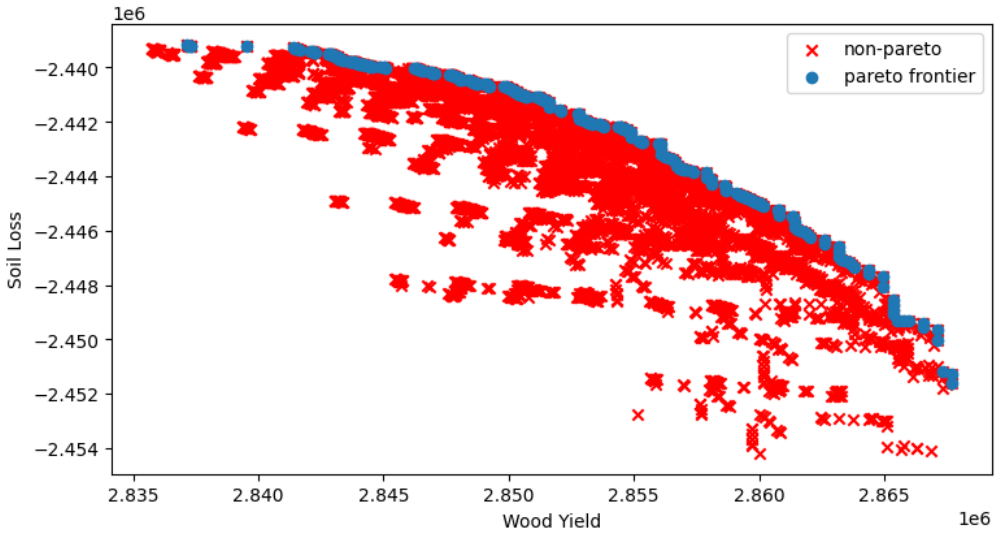
Since these 2 sub-regions have no adjacencies to the Paiva sub-region, it makes sense to treat them as a separate problem. While most of the optimizable criteria are integrated into the implementation the 3 criteria chosen to demonstrate multi-criteria optimization were Wood Yield, Fire Risk Protection and Soil Loss, the Pareto Frontiers were plotted using a python script.

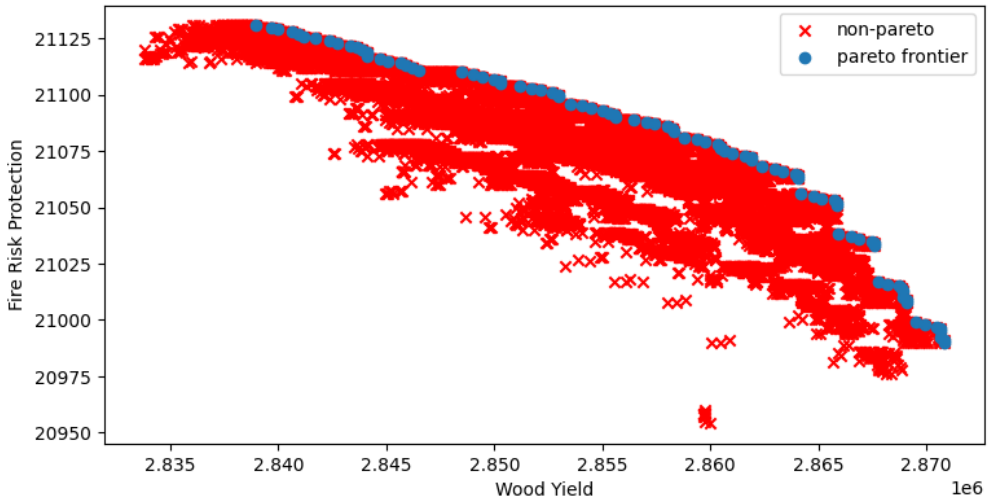
The solver was left running and finding valid solutions for 8 hours each time find thousands of possible solutions and calculating the Pareto Efficient ones.

Table 6.2: Breakdown of solutions found for each combination

Criteria	Nº of Solutions Found	Nº of Pareto Solutions
WY/SL	8668	209
WY/FRP	16701	91
SL/FRP	4195	15
WY/SL/FRP	11276	434

Figure 6.2: Pareto Frontiers for the 3 possible combinations of 2 optimizable criteria





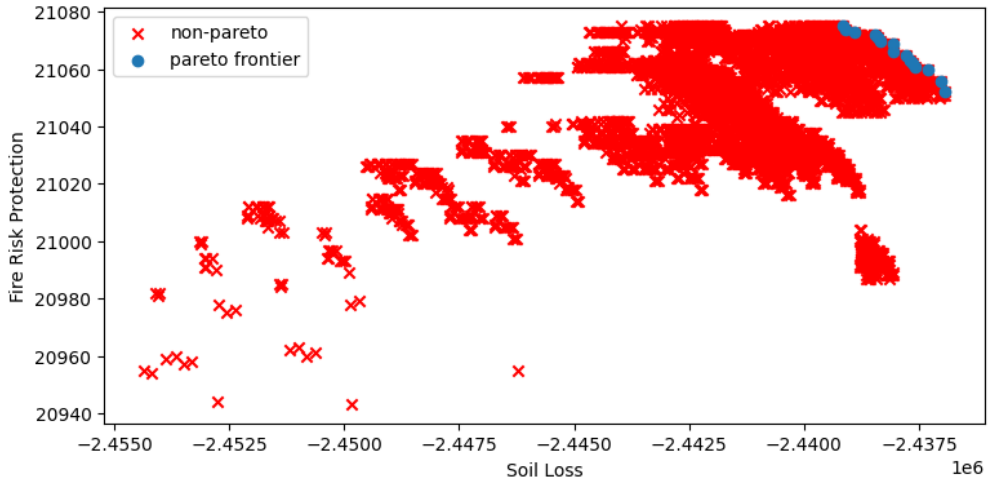
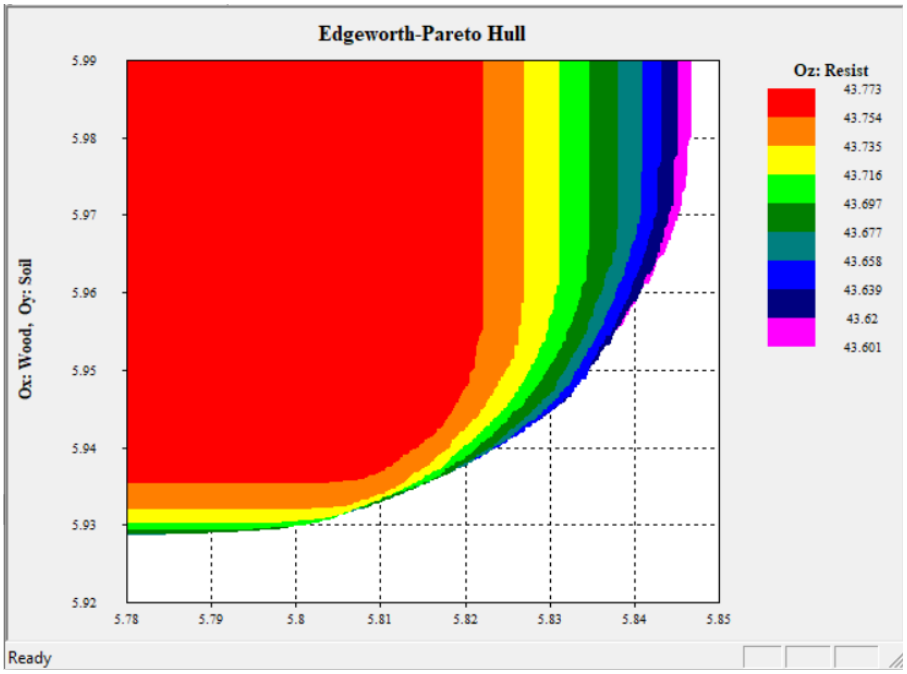


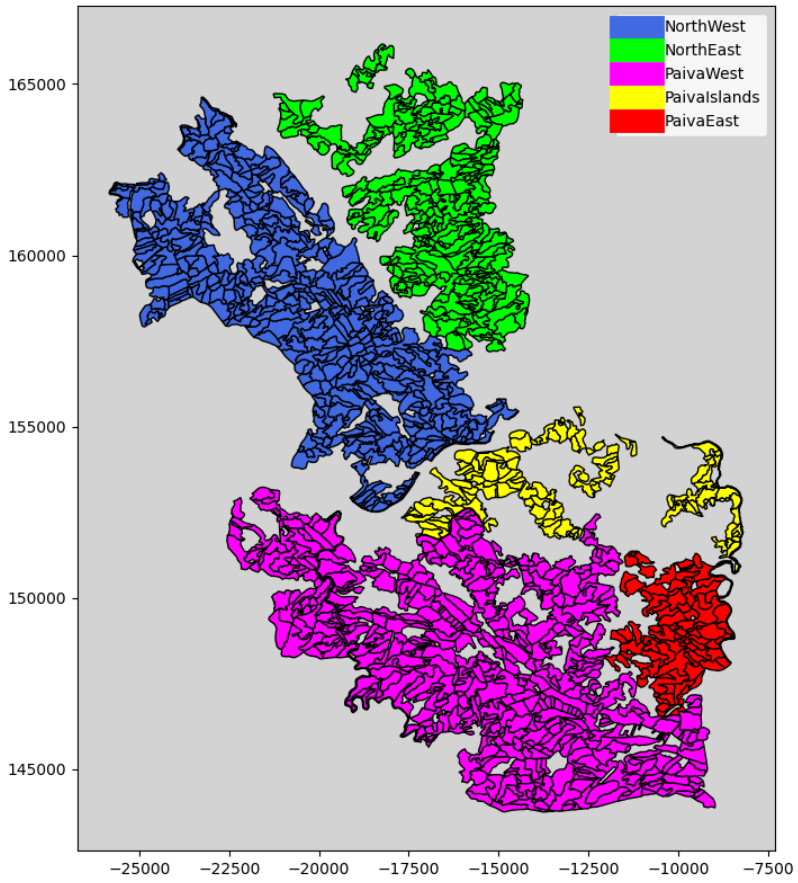
Figure 6.3: Edgeworth-Pareto hull of all the Pareto solutions found when optimizing 3 criteria



6.2.3 New Sub-Regions

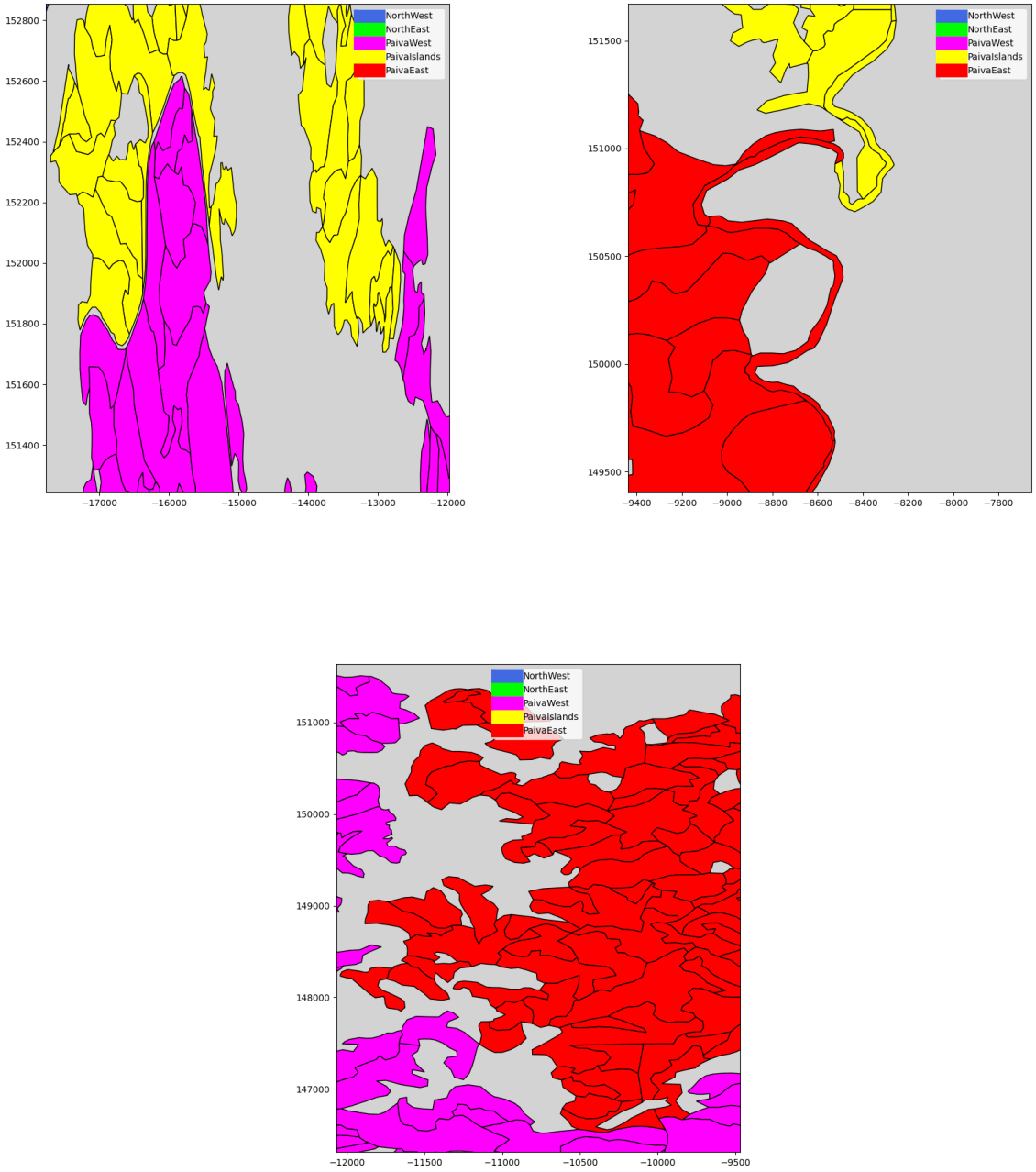
As previously explained the Paiva sub-region even by itself is problematic for the solver, to bypass this we decided to split this sub-region into 3 sub-regions that can be solved by themselves, the choice of where to apply these splits was made by trying to find sections where there are few adjacencies between sub-regions or where they are completely separated. This was also done for the Penafiel and Paredes sub-regions.

Figure 6.4: New Sub-Regions of Vale de Sousa forest



As observed there are few adjacencies between these sub-regions, there are only 2 adjacencies unifying the NorthWest and NorthEast sub-regions, the yellow PaivaIsland sub-region is completely separated from PaivaWest and is connected to PaivaEast only by a slim long MU and the red PaivaEast sub-region only shares a few adjacencies with PaivaWest in the south.

Figure 6.5: Close-up of adjacencies between the Paiva Sub-Regions



6.2.4 Single-Criteria Results with Sub-Regions

Table 6.3: Results of single criteria optimization

Sub-Region	WoodYield(kg)	Soil Loss(ha)	FireRisk Protection
PaivaWest	2437331	-1722778	22066
PaivaEast	413778	-288994	3994
PaivaIslands	439988	-208956	3406
NorthEast	1256902	-721009	11019
NorthWest	1981214*	-828359	14926

In this manner, by separating the Paiva sub-region into 3 separate problems, we can obtain single-criteria optimized solutions for the Paiva sub-region. Note that when finding the Wood Yield optimal solution for the NorthWest sub-region the problem had become too complex for the solver, so to find this solution the constraints related to the other 2 optimizable criteria were removed.

6.3 Conclusions

Though there is further work to be done on this on this multi-criteria implementation, which we'll continue as a part of the MODIFIRE project, so far we proved that our earlier implementation was flexible enough that only some modifications relating to the 3 additional criteria were needed for it to be able to maximize/minimize said criteria, after these modifications the Main execution file ended up with 370 lines of code, the code for the Custom Propagator we developed didn't need to be modified and stayed at 260 lines of code. The approach of dividing unsolvable problems into solvable problems may also prove itself to be significant once we develop an effective way to join the solutions of these solvable problems as a representation of the full problem.

7

Conclusions and Future Work

In this thesis, we designed and implemented a solver able to deal with situations which are beyond the abilities of the existing state-of-the art MILP approaches, by resorting to a custom propagator in finite-domain constraints, implemented within the Choco constraint programming framework. It is clear that constraint programming provides higher levels of abstraction which will clearly be beneficial in modeling and solving concrete cases.

An obvious area to improve upon is reducing the number of constraints applied alongside the further integration of optimizable criteria to the implementation, which themselves increase the number of constraints.

Another is investigating the possibility of utilizing more computational resources through parallel constraint satisfaction, as discussed in the already cited[RM18], since the solving process becomes too complex to finish in a reasonable time frame with the resources currently available. As Constraint Programming relinquishes the notion of control, it stands as a natural candidate for the effective application of non-standard computing architectures and methods such as constraint-based metaheuristic solving[DMDA21].

In regards to multi-criteria optimization, the work of developing an effective way to "join" the Pareto solutions of the different sub-regions to obtain a complete Pareto frontier from which a solution may be picked as a complete solution to the problem continues under the MODFIRE project.

List of Figures

List of Tables

A

Appendix

A.1 Appendix A

These symbols are used in most formulations

- i = Management Unit.
- t = index of time periods.
- A cluster is a set of adjacent MUs.
- A clear-cut is the action of harvesting all the MUs in a cluster during the same period.
- NS is the total number of MUs.
- NT is the total number of time periods.
- The graph G = {V, E} represents the forest, where each node in V = {1...NS} corresponds to a MU and the nodes to each edge in E = {1...NE} correspond to two adjacent MUs.

- The temporal horizon is defined by T = {1...NT}
- p_i^t = reward associated with harvesting unit i in period t.
- a_i = area of management unit i.
- A_{max} is the maximum size of a clear-cut in any period in the planning horizon.
- N_i = Units adjacent to unit i.
- x_i^t = Variable that takes the value of 1 if unit i is harvested during period t and 0 otherwise.

A.2 Appendix B

These symbols are used in chapter 1.2

- f(x_i^t) = recursive function summing all the area contiguously harvested starting from unit i during period t.
- C_i^t = Cost of harvesting unit i in period t.
- LB_t = Lower bound on cost in period t.
- UB_t = Upper bound on cost in period t.

A.3 Appendix C

- v^t is the volume of timber harvest in period t.
- v_i^t is the timber volume of timber obtained from harvesting MU i during period t
- b_{lt} is a lower bound on decreases in the harvest level between periods t and t + 1 (where, for example, $b_{lt} = 1$ would require nondeclining harvests and $b_{lt} = 0.9$ would allow a decrease of up to 10%)
- b_{ht} is an upper bound on increases in the harvest level between periods t and t + 1 (where $b_{ht} = 1$ would allow no increase in the harvest level and $b_{ht} = 1.1$ would allow an increase of up to 10%)
- P_i is the set of indexes corresponding to the management units in path i.
- N_{Pi} is the number of management units in path i.
- Age_i^t is the age of management unit i at the end of the planning horizon if it's harvested in period t.
- Age^T is the target average age of the forest at the end of the planning horizon.

A.4 Appendix D

- B_i is the number of MUs adjacent to MU i.
- k is a spatial unit that could be a MU, a predefined set of MUs or a cluster
- K is the set of all spatial units.
- K_i is the set of units containing MU i.

Bibliography

[Apt03] Krzysztof R. Apt. *Principles of constraint programming*. Cambridge University Press, 2003.

[Bar11] Roman Barták. *History of Constraint Programming*. 01 2011.

[BB99] Kevin Boston and Pete Bettinger. An analysis of monte carlo integer programming, simulated annealing, and tabu search heuristics for solving spatial harvest scheduling problems. *Forest Science*, 45(2):292–301, 1999.

[BB02] Kevin Boston and Pete Bettinger. Combining tabu search and genetic algorithm heuristic techniques to solve spatial harvest scheduling problems. *Forest Science*, 48(1):35–46, 2002.

[BBSG17] Pete Bettinger, Kevin Boston, Jacek Siry, and Donald Grebner. Spatial restrictions and considerations in forest planning. In *Forest Management and Planning*, pages 249–267. Academic Press, 2017.

[BEB14] Paulo Borges, Tron Eid, and Even Bergseng. Applying simulated annealing using different methods for the neighborhood search in forest planning problems. *European Journal of Operational Research*, 233(3):700–710, 2014.

[BHR99] José G. Borges, Howard M. Hoganson, and Dietmar W. Rose. Combining a decomposition strategy with dynamic programming to solve the spatially constrained forest management scheduling problem. *Forest Science*, 45(1):201–212, 1999.

[BK05] Emin Zeki Baskent and Sedat Keles. Spatial forest planning: a review. *Ecological Modelling*, 188:145–173, 2005.

[BLAP20] Frédéric Boussemart, Christophe Lecoutre, Gilles Audemard, and Cédric Piette. Xcsp3-core: A format for representing constraint satisfaction/optimization problems. *arXiv preprint arXiv:2009.00514*, 2020.

[CMB08] Miguel Constantino, Isabel Martins, and José G. Borges. A new mixed-integer programming model for harvest scheduling subject to maximum area restrictions. *Operations Research*, 56(3):542–551, 2008.

[CMDA18] Philippe Codognet, Danny Munera, Daniel Diaz, and Salvador Abreu. Parallel Local Search. In Youssef Hamadi and Lakhdar Sais, editors, *Handbook of Parallel Constraint Reasoning*, pages 381–417. Springer, 2018.

[DMDA21] Jonathan Duque, Danny A. Múnera, Daniel Diaz, and Salvador Abreu. Solving QAP with auto-parameterization in parallel hybrid metaheuristics. In Bernabé Dorronsoro, Lionel Amodeo, Mario Pavone, and Patricia Ruiz, editors, *Optimization and Learning - 4th International Conference, OLA 2021, Catania, Italy, June 21-23, 2021, Proceedings*, volume 1443 of *Communications in Computer and Information Science*, pages 294–309. Springer, 2021.

[EBA22] Eduardo Eloy, Vladimir Bushenkov, and Salvador Abreu. Constraint modeling for forest management. In Tatiana V. Tchemisova, Delfim F. M. Torres, and Alexander Yu. Plakhov, editors, *Dynamic Control and Optimization*, pages 185–200, Cham, 2022. Springer International Publishing.

[Elo] Eduardo Eloy. Bioecosys conference (2/2).

[GMVW09] Marcos Goycoolea, Alan Murray, Juan Pablo Vielma, and Andres Weintraub. Evaluating approaches for solving the area restriction model in harvest scheduling. *Forest Science*, 55(2):149–165, 2009.

[GR05] Eldon Gunn and Evelyn Richards. Solving the adjacency problem with stand-centered constraints. *Canadian Journal of Forest Research*, 35:832–842, 2005.

[GRBC19] Chourouk Gharbi, Mikael Ronnqvist, Daniel Beaudoin, and Marc-Andre Carle. A new mixed-integer programming model for spatial forest planning. *Canadian Journal of Forest Research*, 49:1493–1503, 2019.

[HJ93] John Hof and Linda Joyce. A mixed integer linear programming approach for spatially optimizing wildlife and timber in managed forest ecosystems. *Forest Science*, 39:816–834, 1993.

[LS20] Christophe Lecoutre and Nicolas Szczepanski. Pycsp3: Modeling combinatorial constrained problems in python. *arXiv preprint arXiv:2009.00326*, 2020.

[MB01] Marc E. McDill and Janis Braze. Using the branch and bound algorithm to solve forest planning problems with adjacency constraints. *Forest Science*, 47(3):403–418, 2001.

[MBLB21] Susete Marques, Vladimir Bushenkov, Alexander Lotov, and José G. Borges. Building pareto frontiers for ecosystem services tradeoff analysis in forest management planning integer programs. *Forests*, 12,1244, 2021.

[MH18] Laurent Michel and Pascal Van Hentenryck. Constraint-based local search. In Rafael Martí, Panos M. Pardalos, and Mauricio G. C. Resende, editors, *Handbook of Heuristics*, pages 223–260. Springer, 2018.

[MJ00] Marc McDill and Braze J. Comparing adjacency constraint formulations for randomly generated forest planning problems with four age-class distributions. *Forest Science*, 46:423–436, 08 2000.

[MRBB02] Marc E. McDill, Stephanie Rebain, Marc E. Braze, JanisMcDill, and Janis Braze. Harvest scheduling with area-based adjacency constraints. *Forest Science*, 48(4):631–642, 2002.

[Mur99] Alan Murray. Spatial restrictions in harvest scheduling. *Forest Science*, 45(1):45–52, 1999.

[NSB+07] Nicholas Nethercote, Peter J. Stuckey, Ralph Becket, Sebastian Brand, Gregory J. Duck, and Guido Tack. Minizinc: Towards a standard CP modelling language. In Christian Bessiere, editor, *Principles and Practice of Constraint Programming - CP 2007, 13th International Conference, CP 2007, Providence, RI, USA, September 23-27, 2007, Proceedings*, volume 4741 of *Lecture Notes in Computer Science*, pages 529–543. Springer, 2007.

[PFL16] Charles Prud'homme, Jean-Guillaume Fages, and Xavier Lorca. *Choco Solver Documentation*. TASC, INRIA Rennes, LINA CNRS UMR 6241, COSLING S.A.S., 2016.

[RM18] Jean-Charles Régin and Arnaud Malapert. Parallel constraint programming. In Youssef Hamadi and Lakhdar Sais, editors, *Handbook of Parallel Constraint Reasoning*, pages 337–379. Springer, 2018.

[RvBW06] Francesca Rossi, Peter van Beek, and Toby Walsh, editors. *Handbook of Constraint Programming*, volume 2 of *Foundations of Artificial Intelligence*. Elsevier, 2006.

[STL19] Christian Schulte, Guido Tack, and Mikael Z. Lagerkvist. Modeling. In *Modeling and Programming with Gecode*. Christian Schulte and Guido Tack and Mikael Z. Lagerkvist, 2019. Corresponds to Gecode 6.2.0.

UNIVERSIDADE DE ÉVORA
INSTITUTO DE INVESTIGAÇÃO
E FORMAÇÃO AVANÇADA

Contactos:
Universidade de Évora
Instituto de Investigação e Formação Avançada — IIFA
Palácio do Vimioso | Largo Marquês de Marialva, Apart. 94
7002 - 554 Évora | Portugal
Tel: (+351) 266 706 581
Fax: (+351) 266 744 677
email: iifa@uevora.pt