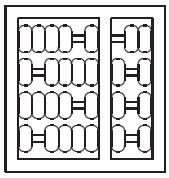


INSTITUTO DE COMPUTAÇÃO
UNIVERSIDADE ESTADUAL DE CAMPINAS

A Hybrid Heuristic for Forest Harvest and Transportation Problems

Arnaldo Vieira Moura
Rafael Augusto Scaraficci

Technical Report - IC-07-34 - Relatório Técnico

November - 2007 - Novembro

A Hybrid Heuristic for Forest Harvest and Transportation Problems

Arnaldo Vieira Moura * Rafael Augusto Scaraficci †

Abstract

This work treats harvest and transportation planning problems stemming from the daily operation of some large pulp and paper companies. The problem consists in scheduling harvest and transportation tasks for a planning horizon of around one year. It comprises a large and complex set of constraints, including constraints related to the structure of the harvest areas, structure and capacity of the harvest teams, the weather, and some properties of the wood logs. We developed a hybrid algorithm that has three phases, namely, construction, local search and post-optimization. The first two are based on a GRASP metaheuristic and are used to generate a pool of elite solutions. In the third phase, a relaxed linear model is generated for each solution in the pool, and the solution of these models are converted into valid solutions. The algorithm was tested with real data and proved to be an adequate approach to treat this problem. Although some restrictions are specific for this problem, the same ideas can inspire similar approaches for solving harvest and transportation planning problems in other situations.

1 Introduction

The sustainable potential production of Brazilian forest plantations is estimated to be around 390 million m^3/year. Out of this total, 27% refer to pine timber and 73% are eucalypt timber. The major consuming segment is the pulp and paper industrues (30%) followed by pig-iron and steel industries (22%), and sawmilling (19%). In 2006, the planted forest generated a gross revenue of about U$ 4 billion, and the gross product value generated by all the timber industrial transformation chain exceeded U$ 36 billion. The pulp and paper segment, together with the solidwood industry, responded for 71% of the total gross product value (Abraf, 2007).

The industry needs to make decisions in both the long term and the short term in order to use the planted areas in a sustainable way, while maximizing their revenues. For many such forest planning problems, there are no suitable optimization tools available, and such problems are thus approached manually. However, due to the dimensions and the complexity of the instances, managers are not always able to make the best decisions.

*Instituto de Computação, Universidade Estadual de Campinas, 13081-970 Campinas.

†Instituto de Computação, Universidade Estadual de Campinas, 13081-970 Campinas, SP. Research supported by FAPESP grant BRL 31,680.

This work treats planning and scheduling of short-term forest harvesting and transportation operations from the perspective of some large Brazilian pulp and paper industries. It consists in determining, for each day in a planning horizon of around one year, where and how long each harvest team should work, and how much wood should be transported from each harvest area to the pulp and paper factory, while satisfying a set of operational and labor requirements. This is a hard scheduling problem, and, in Brazil, it is usually solved manually, requiring expert professionals and many hours of work.

We propose a hybrid algorithm to solve this problem. It is based on some principles of the GRASP metaheuristic (Resende and Ribeiro, 2003; Festa, 2003) and on a relaxed linear model (Bazaraa et al., 1990; Gass, 1984). We chose a GRASP metaheuristic due to the many successes of this technique when used to solve hard optimization problems, and in particular scheduling problems (Festa and Resende, 2002). Our approach comprises three phases, namely, construction, local search and post-optimization. The construction and local search procedures are repeated by a given amount of time, generating a pool of elite solutions. Next, a relaxed linear model is generated for each solution in the pool. These linear models are solved, and the relaxed solutions are converted into valid ones. The algorithm was tested with real data and it proved to be an adequate approach to treat this problem.

Despite some restrictions that are specific to our problem, we believe that the ideas introduced in this work can inspire similar approaches for solving forest harvesting and transportation problems in other scenarios.

The work is organized as follow. Section 2 describes the problem. Section 3 presents a review of the literature. Section 4 introduces a representation model for the problem, and specifies the neighborhoods and the objective function. Section 5 describes the proposed algorithm. Section 6 shows the computation results for some real instances, and Section 7 offers some concluding remarks.

2 Problem Description

We consider an annual harvest and transportation planning problem. The problem starts with a list of harvest blocks, marked as suitable for harvesting during the next 12 months. Each harvest block is a large planted area, and it is decomposed in a set of smaller areas, called harvest units. Each harvest unit is unique, and has its own properties, like the estimated wood volume, average size of the trees, average wood density, and average wood quality index [†]. The units must be felled, and the time to harvest a unit depends on the volume, the size of the trees, and the equipment used by the harvest teams.

Although, the units can be harvested by any team in any period of the year, transportation should be avoided in some areas during parts of the year, due to rains and soft ground. The harvest blocks are classified according to the following levels with respect to transportation difficulty:

0 - the logs can be transported in any period of the year;

[†]The wood quality index is correlated with the pulp yield.

1 - the logs should be transported only during the dry or light-rainy months;

2 - the logs should be transported only during the dry months.

A harvest team works a number of hours per day, which can vary, within a known range, from one week to another. The number of work hours in each day must be the same for all teams. In certain specified periods, the teams cannot work due to equipment maintenance procedures. Each team consists of a harvester and a forwarder. The harvester fells and bucks the trees and the forwarder collects the logs and moves them to pick-up points adjacent to the forest roads. There are two types of harvest teams. An independent harvest team has a support team, which is responsible for transporting the workers, the equipment, and the fuel from the factory to the harvest areas. A dependent harvest team does not have a support team; consequently, it must work associated with an independent team at all times. It is not allowed to have more than one dependent team associated with an independent team. An initial association between the teams is established by fixing a starting point, which determines where each team must start to harvest. Although it is possible to change this association along the planning horizon, it is best to maintain it.

Two harvest teams cannot work in the same harvest unit, and when a team starts to harvest a unit, it cannot move to another unit until the former unit is totally harvested. As soon as a team finishes harvesting an unit, it must start harvesting another one. Hence, a team can harvest different units in the same period. Two independent teams cannot work in the same harvest block, and when an independent team starts to harvest a block, it cannot move to another block until the former block is totally harvested. A dependent team must work in a harvest block where an independent team is working, and it cannot move to another block until all units in the same block are totally harvested. If a team finishes harvesting the last but one unit of a block and other team has not finished the last one, then both teams will have to work together in the last unit of the block. Only in this last case, two teams can work in the same harvest unit.

The distance from block to block can be hundred of kilometers, so it is desirable to minimize the total distance covered by the teams. Distances between the units of a same block are ignored.

All harvested logs have to be transported to a pulp and paper factory. The logs should wait for a certain period of time in the harvest areas before they are transported. The waiting time depends on how rainy was the period when the logs were harvested. If they were harvested in the dry season, they should wait less time than if they were harvested in the rainy season. The logs of a unit are transported following their harvest order. Furthermore, all logs of a harvest unit have to be transported before some logs of another unit harvested by the same team start to be transported. All logs of a harvest block have to be transported before some logs of another block harvested by the same independent team, start to be transported.

The factory requires a daily volume of logs. This volume can be satisfied both with harvested logs and with some logs bought from other companies. The daily volumes of logs provided by other companies and the properties of these logs are previously known. The logs that arrive daily at the factory should satisfy the following requirements:

- the average wood density of the logs that arrive at the factory in a certain day should not vary by more than a determined percentage from the previous day;
- the average wood quality index of the logs that arrive at the factory in a certain day should not be greater than a determined threshold.

The factory can be shut down at some periods for maintenance work. The periods when the factory is not operational are known. During these periods, the factory cannot receive logs.

2.1 Constraint Summary

The constraints of the problem can be divided into two sets. The first one is called the set of hard constraints, and it includes the following:

HC1: the daily volume of logs required by the factory should be met;

HC2: when a team starts to harvest a unit, it cannot move to another one until the former unit is totally harvested;

HC3: when a team starts to harvest a block, it cannot move to another one until all units in that block are harvested;

HC4: as soon as a team finishes harvesting a unit, it must start harvesting another one;

HC5: two teams cannot work in the same unit unless it is the last unit of the block;

HC6: two independent teams cannot work in the same block;

HC7: a dependent team must work in a block where a independent team is working;

HC8: an independent team cannot be associated with more than one dependent team;

HC9: all logs of a harvest unit have to be transported to the factory before some logs of another unit, which was harvested by the same team, start to be transported;

HC10: all logs of a harvest block have to be transported to the factory before some logs of another block, which was harvested by the same independent team, start to be transported;

HC11: the logs of a unit must be transported in the order they were harvested.

In a feasible solution, all hard constraints must be satisfied. The second set is the set of soft constraints, and it includes the following:

SC1: the logs of a harvest block should be transported during the appropriate months of the year, in order to avoid bad conditions of some roads during rainy months;

SC2: the average wood quality index of the logs that arrive at the factory in a certain day should not be greater than a determined threshold;

SC3: the average wood density of the logs that arrive at the factory in a certain day should not vary by more than a determined percentage from the previous day;

SC4: the total distance covered by the teams should be kept to a minimum;

SC5: the harvested logs should wait an appropriate amount of time on the ground before being transported to the factory;

SC6: the initial association between independent and dependent teams should be kept.

The number of soft constraints violated should be minimized. Hence, a practical solution must not violate any hard constraints and should strive to satisfy all soft ones.

3 Literature Review

It is common to divide the forest-planning process into a hierarchy of strategic, tactical and operational plans. The objective of this hierarchy is to decompose the forest-planning process into smaller and less complex planning tasks, with restricted goals. The hierarchical planning paradigm is described in depth by Gunn (1991). The author includes a table of definitions for these plans. Here, it is shown as Table 1. All plans have a common starting point, the current period. However, they differ in some aspects such as their objectives, scopes, and levels of details.

Characteristics	Strategic Planning	Tactical Planning	Operational Control
Objective	Resource	Resource acquisition	Execution utilization
Time Horizon	Long	Middle	Short
Level of Management	Top	Middle Middle	Low Short
Scope	Broad	Medium	Narrow
Source of Information	External Internal &	External Internal &	Internal
Level of Detail	Highly Aggregate	Moderately Aggregate	Very Detailed
Degree of Uncertainty	High	Moderate	Lows
Degree of Risk	High	Moderate	Low

Table 1: Characteristics of decision problems in hierarchy (Gunn, 1991)

According to Gunn (1991), the strategic forest planning problem consists of making long-term decisions about the natural resources that will be available to the company. In fact, the term strategic forest planning also refers to determining the management regimes and which areas within the forest should be harvest in each period (Mitchell, 2004). This latter approach is similar to the tactical forest planning problem described by Gunn (1991),

but it consideres a long-term rather than a medium-term horizon. The strategic time horizon usually varies between 20 to 100 years, and the time periods between 1 to 10 years. The land units involved in strategic planning are highly aggregated areas based on similar silviculture, yield, and age. Strategic forest planning problems are commonly solved using linear program models (Johnson and Scheurman, 1977; Weintraub et al., 1986).

The tactical forest planning problem consists in making more effective use of the company and forest resources in the medium-term. The time horizon usually varies between 1 to 10 years, and the time periods between 1 to 12 months. In this level of planning, land units are less aggregated than the units used in strategic forest planning. The objective of tactical forest planning consists in determining the areas that should be harvested in each planning period, and the roads that should be built before harvesting the areas. These problems can also include environmental constraints (Brumelle et al., 1998; McDill and Braze, 2001), which limit the maximum size of clear-cut areas that are permissible in a forest, and harvest team allocation decisions (Karlsson et al., 2004). Basically, two approaches have been applied to solve tactical forest planning problems: mixed integer programming (McDill and Braze, 2001; Karlsson et al., 2004) and heuristics (Brumelle et al., 1998; Murray and Church, 1995; Boston and Bettinger, 2002). Due to the combinatorial nature of these problems, a straightforward mixed integer programming is only recommended when dealing with small or moderate sized problems. Also, the mixed integer programming approach can be strengthened by including lifting, decomposition and column generation in order to solve larger problems (Weintraub et al., 2000).

The operational forest planning problem consists in making decisions in the short-term, with the time horizon varying between 4 to 52 weeks, and the time periods between 1 to 7 days. A smaller land unit, called a harvest unit, is used in this kind of planning. The main objective is to sequence the harvest in time, and to decide which teams should work on each harvest unit. In addition, it can include decisions about what log-types are obtained and which customers are supplied. These problems are solved using heuristics techniques (Murphy, 1998; Souza, 2004), and, in some cases, mathematical models are applied to small to medium size problems, or may be included as part of a heuristic procedure (Karlsson et al., 2003; Mitchell, 2004).

Table 2 compares some typical operational forest planning problems with the problem treated in this work. Our problem has a larger dimension (365 periods) than the problems reported in the literature. This fact only highly increases the complexity of the problem, and justifies a heuristic approach. Furthermore, it also considers the direct influence of rain conditions in the harvesting and transportation planning.

Almost all other works in the literature consider some kind of transportation constraints, but not all consider the movements of a harvest team from one area to another during the same period. In this work, we present an alternative treatment of these constraints by making use of permutation techniques and a new and adequade decode function.

Unlike other works, we do not treat different log types, because all logs in our instances are used to produce pulp and paper. However, the logs that arrive at the factory should satisfy some constraints associated with their wood quality index and their density.

Characteristics	Murphy (1998)	Karlsson et al. (2003)	Souza (2004)	Mitchell (2004)	Our work (2007)
Planning Horizon	1 time unit	4–6 weeks	52 weeks	4–8 week	52 weeks
Planning Period	1 time unit	5 days	1 week	1 week	1 day
Weather Constraints					√
Transportation Constraint		√	√	√	√
Movement Within a Period		√		√	√
Log types	√	√	√	√	
Solution Method	tabu search	restricted mip	genetic algorithm	mip, column generation, constraint branch	grasp, lp

Table 2: A comparison of operational forest planning models

4 Modeling the Problem

This section describes a model for the problem, which ensures that most of the hard constraints are not violated. Also, it considers three different types of movements, each one giving rise to different neighborhoods, and an objective function that is based on penalties.

4.1 Solution Representation

A solution representation to the problem is based on three components. The first component is a permutation $\mathcal{P}$ of harvest blocks, which are themselves permutations of harvest units. Each $\mathcal{P}[i][j]$ represents the j-th harvest unit of the i-th harvest block of the permutation $\mathcal{P}$. The second component, is a vector $\mathcal{H}$ of real numbers, where each element $\mathcal{H}[t]$ establishes the number of hours that the teams should work during day t. The third component is a decode function that sets, for each day, where each team should work and how much wood should be transported from each harvest area to the factory.

The decode function assigns the harvest areas to the harvest teams by following their order in the permutation $\mathcal{P}$ and respecting the constraints **HC2** − **HC8**, and **SC6**. See Section 2.1. Also, it marks the logs to be transported following their harvest order. This assignment rule guarantees that constraints **HC9** − **HC11** are not violated. The only hard constraint that does not have its satisfiability ensured is constraint **HC1**. However, in practice, it is difficult to violate this constraint because the harvest process begins 30–40 days before the transportation process.

The decode function generates a harvest and a transportation plan, which are, respectively, represented by two lists of tasks, namely, $\mathcal{L}_{hp}$ and $\mathcal{L}_{tp}$. Each element $l \in \mathcal{L}_{hp}$ is a quadruple (i, v, t, k), where i identifies the harvest unit, v is the volume of logs that should be harvested, t is the time when the logs should be harvested, and k is the team that should harvest those logs. Each element $l \in \mathcal{L}_{tp}$ is a quadruple (i, v, t, t'), where i identifies the

harvest unit, v is the volume of logs that should be transported, t is the time when the logs should be harvested, and t' is the time when the logs should be transported to the factory. The pseudo-code of the decode function is presented as Algorithm 1, where:

- b_k identifies the harvest block that is being assigned to team k;
- u_k identifies the harvest unit that is being assigned to team k;
- h_k is the number of work-hours that has not been allocated to team k;
- h'_k is the number of work-hours that needs to be allocated to team k in order to finish harvesting unit u_k;
- d_t is the demand of the factory for their own harvested logs during a day t;
- *GetVolume*(j, k, h) returns the average volume of logs that team k will harvest in the unit j if it works h hours there;
- *GetWorkHours*(j, k) returns the number of work-hours that needs to be allocated to team k to finish harvesting unit j;
- *UpdateVariables*(j, k, h) updates $h'_{k'}$, where k' is a team associated with team k, which has also been allocated to harvest unit j. The variable $h'_{k'}$ will be set to h hours if k has the same productivity as k', or to a value directly propotional to their productivities;
- *GetNextAvailableBlock*$(\mathcal{P}, k)$ returns the next available block in the permutation $\mathcal{P}$. For a team k that is not associated with any other team, the next available block will be the first block in the permutation $\mathcal{P}$ that has not been allocated to any other team. For a team k that is associated with a team k', the next available block will be the block allocated to team k' if its units have not been totally allocated; otherwise, it will be first block in the permutation $\mathcal{P}$ that has not been allocated to any other team. If there is no next available block, it will return an invalid block identifier;
- *GetNextAvailableUnit*$(\mathcal{P}, i, k)$ returns the next available unit in the i-th block of the permutation $\mathcal{P}$. The next available unit is the first unit that has not been allocated to any other team. If there is no unallocated unit in the i-th block, and team k is associated with a team k' that is being allocated to the j-th unit in the i-th block, this function will return $\mathcal{P}[i][j]$; otherwise, it will return an invalid unit identifier.

In Algorithm 1, lines 2–30 and 31–43 generate, respectively, a set of harvest tasks and transportation tasks for a day t. Every time that a harvest task is generated (lines 5–20), special processing is undertaken if the unit has been allocated to two teams. In this case, it is necessary to update the value of some variables to avoid assigning non-existent trees to be harvested (lines 10–12 and 18–20). The initial association between the teams is controlled by the functions *GetNextAvailableBlock* and *GetNextAvailableUnit* which ensure that the associated teams will work in the same harvest blocks.

Algorithm 1 Decode Function

Require: a permutation of areas $\mathcal{P}$, a vector of work-hours $\mathcal{H}$

for $t := 1$ to T **do**
  for $k := 1$ to K **do**
    $h_k := \mathcal{H}[t]$
    while $h_k > 0$ and u_k is a valid harvest unit **do**
      if $h'_k > 0$ and $h_k > h'_k$ **then**
        $h_k := h_k - h'_k$
        $h'_k := 0$
        insert $(u_k, GetVolume(k, u_k, h'_k), t, k)$ at the end of $\mathcal{L}_{hp}$
        insert $(u_k, GetVolume(k, u_k, h'_k), t, k)$ at the end of $\overline{\mathcal{L}}_{hp}$
        if u_k is shared **then**
          $UpdateVariables(u_k, k, h'_k)$
        end if
      else if $h'_k > 0$ and $h'_k > h_k$ **then**
        $h'_k := h'_k - h_k$
        $h_k := 0$
        insert $(u_k, GetVolume(k, u_k, h_k), t, k)$ at the end of $\mathcal{L}_{hp}$
        insert $(u_k, GetVolume(k, u_k, h_k), t, k)$ at the end of $\overline{\mathcal{L}}_{hp}$
        if u_k is shared **then**
          $UpdateVariables(u_k, k, h'_k)$
        end if
      else if b_k has not been totally harvested **then**
        $u_k := GetNextAvailableUnit(b_k)$
        $h'_k := GetWorkHours(u_k, k)$
      else
        $b_k := GetNextAvailableBlock(\mathcal{P}, k)$
        $u_k := GetNextAvailableUnit(b_k)$
        $h'_k := GetWorkHours(u_k, k)$
      end if
    end while
  end for
  while $\overline{\mathcal{L}}_{hp}$ is not empty and $d_t > 0$ **do**
    l:= remove the first element of $\overline{\mathcal{L}}_{hp}$
    if $l.v > d_t$ **then**
      $l.v := l.v - d_t$
      insert $(l.i, d_t, l.t, t)$ at the end of $\mathcal{L}_{tp}$
      insert l at the front of $\overline{\mathcal{L}}_{hp}$
      $d_t := 0$
    else
      $d_t := d_t - l.v$
      insert $(l.i, l.v, l.t, t)$ at the end of $\mathcal{L}_{tp}$
      $l.v := 0$
    end if
  end while
end for
return a harvest plan $\mathcal{L}_{hp}$ and a transportation plan $\mathcal{L}_{tp}$

4.2 Neighborhoods

In the algorithm discussed later, three types of movements are considered, each one giving rise to different neighborhoods. The first type of movement, called a *change work-hours* movement, consists of an increment or decrement in the number of hours that the teams should work in each day of a certain week. It is represented by a pair (w, δ), where w identifies the week and δ is a real value that is added to the work-hours of each day in the week w. This movement changes the values of some elements in vector $\mathcal{H}$. Two precautions must be taken with this type of movement. The first one is that $\mathcal{H}$ is a vector of work-hours per day, so it is necessary to convert the week w in days in order to get the right values for $\mathcal{H}$. The second one is that the increment or decrement applied to each day must not exceed the work hour's limits. We guaranteed this by checking the movement before applying it. The set of solutions generated by the execution of all *change work-hours* movements over a solution S defines a neighborhood to the problem, here called $N_1(S)$.

The second type of movement, called an *harvest unit swap* movement, consists of a change of position between two harvest units of the same block in the permutation $\mathcal{P}$. It is represented by the triple (b, u_1, u_2), where b identifies a block in the permutation $\mathcal{P}$, and u_1 and u_2 identify two distinct harvest units in block b. The set of solutions generated by the execution of all *harvest unit swap* movements over a solution S defines a second neighborhood to the problem, here called $N_2(S)$.

The last type of movement, called an *harvest block swap* movement, consists of the change of positions of two harvest blocks in the permutation $\mathcal{P}$. It is represented by a pair (b_1, b_2), where b_1 and b_2 identify two distinct blocks in the permutation $\mathcal{P}$. The set of solutions generated by the execution of all *harvest block swap* movements over a solution S defines a third neighborhood to the problem, here called $N_3(S)$.

4.3 Objective Function

A solution S is evaluated according to the following objective function that should be minimized:

$$\begin{aligned} f_{obj}(S) =& f_{HC1}(S, \alpha_{HC1}) + f_{SC1}(S, \alpha_{SC1}) + f_{SC2}(S, \alpha_{SC2}) + f_{SC3}(S, \alpha_{SC3}) + \\ & f_{SC4}(S, \alpha_{SC4}) + f_{SC5a}(S, \alpha_{SC5a}) + f_{SC5b}(S, \alpha_{SC5b}) \end{aligned}$$

The term f_{HC1} is associated with constraint **HC1**, and it penalizes the volume of logs not delivered on time by a factor of α_{HC1}. The term f_{SC1} is associated with constraint **SC1**, and it penalizes the volume of logs not delivered in the appropriated periods by a factor of α_{sc1} times how far it is from an appropriate period. The terms f_{SC2} and f_{SC3} are, respectively, associated with constraints **SC2** and **SC3**, and they penalize how far the wood quality index and the average density of the logs are from the desired values by a factor of α_{SC2} and α_{SC3}, respectively. The term f_{SC4} is associated with constraint **SC4**, and it penalizes each kilometer covered by the teams by a factor of α_{SC4}. The terms f_{SC5a} and f_{SC5b} are associated with constraint **SC5**, and they penalize, respectively, how many periods before or after the appropriate time waiting window the logs are delivered. Note

that the constraints not considered in the objective function are automatically satisfied by the solution representation model. See Section 4.1. The constants α_{HC1}, α_{SC1}, α_{SC2}, α_{SC3}, α_{SC4}, α_{SC5a}, and α_{SC5b} are chosen to satisfy the condition: $\alpha_{HC1} >> \alpha_{SC1} >> \alpha_{SC2} > \alpha_{SC3} > \alpha_{SC4} >> \alpha_{SC5a} > \alpha_{SC5b}$. This condition reflects the importance of not violating each type of constraint in a final solution. Obviously, solution S is feasible if and only if $f_{HC1}(S, \alpha_{HC1}) = 0$.

The objective function uses the harvest and transportation plans, which were generated by the decode function, to calculate the solution's penalties. It basically checks the plans and penalizes the violated constraints, according to their weights. When a movement $\gamma \in N_1(S) \cup N_2(S) \cup N_3(S)$ is executed over a solution S, a new solution S' is generated with partially modified harvest and transportation plans. In fact, movement γ only affects the plans from a determined planning period onwards. For example, suppose that γ is a *change work-hours* movement that modifies the work-hours of the days of the tenth week. In this case, the harvest and transportation plans of the solution S' will be same as the plans of the solution S for the first nine weeks. This is also valid for the other two kind of movements. Therefore, when a movement is executed over a solution, the decode function and the objective function can be quickly recalculated because it is not necessary to recalculate the part of the plans that remains unchanged.

5 A Hybrid Heuristic Approach

The hybrid algorithm we propose is based on some principles used in a typical GRASP metaheuristic (Resende and Ribeiro, 2003; Festa, 2003) and on a specific relaxed linear model (Bazaraa et al., 1990; Wolsey, 1998). It comprises three phases: construction, local search and post-optimization. Initially, we use a partially greedy procedure to build a solution, which is refined by means of a particular local search procedure. These two phases are repeated by a certain period of time, and the best solutions are stored in a pool. Then, a relaxed linear model is generated for each solution in the pool. These linear models are solved, and the relaxed solutions are converted into valid ones. The algorithm returns the best solution obtained over the three phases.

The pseudocode of the algorithm is presented as Algorithm 2. The algorithm is multi-thread, so it can take advantage of multi-core machines increasing the speedup. In lines 1–2, the cost of the solution S^* is set to infinity and the pool $\mathcal{E}$ is initialized. The code between lines 3 to 15 are run in n different threads. In each thread, the variable *time_counter*, which measures the execution time of the first two phases, is initialized, and then, iteratively, a solution is constructed and improved by a local search procedure. These two procedures are repeated while the *timer_counter* is less than the *time_threshold*, while the best *pool_size* solutions are stored in the pool $\mathcal{E}$ (lines 5–14). One thread at a time updates the solution S^* and the pool $\mathcal{E}$. Then the post-optimization phase is initialized (lines 16–23), where, for each solution $S \in \mathcal{E}$, a relaxed linear model $\mathcal{M}$ is generated and solved, and the relaxed solution is converted into a valid solution S'. Finally, the best overall solution S^* is returned.

Algorithm 2 Hybrid Heuristic

Require: *time_threshold* > 0, *pool_size* > 0, $n > 0$
1: $f_{obj}(S^*) := \infty$
2: $\mathcal{E} := \emptyset$
3: start n threads
4: start *timer_counter*
5: **repeat**
6: $S := Construction()$
7: $S := LocalSearch(S)$
8: lock access
9: **if** $f_{obj}(S) < f_{obj}(S^*)$ **then**
10: $S^* := S$
11: **end if**
12: update pool $\mathcal{E}$ to contain the best *pool_size* solutions
13: unlock access
14: **until** *timer_counter* $>$ *time_threshold*
15: finalize the threads
16: **for all** $S \in \mathcal{E}$ **do**
17: generate a relaxed linear model $\mathcal{M}$
18: resolve the relaxed linear model $\mathcal{M}$
19: convert the relaxed solution into a valid solution S'
20: **if** $f_{obj}(S') < f_{obf}(S^*)$ **then**
21: $S^* := S'$
22: **end if**
23: **end for**
24: **return** S^*

5.1 The Construction Procedure

A solution is constructed in two passes. Initially, a vector $\mathcal{H}$ of work-hours is filled in a semi-random way that prioritizes the amount of time close to the average number of hours that each team needs to work in order to supply the demand of the factory. Then, the permutation $\mathcal{P}$ is constructed, also in a semi-random way. Here, we seek to schedule the harvest blocks within a time window that allows the logs to be transported to the factory during the appropriate periods, while avoiding soft-ground.

The pseudocode of the algorithm is presented as Algorithm 3. Line 1 calculates the average number of hours ϕ that each team should work in each day in order to meet the factory demand. The value of ϕ is given by:

$$\phi = \frac{\Delta}{\Delta'} \times \frac{T_1 + \ldots + T_k \ldots + T_K}{K^2} \times \frac{1}{\overline{T}},$$

where Δ is the total demand of the factory, Δ' is the volume of logs avaiable to harvest, T_k is the total number of hours that team k would spend in order to harvest all areas, and $\overline{T}$ is the number of days that the teams work. We assume that $\Delta' \geq \Delta$. In lines $2-3$, the tolerated time window tw, is divided into $(1 + \left\lceil \frac{|\mathrm{tw}|}{\delta} \right\rceil)$ points of time. Each point δ_i is associated with a probability p_i that decays by a linear factor of ρ as it moves away δ hours

from the point closest to ϕ. For example, suppose that the time window varies from $2X$ to $8X$ hours and $\delta = X$ hours. We would have the following points: $\delta_1 = 2X$, $\delta_2 = 3X$, $\delta_3 = 4X$, $\delta_4 = 5X$, $\delta_5 = 6X$, $\delta_6 = 7X$, and $\delta_7 = 8X$. If δ_5 is the point closest to ϕ, the probability associated with the points will be: $p_1 = p - 4\rho$, $p_2 = p - 3\rho$, $p_3 = p - 2\rho$, $p_4 = p - \rho$, $p_5 = p$, $p_6 = p - \rho$, and $p_7 = p - 2\rho$. The value chosen for ρ must guarantee that each $p_i \geq 0$. For each $\mathcal{H}[t]$, where t is a period when the teams work, a value of work-hours is randomly chosen according to the probabilities p_i and respecting the fact that every day of a week must have the same number of work-hours.

In lines 14–23, the starting harvest units are inserted in the initial positions of the permutation $\mathcal{P}$. We consider that each team k needs more than one period to harvest its initial area. This condition is required by the decode function. The positions of the other harvest units in the allocated blocks are then shuffled.

Assume that g represents an independent team, or a pair of teams that should work in the same harvest block, and that G is the set of these elements. Line 24 updates h_g, which is the estimated time when group of teams g would finish to harvest its last allocated block. In lines 25–43, the remaining harvest blocks are inserted into the permutation $\mathcal{P}$. The list $\mathcal{L}_w$ contains the demand of the planning horizon, segmented according to the estimated weather conditons. List $\mathcal{L}_w$ is constructed by following the time line. Each element $l \in \mathcal{L}_w$ indicates the total demand Λ of its time segment and also the weather associated with it. In line 26 the first element $l \in \mathcal{L}_w$ is removed, and the next line estimates the minimum (h_{min}) and maximal (h_{max}) time to supply the demand of the segment l. The estimated times h_{min} and h_{max} depends on the weather. Consider that c_g is the average volume that group of teams g can harvest in an hour, h'_{min} and h'_{max} are, respectively, the minimal and maximal estimated times to meet the demand of the previous segment, Δ_ω is the remaining demand of the factory for logs that should be transported under weather $\omega \in \{$rainy, light-rainy, dry$\}$, Δ_ϖ is the remaining volume of logs not marked for transporation that are in areas with transportation level of difficulty $\varpi \in \{0, 1, 2\}$, and N_ω is the number of remaining segments with weather $\omega \in \{$rainy, light-rainy, dry$\}$. If the weather is rainy the estimated times are given by the equations:

$$h_{min} = h'_{max} + \frac{\Lambda}{\sum_{g \in G} c_g},$$

$$h_{max} = \begin{cases} \chi, & \text{if } \chi > 0, \\ h_{min}, & \text{otherwise}, \end{cases} \qquad \chi = h_{min} + \frac{\Delta_{light\text{-}rainy} + \Delta_{dry} - \Delta_1 - \Delta_2}{N_{rainy} \sum_{g \in G} c_g}.$$

If the weather is light-rainy and it follows a rainy season the estimated times are given by the equations:

$$h_{min} = h'_{min} + \frac{\Lambda}{\sum_{g \in G} c_g},$$

$$h_{max} = \begin{cases} \chi, & \text{if } \chi > 0, \\ \\ h_{min}, & \text{otherwise,} \end{cases} \qquad \chi = h_{min} + \frac{\Delta_{dry} - \Delta_2}{N_{light\text{-}rainy} \sum_{g \in G} c_g}.$$

If the weather is light-rainy and it follows a dry season the estimated times are give by the equations:

$$h_{min} = \begin{cases} \chi, & \text{if } 0 < \chi < h_{max}, \\ \\ h_{max}, & \text{otherwise,} \end{cases} \qquad \chi = \frac{-\Delta_{light\text{-}rainy} - \Delta_{dry} + \Delta_1 + \Delta_2 + \sum_{g \in G} c_g h_g}{\sum_{g \in G} c_g},$$

$$h_{max} = h'_{max} + \frac{\Lambda}{\sum_{g \in G} c_g}.$$

If the weather is dry the estimated times are give by the equations:

$$h_{min} = \begin{cases} \chi, & \text{if } 0 < \chi < h_{max}, \\ \\ h_{max}, & \text{otherwise,} \end{cases} \qquad \chi = \frac{-\Delta_{rainy} - \Delta_{light\text{-}rainy} + \Delta_0 + \Delta_1 + \sum_{g \in G} c_g h_g}{\sum_{g \in G} c_g},$$

$$h_{max} = h'_{min} + \frac{\Lambda}{\sum_{g \in G} c_g}.$$

In lines 28–39, harvest blocks are allocated until there are no blocks to be allocated or the following condition is met: $h_g \geq h_{min}, \forall g \in G$. A list of candidate blocks $\mathcal{C}$ is constructed based on the minimum time h_g and using only blocks with transportation levels of difficulty 0, 1 and 2 for rainy, light-rainy and dry seasons, respectively. If such a list $\mathcal{C}$ contains suitable harvest blocks, the cost of inserting block b in permutation $\mathcal{P}$ is given by:

$$g(b) = f(b, min(h_g), h_{min}, h_{max}, \mathcal{C} - \{b\}),$$

$$f(b, h_1, h_2, h_3, L) = \begin{cases} \left| \frac{h_2 + h_1 - h - t_g(b)}{2} \right| & \text{if } (h + t_g(b)) \geq h_1 \\ \\ \frac{\sum_{b' \in L} f(b', h, h_1, h_2, L - \{b'\})}{|L|}, & \text{otherwise ,} \end{cases}$$

where $t_g(b)$ is the average amount of time that the group of teams g needs to harvest block b.

Next, a restricted candidate list (RCL) is built, with its length determined by a parameter α. Parameter α is randomly chosen in the interval $[\mu_1, \mu_2]$, where $0 \leq \mu_1 < \mu_2 \leq 1$. An element (harvest block) belongs to the RCL if the following conditions are met:

$$\underline{b} = min\{g(b)|b \in \mathcal{C}\};$$

$$\overline{b} = max\{g(b)|b \in \mathcal{C}\};$$

$$RCL = \{b \in \mathcal{C}|g(b) \leq \underline{b} + \alpha(\overline{b} - \underline{b})\}.$$

A block $b \in RCL$ is chosen randomly. Then the position of its harvest units are shuffled and it is inserted at the end of permutation $\mathcal{P}$. Next, the estimated time h_g is updated. Finally, at the end of the process, if there are blocks still not allocated, they are randomly inserted at the end of the permutation $\mathcal{P}$.

5.2 The Local Search Procedure

The proposed local search procedure starts from a solution built by the constructive procedure and, in a mixed way, explores the three neighborhood sets defined for the problem. See Section 4.2. Let S be the current solution, γ be a movement, and let $(S \oplus \gamma)$ be the solution generated after the execution of movement γ over solution S. The local search algorithm works in the following way. In a first phase, it seeks for better solutions using *change work-hours* movements. Then, in a second phase, it uses *harvest unit swap* movements, and, in a third phase, it uses *harvest block swap* movements. These phases are repeated until a local minimum is reached. Since it is expensive to evaluate all the solutions in a given neighborhood in order to select the best one, we adopted the first improvement strategy, i.e. the algorithm always set $(S \oplus \gamma)$ as the new current solution when the condition $f_{obj}(S \oplus \gamma) < f_{obj}(S)$ is first met.

The movements are generated by traversing the vector $\mathcal{H}$ and the permutation $\mathcal{P}$ from left to right. So, a *change work-hours* movement that affects a week w will be evaluated before than a movement that affects week $(w + 1)$, and a *harvest unit swap* movement that swap units u_i and u_{i+n}, where u_i precedes u_{i+n}, will be evaluated before the movement that swaps units u_{i+1} and u_{i+n}, where u_i precedes u_{i+1} and u_{i+1} precedes u_{i+n}. The same reasoning applies to the *harvest blocks swap* movements.

Each phase of the algorithm consists in seeking for better solutions generated by traversing one of the components of $\mathcal{H}$ or $\mathcal{P}$, such an action depending on the type of the respective movement. In practice, this evaluation strategy worked very well, producing better results than a random evaluation of the movements. It approximates a result similar to an exhaustive evaluation of the search space, that is, selecting the best overall movement in each iteration.

The pseudocode of the algorithm is presented as Algorithm 4. It requires a solution S and an increment δ. In the first phase (lines 3–13), two *change work-hours* movements, γ and γ', are generated for a week w. The better of the two movements is applied to

Algorithm 3 Construction Procedure

Require: the starting points $\{(b_1, u_1), \ldots (b_k, u_k), \ldots (b_K, u_K)\}$
calculate the average amount of time ϕ
divide the time windows into points δ_i
assign for each point δ_i a probability p_i
for all week w **do**
  h:= random choose a point based on the probabilities p_i
  for all period $t \in w$ **do**
    if teams do not work in day t **then**
      $\mathcal{H}[t]$:= 0
    else
      $\mathcal{H}[t]$:= h
    end if
  end for
end for
for $k := 1$ to K **do**
  if block b_k has not been inserted into permutation $\mathcal{P}$ **then**
    set unit u_k as the first unit in the block b_k
    shuffle the positions of the units in b_k, but the first unit u_k
    insert block b_k at the end of permutation $\mathcal{P}$
  else
    set unit u_k as the second unit in $b_k \in \mathcal{P}$
    shuffle the positions of the units in $b_k \in \mathcal{P}$, but the first two units
  end if
end for
update h_g for $g \in G$
while $\mathcal{L}_w \neq \emptyset$ and there are block to be allocated **do**
  remove the first element l of the list $\mathcal{L}_w$
  determine the estimated times h_{min} and h_{max} to satisfy the demand Λ of l
  while $min(h_g) \geq h_{min}$ and there are block to be allocated **do**
    construct the candidate list $\mathcal{C}$ with the appropriate blocks
    calculate the cost of each $b \in \mathcal{C}$
    choose a random $\alpha \in [\mu_1, \mu_2]$
    $\underline{b} := min\{g(b)|b \in \mathcal{C}\}$
    $\overline{b} := max\{g(b)|b \in \mathcal{C}\}$
    $RCL := \{b \in \mathcal{C}|g(b) \leq \underline{b} + \alpha(\overline{b} - \underline{b})\}$
    choose a random block $b \in RCL$
    shuffle the units of the block b
    insert b at the end of $\mathcal{P}$
    $h_g := h_g + t_g(b)$ for minimal h_g
  end while
end while
if there are more blocks to be allocated **then**
  instert the blocks randomly at the end of permutation $\mathcal{P}$
end if

the current solution S if it is valid and the solution generated by it is better than the current solution. A *change work-hours* movement is valid if it does not produce a value

Algorithm 4 Local Search Procedure

Require: $S \neq \emptyset$, $\delta > 0$

1: **repeat**
2: $improved := false$
3: **for** $w := 1$ to W **do**
4: $\gamma := (w, +\delta)$
5: $\gamma' := (w, -\delta)$
6: **if** γ is valid and $f_{obj}(S \oplus \gamma) < f_{obj}(S)$ and ((γ' is valid and $f_{obj}(S \oplus \gamma) < f_{obj}(S \oplus \gamma')$) or γ' is invalid) **then**
7: $S := S \oplus \gamma$
8: $improved := true$
9: **else if** γ' is valid and $f_{obj}(S \oplus \gamma') < f_{obj}(S)$ **then**
10: $S := S \oplus \gamma'$
11: $improved := true$
12: **end if**
13: **end for**
14: **for** $j := 1$ to J **do**
15: **for** $i := 1$ to I_j **do**
16: **for** $i' := (i + 1)$ to I_j **do**
17: $\gamma'' := (j, i, i')$
18: **if** γ'' is valid and $f_{obj}(S \oplus \gamma'') < f_{obj}(S)$ **then**
19: $S := S \oplus \gamma''$
20: $improved := true$
21: **end if**
22: **end for**
23: **end for**
24: **end for**
25: **for** $j := 1$ to J **do**
26: **for** $j' := (j + 1)$ to J **do**
27: $\gamma''' := (j, j')$
28: **if** γ''' is valid and $f_{obj}(S \oplus \gamma''') < f_{obj}(S)$ **then**
29: $S := S \oplus \gamma'''$
30: $improved := true$
31: **end if**
32: **end for**
33: **end for**
34: **until** $improved = false$
35: **return** S

of work-hours outside the tolerated limits. In the second phase (lines 14–24), *harvest unit swap* movements are generated following the order of the areas in the permutation $\mathcal{P}$. The variables j, i and i' represents, respectively, the j-th block, the i-th unit in the j-th block, and the i'-th unit in the j-th block of the permutation $\mathcal{P}$.

A *harvest unit swap* movement γ'' is executed over the current solution S if it is valid and the solution $(S \oplus \gamma'')$ is better than the solution S. The movement γ'' is valid if it does not swap the initial harvest unit associated with each team.

In the third phase (lines 25–33), the *harvest block swap* movements are generated fol-

lowing the block order in permutation $\mathcal{P}$. The variables j and j' represent, respectively, the j-th and j'-th block in the permutation $\mathcal{P}$. A *harvest block swap* movement γ''' is executed over the current solution S if it is valid and the solution $(S \oplus \gamma''')$ is better than S. The movement γ''' is valid if it does not swap the initial harvest block associated with each team.

5.3 The Post-Optimization Procedure

The decode function generates a transportation plan based on a FIFO principle, i.e. it marks the logs to be transported following the order of the elements in the transportation list. See Section 4.1. This strategy is somewhat restricted; it does not have the flexibility to delay or anticipate the transport of some logs in order to reduce the violation of some constraints. In orde to ameliorate this, we enhanced the algorithm with a post-optimization phase that uses a relaxed linear model whose solution can be converted into a valid transportation plan. Although this phase does not guarantee improvement in the transportation plan (because the model is not exact), in practice the solutions generated in this phase proved to be much better. An exact model was not used because it proved hard to solve due to the large number of binary variables that were present in the model.

The idea is to generate relaxed transportation models for the best solutions sought in the first two phases. These models are solved using a linear solver, and then the relaxed solutions are converted into valid transportation plans. If a better solution is obtained it is returned by the algorithm. The best solutions are used to fix the periods when the logs of a harvest unit can be transported. For example, if in a solution S a harvest unit u is transported between day t and day $(t + n)$, in the relaxed linear model, then unit u also has to be transported during this time interval. However, the model does not fix how much wood should be transported in each day within this time window. A description of the model is presented in the sequel.

Consider the following sets, variables, and constants:

T**:** the set of time periods in the planning horizon;

Q**:** the set of time periods in the planning horizon where the factory's demand in the previous day is zero;

J**:** the set of harvest units with logs available to be transported;

K**:** the set of companies that provide third party logs to the pulp factory;

x_{jt}**:** the volume of wood that should be transported from harvest unit j to the pulp factory during period t;

$x_{jtt'}$**:** the volume of logs harvested during period t that should be transported from harvest unit j to the pulp factory during period t';

y_t**:** a measure of violation for constraint **SC2** during the time period t;

z_t**:** a measure of violation for constraint **SC3** during the time period t;

Δ_t**:** the pulp factory's demand of logs during the period t;

Γ**:** the wood quality index threshold;

Λ**:** the tolerated average wood density variation;

$\omega(j,t)$**:** a value equal zero if the logs of the harvest unit j can be transported during period t without violating constraint **SC1**, otherwise it is set to how far it is from an appropriate period;

$\varpi(t,t')$**:** the number of periods the logs harvested during day t are early if they are delivered during day t';

$\varpi'(t,t')$**:** the number of periods the logs harvested during the day t are late if they are delivered during day t';

θ_{kt}**:** the volume of logs provided by company k during the period t;

γ_j**:** the wood quality index of the logs of the harvest unit j;

γ_{kt}**:** the wood quality index of the logs provided by company k during day t;

λ_j**:** the average density of the logs of harvest unit j;

λ_{kt}**:** the average density of the logs provided by comapny k during day t;

$v(j)$**:** the total volume of logs avaiable to be transported in harvest unit j;

$v(j,t,t')$**:** the volume of logs in harvest unit j that were harvested during period t and that can be transported during period t';

$\alpha_{SC1}, \alpha_{SC2}, \alpha_{SC3}, \alpha_{SC5a}, \alpha_{SC5b}$: the weights associated with the respecive constraint violations.

Now the model can be formulated as:

$$\begin{aligned} min \quad & \alpha_{SC1} \sum_{t\in T}\sum_{j\in J} \omega(j,t) x_{jt} + \alpha_{SC2} \sum_{t\in T} y_t + \alpha_{SC3} \sum_{t\in T-Q} z_t + \\ & \alpha_{SC5a} \sum_{t\in T}\sum_{t'\in T} \varpi(t,t') x_{jtt'} + \alpha_{SC5b} \sum_{t\in T}\sum_{t'\in T} \varpi'(t,t') x_{jtt'}, \end{aligned} \tag{1}$$

subject to:

$$\sum_{j\in J} x_{jt} + \sum_{k\in K} \theta_{kt} = \Delta_t, \qquad t \in T, \tag{2}$$

$$\frac{\sum_{j\in J} \gamma_j x_{jt} + \sum_{k\in K} \gamma_{kt}\Theta_{kt}}{\Delta_t} - y_t \leq \Gamma, \qquad t \in T, \tag{3}$$

$$(1-\Lambda)\frac{\sum_{j\in J}\lambda_j x_{j(t-1)}+\sum_{k\in K}\lambda_{k(t-1)}\theta_{k(t-1)}}{\Delta_{(t-1)}} - \frac{\sum_{j\in J}\lambda_j x_{jt}+\sum_{k\in K}\lambda_{kt}\theta_{kt}}{\Delta_t} - z_t \leq 0, \qquad t \in T-Q, \tag{4}$$

$$-(1+\Lambda)\frac{\sum_{j\in J}\lambda_j x_{j(t-1)}+\sum_{k\in K}\lambda_{k(t-1)}\theta_{k(t-1)}}{\Delta_{(t-1)}} + \frac{\sum_{j\in J}\lambda_j x_{jt}+\sum_{k\in K}\lambda_{kt}\theta_{kt}}{\Delta_t} - z_t \leq 0, \qquad t \in T-Q, \tag{5}$$

$$x_{jt'} - \sum_{t\leq t'} x_{jtt'} = 0, \qquad t' \in T, j \in J, \tag{6}$$

$$\sum_{t\in T}\sum_{t\in T'} x_{jtt'} \leq v(j), \qquad j \in J \tag{7}$$

$$\sum_{t'\in T} x_{jtt'} \leq v(j,t,t'), \qquad t \in T, j \in J \tag{8}$$

$$x_{jt}, x_{jtt'}, y_t, z_t \geq 0, \qquad t \in T, t' \in T, j \in J \tag{9}$$

This model asks for minimization of the total cost associated with the violation of constraints **SC1**, **SC2**, **SC3**, and **SC5**, which are related to transportation planning. Constraint (2) specifies that the total volume of logs that arrive at the factory in a period t has to be equal to the factory's demand in that period. Constraint (3) measures the violation of constraints **SC2**. For each period $t \in T$, it verifies if the average wood quality index of the logs that arrive at the factory is less than a given threshold. Constraints (4) and (5) measure the violation of constraint **SC3**. For each period $t \in T - Q$, they verify, respectively, if the average density of the logs that arrive at the factory varies less or more than a percentage from the average density of the previous period. Constraints (8), (6), and (7) ensure that all logs scheduled to be transported from the harvest units to the factory do not exceed the volume of logs that were harvested and that are avaiable to be transported in each unit. Constraints (9) requires that all variables be non negative.

This model is relaxed because it does not require that the logs of a harvest unit be transported by following the order in which they were harvested (constraint **HC11**). For example, a relaxed solution can schedule some logs of a harvest unit that were harvested during period $(t+1)$ to be transported during period $(t+n)$ and schedule some logs of the same unit that were harvested during period t to be transported during period $(t+n+1)$.

To convert a relaxed solution into a valid solution that satisfies constraint **HC11**, we use the values of the variables x_{jt} of the relaxed solution. According to these values, another transportation plan is generated. This new plan will transport exactly a volume x_{jt} from harvest unit j during the period t, while respecting the harvest order of the logs in the unit. Based on this new plan, the objective function is recalculated.

6 Computational Results

This section reports on results of some computational experiments that were realized using the proposed algorithm. The experiments were carried out on an Intel quad-core 2.40 GHz machine, with 4GB of memory. The algorithm was coded in C++ and was compiled with GCC-4.0.0, using the compier option -O3. To solve the linear models, we used the Glpk-4.9 public domain solver.

6.1 The Test Instances

Two real instances from a large Brazilian pulp and paper company were used for the experiments. Some of their characteristics are summarized in Table 3. Both instances have the same planning horizon, which begins in November 30th and ends by December 31st of the next year. The harvesting starts 35 days before transportation starts and ends 21 days before transportation ends. The logs harvested during the rainy months (Jan, Feb, Nov, Dec), during the light-rainy months (Mar, Apr, May, Sep, Oct), or during the dry months (Jun, Jul, Aug) should wait, respectively, between 30 to 40 days, 25 to 32 days, and 20 to 27 days before being transported to the factory. The instances also have the same structure of harvest teams: two independents and one dependent team. The teams can work between 14 to 17 hours per day. Remember that the number of hours that the harvest teams work in a day must be the same for all days of the week.

In instance inst-1, the factory stops from April 4th to April 27th in order to do maintenance work. The harvest teams stop from March 15th to April 7th for the same reason. This instance has a list of 309 harvest units divided into 26 harvest blocks, and around 18% of the logs should be transported during the dry months, 30% during the dry and light-rainy months and 50% can be transported at any period. Around 6% of the logs processed by the factory are supplied by other companies. In instance inst-2, the factory stops for maintenance twice, from March 1st to March 9th and between November 1st and November 2nd. The teams do not stop at anytime. This instance has a list of 259 harvest units divided into 16 harvest blocks, and around 16% of the logs should be transported during the dry months, 38% during the dry and light-rainy months and 46% can be transported at any period. Around 8% of the logs processed by the factory are supplied by other companies.

6.2 The Experiments

We run the algorithm 30 times for each instance. The results are reported in Table 4. On these tests, we set the parameters as follows: $\delta = 0.5$ hour, $\mu_1 = 0.1$, $\mu_2 = 0.4$, $\alpha_{HC1} = 100$, $\alpha_{SC1} = 50$, $\alpha_{SC2} = 10$, $\alpha_{SC3} = 7$, $\alpha_{SC4} = 5$, $\alpha_{SC5a} = 0.1$, $\alpha_{SC5b} = 0.05$, $pool\_size = 10$, $n = 4$ threads, and $time\_threshold = 15$ minutes. The total running time of each execution was around 17 minutes, including all the heuristic phases.

The average objective value found for instance inst-1 was $125,809$ with a standard deviation of $1,557$, while for instance inst-2 it was $6,980$ with a standard deviation of 761. All solutions that were found were also feasible, and no one violated the soft-constrain **SC1**, i.e. all plans scheduled, the transportation of the logs within the appropriate months,

Characteristic	inst-1	inst-2
Harvest Begin	30-Nov-2004	30-Nov-2006
Harvest End	10-Dec-2005	10-Dec-2007
Transportation Begin	4-Jan-2005	4-Jan-2005
Transportation End	31-Dec-2005	31-Dec-2005
Rainy Months (Waiting Time)	Jan, Feb, Nov, Dec (30 – 40 days)	Jan, Feb, Nov, Dec (30 – 40 days)
Light-rainy Months (Waiting Time)	Mar, Apr, May, Sep, Oct (25 – 32 days)	Mar, Apr, May, Sep, Oct (25 – 32 days)
Dry Months (Waiting Time)	Jun, Jul, Aug (20 – 25 days)	Jun, Jul, Aug (20 – 25 days)
Factory's Maintainance Periods	4–27-Apr	1–9-May, 1–2-Nov
Teams' Maintainace Periods	15-Mar–7-Apr	–
N. of Harvest Blocks	26	16
N. of Harvest Units	309	259
Blocks with Difficulty 2	18%	16%
Blocks with Difficulty 1	30%	38%
Blocks with Difficulty 0	50%	46%
Logs from other companies	$0,6\%$	$0,8\%$
N. of Indep. Teams	2	2
N. of Dep. Teams	1	1
Work Hours	14–17	14–17

Table 3: Characteristics of the instances

Instance	Min. f_{obj}	Max. f_{obj}	Avg. f_{obj}	Median f_{obj}	St. Dev.
inst-1	123169	125809	125809	123500	1557
inst-2	5745	8715	6980	6918	761

Table 4: Results for run the algorithm 30 times for each instance.

avoiding the soft-ground conditions. The average displacement for a team varied from one instance to another due to the number of harvest blocks and their locations. In instance inst-1, the average displacement was 450km, and, in instance inst-2, it was 330 Km.

The difference between the average objective values for the two instances is, mainly, due to the properties of the starting points. In instance inst-1, the initial harvest units had values for the wood quality index that were greater than the wood quality index's threshold; consequently, it resulted in the application of a large amount of penalties that were intrinsic to the instance. This undesirable situation did not occur with instance inst-2.

Figures 1 and 2 show the behavior of the wood quality index and the logs' density variation, respectively, for instances inst-1 and inst-2. The horizontal lines delimit the ideal variation for these two quantities. Figures 3 and 4 indicate when the schedules did not respect the transportation waiting time, respectively, for instances inst-1 and inst-2. The upper chart indicates the maximum number of days that logs arrived before or after the

desired time windows, and the lower chart indicates the average volume of logs that arrived at the factory in that situation. The charts were ploted using information from those runs that resulted in objective values nearest to the average objective of all runs. All these figures show results when the post-optimization phase was and was not applied.

As can be observed from Figures 1, 2, 3, and 4, the post-optimization phase contributes significantly to improve the quality of the solutions. Mainly, it helps reduce violations of constraints **SC2** and **SC3** (see Figures 1 and 2). Some violations of constraint **SC5** were induced by the factory maintenance's periods (see Figures 3 and 4). That is, because some of the logs were harvested before and were transported after the maintenance periods and therefore had to wait on the ground more than the stipulated transportation window limit.

It should be remarked that these solutions were analysed by company engineers, who considered our schedules much superior than their manual plans.

7 Conclusions

This work treats a harvest planning problem from the perspective of large Brazilian pulp and paper companies. The problem comprises a large and complex set of constraints, including constraints related to the structure of the harvest areas, structure and capacity of the harvest teams, the weather conditions during the harvest months, and the properties of the logs to be harvested.

We developed a flexible representation for the problem using permutation of harvest areas, a vector of work-hours, and a new decode function. This representation satisfies most hard constraints and also allowed teams to move between harvest units within a planning period. These characteristics single out this model as a new hybrid approach to treat this kind of problems.

An algorithm was coded using a hybrid approach that comprises three phases: a construction, a local search, and a post-optimization phase. The first two are based on some principles of a typical GRASP metaheuristic and are used to construct a pool of elite solutions. The third phase consists in generating a relaxed linear model for each solution in the pool, solving these models, and converting the new solutions into valid ones. The algorithm was tested with real field data and proved to be a very adequate approach to treat this problem.

One possible improvement to the algorithm consists in exploring memory aspects, using the knowledge obtained in the previous iterations, in order to better guide the construction procedure. Another possibility is to enhance the algorithm by applying a path-relinking strategy, or a more robust local search procedure, like a tabu search.

References

Abraf (2007). *Abraf Statistical Yearbook 2007: base year 2006*. Brasilia, DF, Brazil.

Bazaraa, M. S., Jarvis, J. J., and Sherali, H. D. (1990). *Linear programming and network flows*. John Wiley & Sons, Inc., New York, NY, USA.

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

Brumelle, S., Granot, D., Halme, M., and Vertinsky, I. (1998). A tabu search algorithm for finding good forest harvest schedules satisfying green-up constraints. *European Journal of Operational Research*, 106(2):408–424.

Festa, P. (2003). Greedy randomized adaptive search procedures. *AIROnews*, 7(4):7–11.

Festa, P. and Resende, M. (2002). GRASP: An annotated bibliography. In Ribeiro, C. and Hansen, P., editors, *Essays and surveys in metaheuristics*, pages 325–367. Kluwer Academic Publishers.

Gass, S. I. (1984). *Linear programming: methods and applications*. McGraw-Hill, Inc., New York, NY.

Gunn, E. A. (1991). Some aspects of hierarchical production planning in forest management. In *Proceedings of the 1991 Symposium on Systems Analysis in Forest Resources*, pages 54–62.

Johnson, K. N. and Scheurman, H. L. (1977). Techniques for prescribing optimal timber harvest and investment under different objectives–discussion and synthesis. *Forest Science*, 18(1):1–31.

Karlsson, J., Rönnqvist, M., and Bergström, J. (2003). Short-term harvest planning including scheduling of harvest crews. *International Transactions in Operational Research*, 10(5):413–431.

Karlsson, J., Rönnqvist, M., and Bergström, J. (2004). An optimization model for annual harvest planning. *Canadian Journal of Forest Research*, 34(8):1747–1754.

McDill, M. E. and Braze, J. (2001). Using the branch and bound algorithm to solve forest planning problems with adjacency constraints. *Forest Science*, 43(3).

Mitchell, S. A. (2004). *Operational forest harvest scheduling optimisation*. PhD thesis, University of Auckland, New Zealand.

Murphy, G. (1998). Allocation of stands and cutting patterns to logging crews using a tabu search heuristic. *International Journal of Forest Engineering*, 9(1):31–37.

Murray, A. T. and Church, R. L. (1995). Heuristic solution approaches to operational forest planning problems. *OR Spectrum*, 17(2–3):193–203.

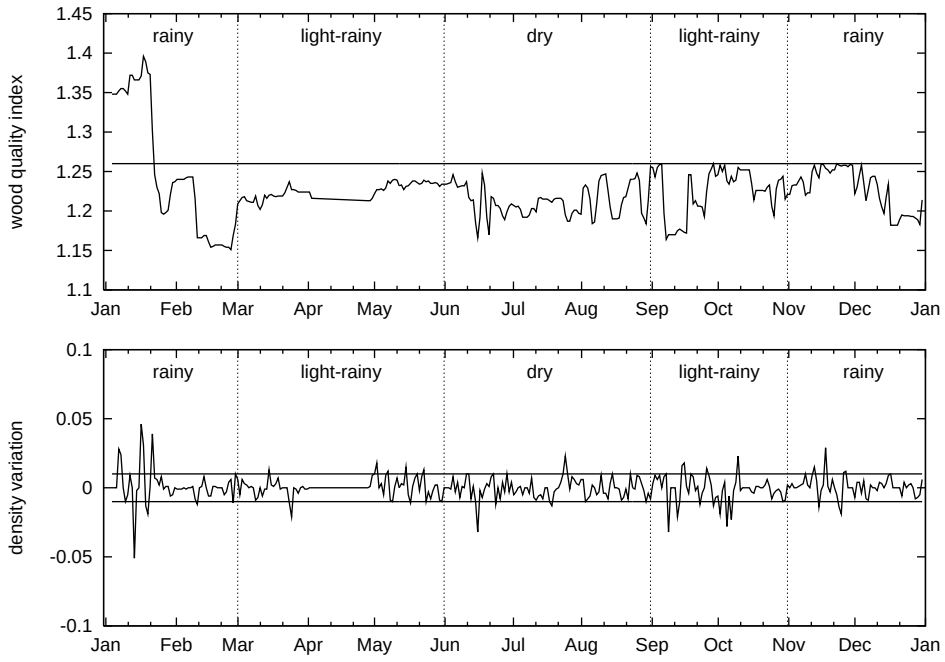
Resende, M. and Ribeiro, C. (2003). Greedy randomized adaptive search procedures. In Glover, F. and Kochenberger, G., editors, *Handbook of Metaheuristics*, pages 219–249. Kluwer Academic Publishers.

Souza, D. O. (2004). Algoritmos genéticos aplicados ao planejamento do transporte principal de madeira. Master's thesis, Universidade Federal do Paraná, Curitiba, Brazil. In Portuguese.

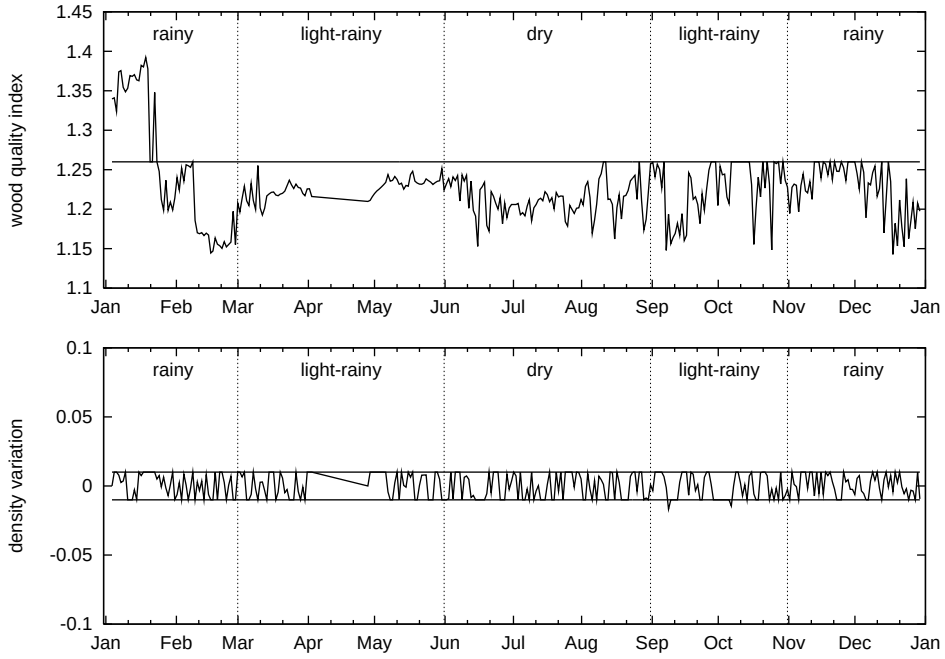
Weintraub, A., Church, R., Murray, A., and Guignard, M. (2000). Forest management models and combinatorial algorithms: analysis of state of the art. *Annals of Operations Research*, 96(1–4):271–285.

Weintraub, A., Guitart, S., and Kohn, V. (1986). Strategic planning in forest industries. *European Journal of Operational Research*, 24(1):152–162.

Wolsey, L. A. (1998). *Integer Programming*. Wiley-Interscience, New York, NY.

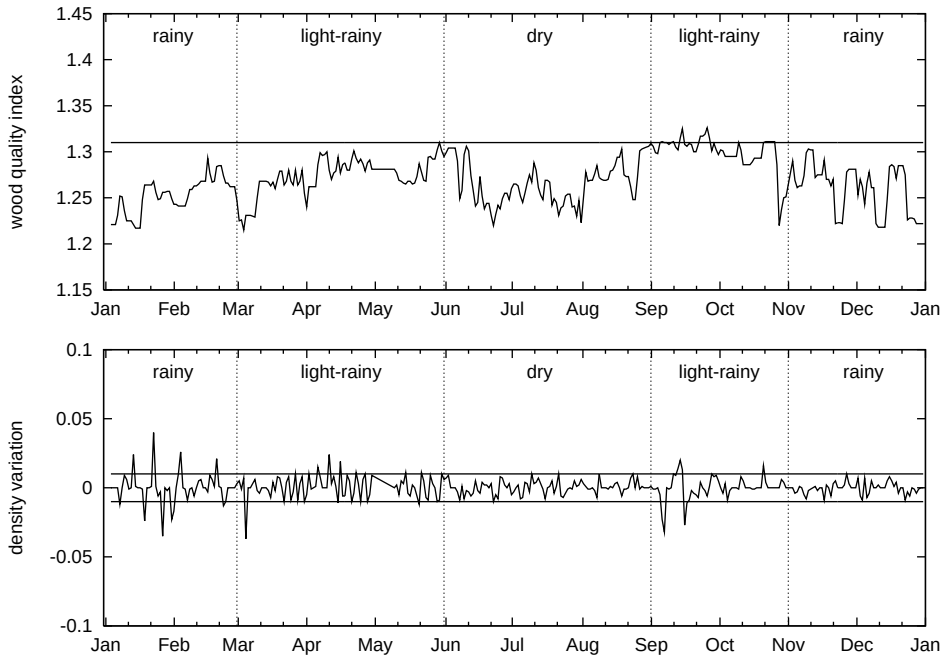


(a) Result without post-optimization phase

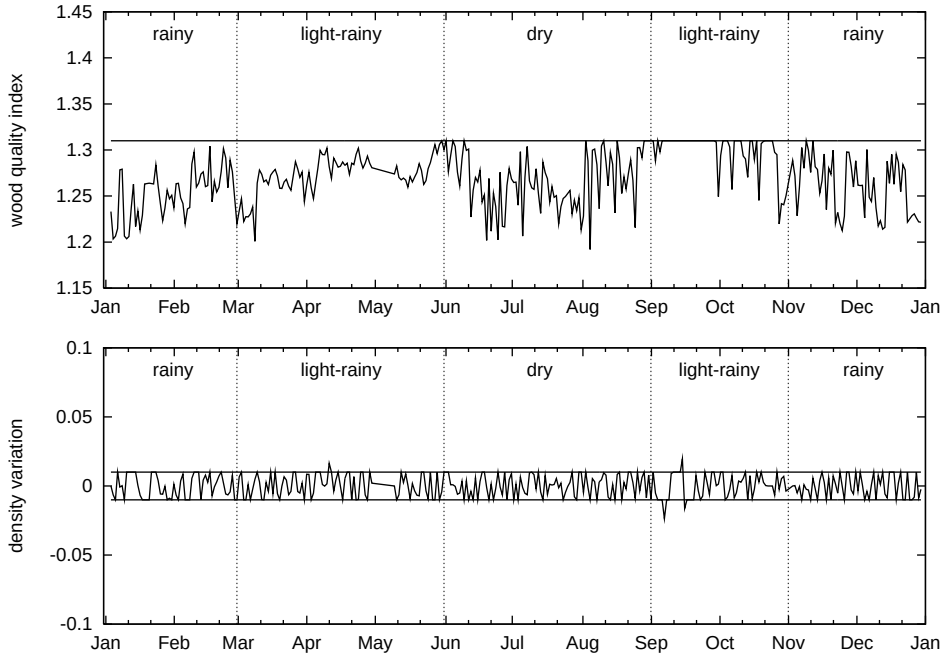


(b) Result with post-optimization phase

Figure 1: Behavior of the wood quality index and the logs' density variation for an average solution of instance inst-1.

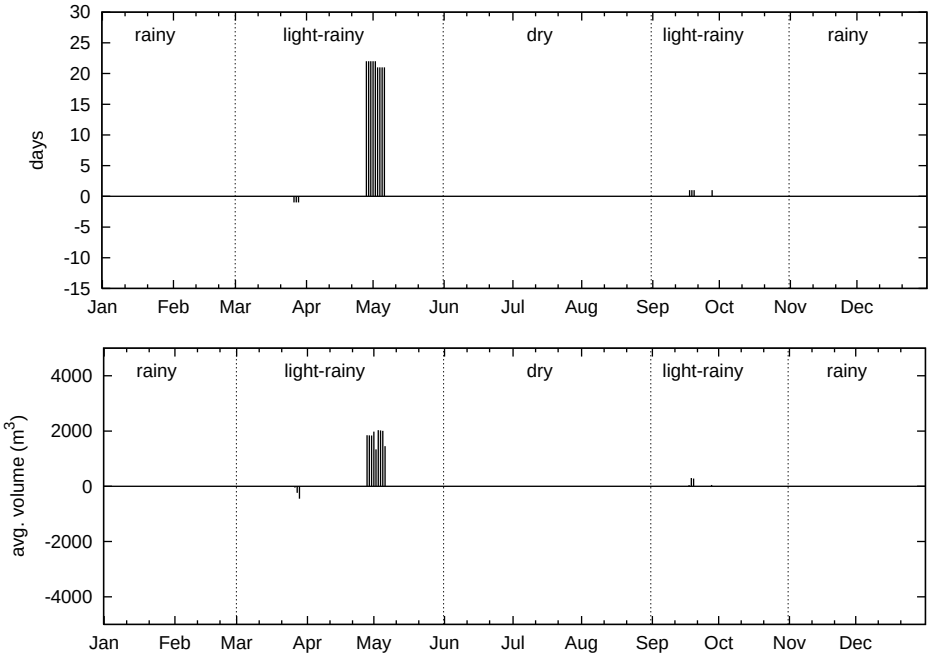


(a) Result without post-optimization phase

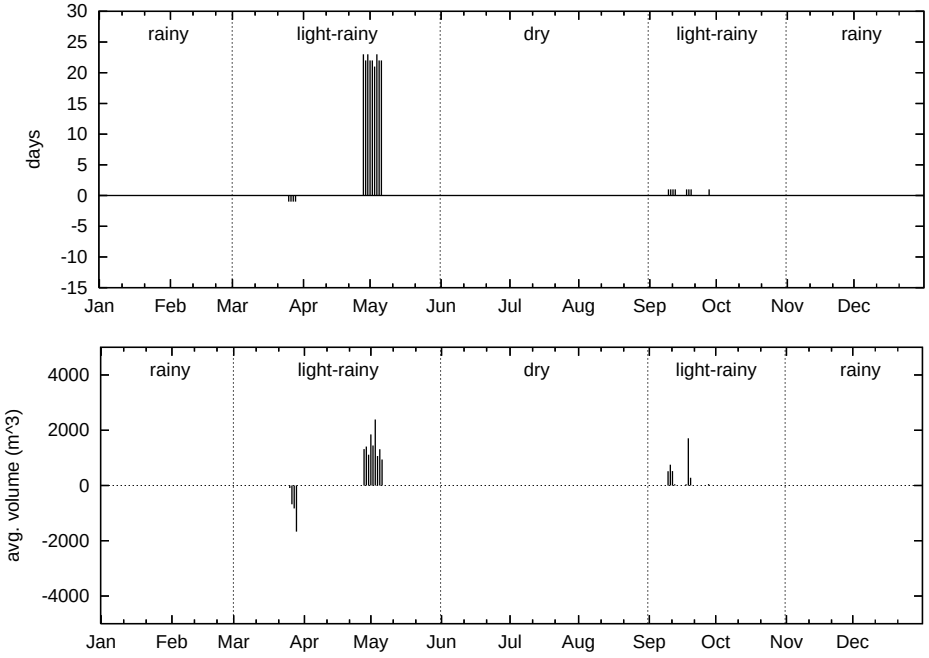


(b) Result with post-optimization phase

Figure 2: Behavior of the wood quality index and the logs' density variation for an average solution of instance inst-2.

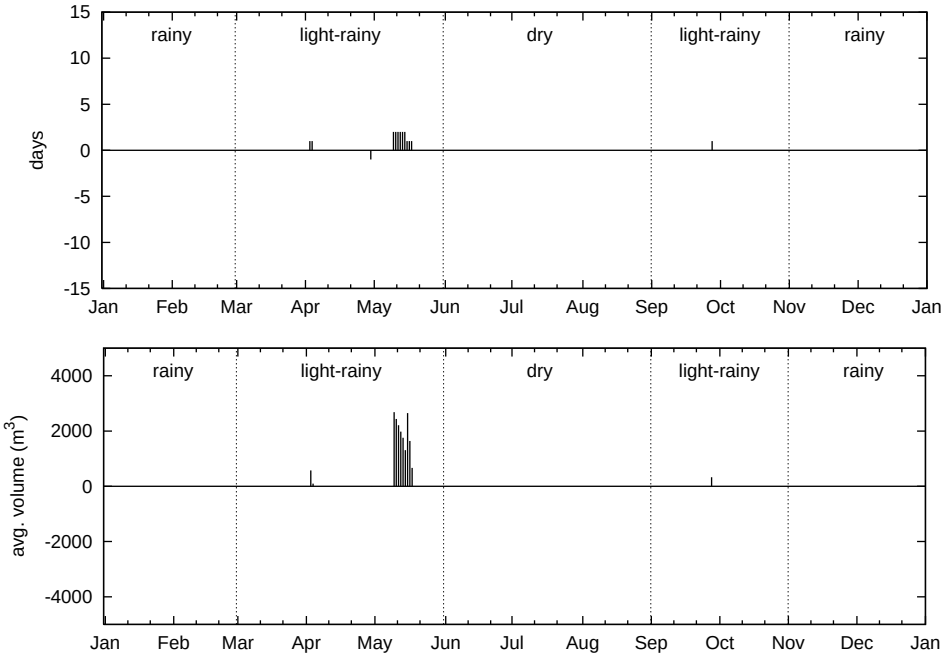


(a) Result without post-optimization phase

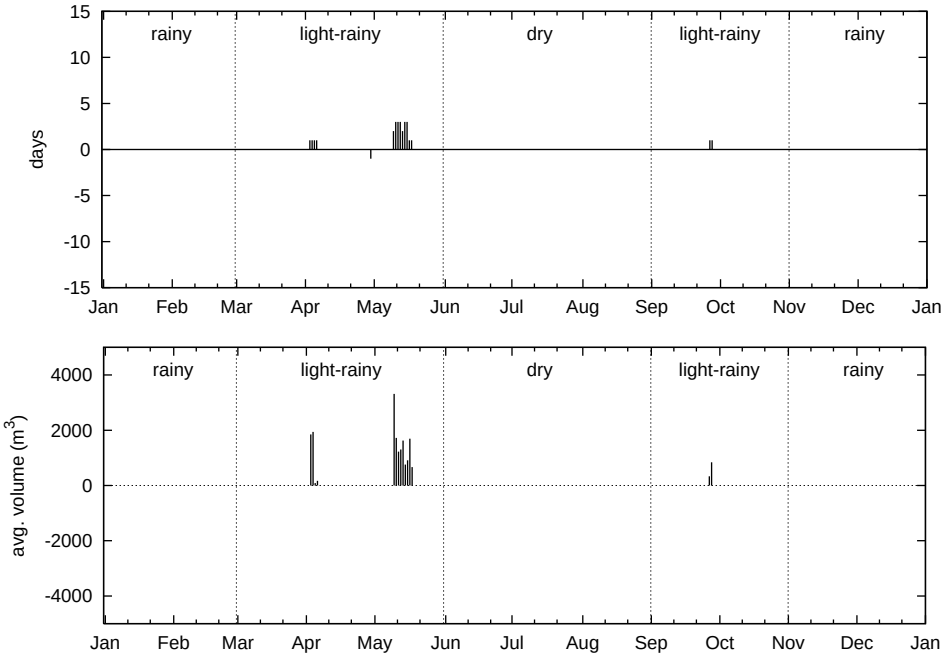


(b) Result with post-optimization phase

Figure 3: Behavior of the logs' waiting time for an average solution of instance inst-1



(a) Result without post-optimization phase



(b) Result with post-optimization phase

Figure 4: Behavior of the logs' waiting time for an average solution of instance inst-2