Available online at www.sciencedirect.com
ScienceDirect
European Journal of Operational Research 179 (2007) 788–805
EUROPEAN JOURNAL OF OPERATIONAL RESEARCH
www.elsevier.com/locate/ejor

A tabu search approach for solving a difficult forest harvesting machine location problem

Andres Diaz Legües [a], Jacques A. Ferland [b,1], Celso C. Ribeiro [c,2], Jorge R. Vera [d], Andres Weintraub [a,*,3]

[a] Department of Industrial Engineering, University of Chile, Casilla 2777, Av. Republica 701, Santiago, Chile
[b] Département Informatique et Recherche Opérationnelle, Université de Montréal, C.P. 6128, Succursale Centre-Ville, Montréal, Que., Canada H3C 3J7
[c] Department of Computer Science, Universidade Federal Fluminense, Rua Passo da Pátria 156, Niterói, RJ 24210-240, Brazil
[d] Department of Industrial and Systems Engineering, Catholic University of Chile, Santiago, Vicuna Mackenna 4860, Santiago, Chile

Received 8 April 2004; accepted 23 March 2005
Available online 20 December 2005

Abstract

This paper deals with two main problems in forest harvesting. The first is that of selecting the locations for the machinery to haul logs from the points where they are felled to the roadside. The second consists in designing the access road network connecting the existing road network with the points where machinery is installed. Their combination induces a very important and difficult problem to solve in forest harvesting. It can be formulated as a combination of two difficult optimization problems: a plant location problem and a fixed charge network flow problem. In this paper, we propose a solution approach based on tabu search. The proposed heuristic includes several enhancements of the basic tabu search framework. The main difficulty lies in evaluating neighboring solutions, which involves decisions related to location of machinery and to road network arcs. Hence, the neighborhood is more complex than in typical applications of metaheuristics. Minimum spanning tree algorithms and Steiner tree heuristics are used to deal with this problem. Numerical results indicate that the heuristic approach is very attractive and leads to better solutions than those provided by state-of-the-art integer programming codes in limited computation times, with solution times significantly smaller. The numerical results do not vary too much when typical parameters such as the tabu tenure are modified, except for the dimension of neighborhood.

* Corresponding author. Tel.: +56 2 6784046; fax: +56 2 6897895.
E-mail addresses: adlegues@rdc.cl (A.D. Legües), ferland@iro.umontreal.ca (J.A. Ferland), celso@inf.puc-rio.br (C.C. Ribeiro), jvera@ing.puc.cl (J.R. Vera), aweintra@dii.uchile.cl (A. Weintraub).
[1] This author's research was supported by NSERC grant (OGP 0008312) from Canada.
[2] Research of this author was supported by FAPERJ and CNPq, Brazil.
[3] This author's research was supported by FONDECYT #1040520.

0377-2217/$ - see front matter © 2005 Elsevier B.V. All rights reserved.
doi:10.1016/j.ejor.2005.03.071

Keywords: Tabu search; GRASP; Path relinking; Simulated annealing; Steiner tree; Forest harvesting; Machinery location; Network design

1. Introduction

The problem of how to use machinery when harvesting an area is an important one in forest industries. Machines are needed to haul logs from the point where they are felled to the roadside, where they are loaded into trucks for further transportation. The use of machinery depends on the slope of the ground. High slopes require the use of cable logging (towers), while flat grounds are handled using skidders (tractors). Access roads (usually quite short) are needed to connect the existing road network with the points where machinery is installed.

A computational system, PLANEX [4], has been used during the last decade by Chilean forest enterprises as a decision support system for locating the harvesting machineries and the access roads to build. The system is based on a graphic interactive interface linked to a geographical information system (GIS) storing information on topography, timber availability and geographical barriers like rivers and ravines. The GIS information is available for each 10 by 10 m cell. The decision process of PLANEX is based on a greedy heuristic. The system has been used very successfully, leading to important cost savings as well as better preservation of the environment [4,5].

Exact formulations have been proposed to improve the solution process. In [16], a model representing this problem was formulated. It is a difficult combinatorial problem and commercial mixed integer programming software is able to solve only small or medium size instances. A moderately better approach was proposed in [16] using Lagrangian relaxation to decompose the problem into its two basic components. The first sub-problem is a plant location problem: the machinery locations act as plants and timber cells represent customers. The second sub-problem is a fixed charge network flow problem. Both sub-problems are difficult to solve, in particular the fixed charge network flow problem. Strengthening the formulation of each sub-problem and using other enhancements made it possible to obtain better solutions, but still only for moderate size problems of up to 40 ha, 4000 cells, and 60 potential machine locations.

In this paper, we propose an alternate solution approach based on tabu search. The basic principles of this heuristic are the following. Evaluating neighbor solutions is difficult. Indeed, when we consider just the location problem, a typical neighbor solution is obtained either by modifying the status of one location (opening or closing it) or by exchanging the status of two locations (opening the one currently closed and closing the one currently opened). These modifications are easy to execute but, in our case, given any modification of locations, the corresponding access road network must be modified accordingly. This operation is very time consuming if done exactly. Thus, we approximate the cost of the new access road network by referring to a spanning tree of all relevant nodes, defined a priori. Hence, we obtain rapidly a suboptimal value of the neighbor solution. Once a set of locations is identified with this approach, we use a Steiner tree heuristic to determine the associated access road network. Other typical enhancements of the basic tabu search approach are also implemented and tested: (a) reduction of the neighborhood size [2], carried out by evaluating sequentially only a fraction of the neighboring solutions at each iteration, (b) variable tabu tenure [14], (c) intensification and diversification strategies, (d) GRASP [6,12] and other randomized selection processes to select the modifications generating the neighbor solutions, and (e) path relinking [8,11].

The numerical results indicate that reducing the neighborhood size has an important impact on CPU time, without significantly deteriorating the quality of the solutions. The impact of the other enhancements is less significant. The approach is tested using real life size problems with up to 500 ha, 50,000 cells, and 520 potential machine locations. As noted above, the approach proves to be very stable. When compared

with CPLEX 8.1, the results indicate that the tabu search approach leads to better solutions, with solution times of two orders of magnitude smaller. In addition, the tabu search heuristic generates solutions reasonably close to the optimum when compared with the bounds provided by those obtained with CPLEX.

Section 2 provides a description of the problem and the mathematical formulation of the model. The solution approach and the tabu search heuristic for the machinery location sub-problem are introduced in Sections 3 and 4, respectively. Section 5 describes different enhancements of the basic tabu search approach. Section 6 is devoted to the access road network design sub-problem. In Section 7, we present the numerical results. Concluding remarks are made in Section 8.

2. Problem formulation and mathematical model

Timber harvesting is typically carried out as follows. Trees are felled with saw equipment, and then must be hauled to trucks to be transported for further processing to destinations such as saw mills. Two different types of machinery are used in forest harvesting: towers and skidders. The towers are used in the steeper part of the forest. They are sort of cranes carried by heavy trucks and placed on the top of hills. Cables are drawn from the cranes and anchored at the bottom of the hill, in order to take the timber from the hillside to the top, where it is loaded on trucks and transported to the processing plant. Towers have engines providing traction power to the cables. Skidders are used in the more flat regions of the forest. They are nimble tractors moving on uneven ground as long as it is not too steep to carry the timber to storage areas along the roads. Skidders are less expensive to operate than towers. Since skidders are not fast, the road network has to be designed so as that they need not carry logs over distances longer than 300 m. Access roads are required to transport the timber harvested using both types of machinery.

The area to be harvested is partitioned into cells where timber is available, or where machinery can be installed, or representing road intersections. With each potential machinery location is associated a *reach zone* consisting of a set of cells which can be harvested from that location. Furthermore, for any given cell, the use of a tower or a skidder is predetermined by the configuration of the ground. Finally, some cells are associated with the exits of the forest. The original topographic curves are illustrated in Fig. 1, and they lead to a network configuration where nodes represent cells, road intersections, or exits, and arcs represent road segments as shown in Fig. 2.

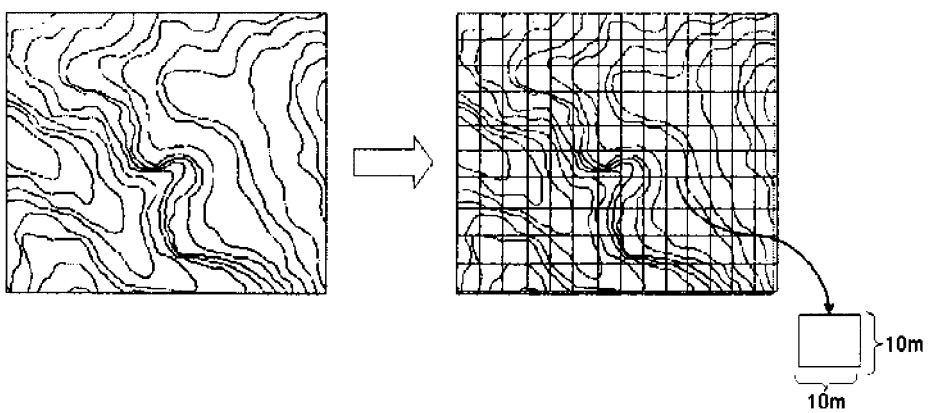

Fig. 1. Harvested area partitioned into cells.

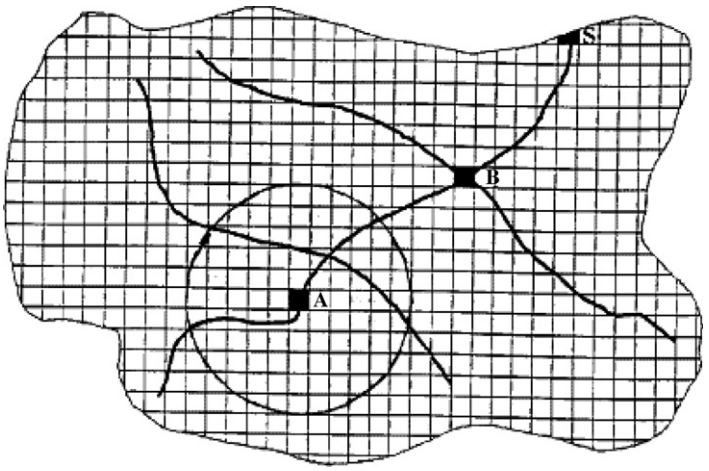

Fig. 2. Network of a harvested area, with A being a machinery location. The circle corresponds to its reach zone. B is an intersection of roads, while S is the exit of the forest.

In addition to locating optimally the different machineries over the area, a road network has to be designed to minimize the cost to exit the timber. Some road sections may already exist, but others have to be constructed.

The mathematical model refers to the following notation. We assume that a set of K different types of machinery is available (although only two types of machinery are used in our application). The area to be harvested is partitioned into a set $C = \{1, 2, \ldots, n\}$ of n cells. This set includes four subsets of different kinds of cells:

M: set of cells to be harvested,
N: set of cells representing intersections of (potential and existing) road portions of the network,
T^k: set of cells where machinery of type k can be located, with $k = 1, 2, \ldots, K$ (we assume there is no harvestable timber in these cells),
S: set of exit cells (note that $S \subset N$).

We also denote $T = \cup_{k=1,2,\ldots,K} T^k$. The following notation is used to specify the *reach zone* of each potential location of machinery in T:

$$P_{ij}^k = \begin{cases} 1, & \text{if cell } j \in M \text{ can be harvested from location } i \in T^k \text{ using machinery of type } k \\ 0, & \text{otherwise.} \end{cases}$$

Furthermore, let O_j be the timber volume available in cell $j \in M$.

A graph $G = (N, A)$ where $A \subset N \times N$ is used to described the road network. Some road sections (edges) already exist, while others are potential road sections to be constructed. For each road section $(q, r) \in A$, let K_{qr} be the capacity of the road section bounding the flow of the timber allowed on road section $(q, r) \in A$.

Finally, to specify the objective function, we use the following costs and revenues:

c_i^{1k}: installation (fixed) cost for machinery of type k in cell $i \in T^k$,
c_{ij}^{2k}: unit harvesting cost of using a machinery of type k in cell $i \in T^k$ to harvest cell j,
c_{qr}^3: construction cost of road section $(q, r) \in A$ (note that if road section $(q, r) \in A$ already exists, then $c_{qr}^3 = 0$),

c^4_{qr}: unit transportation cost of timber on road section $(q,r) \in A$,
δ: unit revenue for timber harvested.

The following variables are used in the model:

- Installation decision variables:

$$x_i^k = \begin{cases} 1, & \text{if machinery of type } k \text{ is located in cell } i \in T^k, \\ 0, & \text{otherwise.} \end{cases}$$

- Road construction decision variables:

$$z_{qr} = \begin{cases} 1, & \text{if road section } (q,r) \in A \text{ is used (i.e., } (q,r) \text{ has to be constructed if it does not already exist),} \\ 0, & \text{otherwise.} \end{cases}$$

- Variables associated with the timber volume harvested:
w^k_{ij}: timber volume harvested in cell $j \in M$ using machinery type k in cell $i \in T^k$,
y_i: timber volume harvested in cell $i \in T$,
f_{qr}: timber volume flowing through road section $(q,r) \in A$,
g_s: timber volume flowing through exit $s \in S$.

The mathematical model is formulated in [16] as follows:

$$\text{Max}\,\delta \quad \sum_{i\in T} y_i - \sum_{i\in T}\sum_k c_i^{1k} x_i^k - \sum_{i\in T}\sum_{j\in M}\sum_k c_{ij}^{2k} w_{ij}^k - \sum_{(q,r)\in A} c^3_{qr} z_{qr} - \sum_{(q,r)\in A} c^4_{qr} f_{qr} \tag{2.1}$$

subject to:

$$\sum_k x_i^k \leqslant 1 \quad i \in T, \tag{2.2}$$

$$w_{ij}^k \leqslant x_i^k O_j, \quad j \in M,\ i \in T^k,\ k \in K, \tag{2.3}$$

$$\sum_i \sum_k P_{ij}^k w_{ij}^k \leqslant O_j, \quad j \in M, \tag{2.4}$$

$$y_i = \sum_j \sum_k P_{ij}^k w_{ij}^k, \quad i \in T, \tag{2.5}$$

$$f_{qr} + f_{rq} \leqslant K_{qr} z_{qr}, \quad (q,r) \in A, \tag{2.6}$$

$$\sum_{(q,r)\in A} f_{qr} - \sum_{(r,t)\in A} f_{rt} = \begin{cases} -y_r, & r \in T, \\ 0, & r \in N - (T \cup S), \\ g_r, & r \in S, \end{cases} \tag{2.7}$$

$$x_i^k = 0 \text{ or } 1 \quad i \in T \tag{2.8}$$

$$z_{qr} = 0 \text{ or } 1, \quad (q,r) \in A, \tag{2.9}$$

$$y_i \geqslant 0, \quad i \in T, \tag{2.10}$$

$$w_{ij}^k \geqslant 0, \quad j \in M,\ i \in T^k,\ k \in K, \tag{2.11}$$

$$f_{qr} \geqslant 0, \quad (q,r) \in A. \tag{2.12}$$

The objective function (2.1) includes five terms. The first term is related to harvesting revenue. The next two terms are related to machinery location: the installation cost of machinery at the different locations and the

variable harvesting cost at these locations. The last two terms are related to the transportation of the harvested timber: the road sections construction cost and the variable flow cost of timber on these road sections. Constraints (2.2) indicate that at most one type of machinery can be located in each potential cell $i \in T$. In constraints (2.3), the timber volume harvested in cell $j \in M$ through location $i \in T$ is bounded by the total volume O_j available in cell j if machinery of type k is located at i, otherwise it is equal to zero. Constraints (2.4) limit the total volume of timber harvested in cell $j \in M$. The total volume harvested through each location $i \in T$ is specified in constraints (2.5). Constraints (2.6) indicate that if road section $(q,r) \in A$ is used, then the flow through it is limited by K_{qr}, otherwise it is equal to 0. Note that flow in arc (q,r) can go in both directions. Constraints (2.7) are the flow conservation constraints through the road network. In cells $r \in T^k$, representing machine locations, flows of timber y_r are produced and sent into the road network. In cells $r \in N$, representing road intersections, timber flow conservation constraints are imposed. Finally, in cells $r \in S$, representing exit nodes, g_r denotes the timber flowing through that exit. Finally, constraints (2.8) may indicate lower and upper bounds on timber flows through exit nodes. If multiple exit nodes exist, we introduce a dummy node Nd, and we link each exit node to Nd with an arc of cost zero.

3. Solution approaches

The difficulty to deal with the model introduced in the preceding section follows from the fact that it includes two difficult sub-problems to solve: a machinery location problem and a road network design problem. These sub-problems are interrelated through the flow conservation constraints (2.6). In [16], the authors use a Lagrangian relaxation approach to solve this problem. They combine a strengthening strategy using additional constraints with the Lagrangian relaxation technique to deal with the dual of the problem and to improve the solutions. Then they obtain a feasible solution for the (primal) problem using a Lagrangian heuristic. This approach can only solve small to medium size problems [16]. Computational tests showed this approach to be only moderately computationally superior to CPLEX 3.0.

In this paper, we propose a heuristic solution approach that can take advantage of the structure of the problem using a tabu search heuristic. A modified version of the typical location problem is developed to consider that whenever the machine location configuration is modified, then the corresponding access road network also changes. Since determining an optimal solution for the corresponding road network is very time consuming, we developed instead an approximate solution strategy. A simple approach based on spanning trees is used in most of the iterations. To improve the network specification at the end of the solution process, we use a Steiner tree heuristic.

4. Tabu search heuristic for the machinery location sub-problem

In this section we introduce a tabu search heuristic to solve the machinery location problem where only two types of machinery (towers and skidders) are used; i.e., $K = 2$ [3]. Furthermore, in real life applications, in general only one type of machinery can be used in any potential location according to the type of ground (hillside or flat ground). Hence the notation to introduce the method can be simplified accordingly, since for each potential location $i \in T$ we know a priory the type $k(i)$ of machinery that can be used. Consequently, we may eliminate the index k from any further consideration, and we denote a feasible selection of locations by $x = [x_1, x_2, \ldots, x_{|T|}]$. However, note that the current specific method introduced in this section can be easily generalized to deal with the more general model introduced in Section 2.

The main principles of tabu search were independently proposed by Glover [7] and Hansen [9]. This metaheuristic is based on a neighborhood or local search technique. The latter is an iterative procedure,

where at each iteration we move from a current solution x to a new solution x' in a neighborhood $N(x)$ of x. Hence the notion of neighborhood is a basic component of such a procedure.

In general, the neighbor solutions x' are generated by applying slight modifications $m \in M$ to solution x. The set of *modifications* or *moves* M is specific to each problem. Referring to [17], we denote by

$$x' = x \oplus m, \quad m \in M,$$

the neighbor solution x' generated by applying the modification $m \in M$ to solution x. Then, the neighborhood $N(x)$ of x is specified as follows:

$$N(x) = \{x' : x' = x \oplus m \text{ for some } m \in M\}.$$

For our location problems, we use two different types of modifications:

- 1-OPT: set of modifications where some location $i \in T$ is opened (a new machinery is located in $i \in T$) or closed (the machinery is removed from $i \in T$). Hence, if $x' = x \oplus m$ for some $m \in$ 1-OPT, then for some $i \in T$, either $x_i = 0$ and $x'_i = 1$, or $x_i = 1$ and $x'_i = 0$, while the rest of the variables remain unchanged.
- 2-OPT: set of modifications involving a pair of locations, in which one is opened and the other is closed (i.e., switching the status of two locations). Hence, if $x' = x \oplus m$ for some $m \in$ 2-OPT, then for some pair of locations $i_1, i_2 \in T$, $x_{i_1} = 1$, $x'_{i_1} = 0$, $x_{i_2} = 0$, and $x'_{i_2} = 1$, while rest of the variables remain unchanged.

Hence, the set of modification M is the union of the two sets 1-OPT and 2-OPT:

$$M = \text{1-OPT} \cup \text{2-OPT}.$$

This type of modifications has been used frequently in metaheuristic approaches in forest planning, mostly related to forest harvesting with spatial constraints which preclude harvesting adjacent land units in the same period. For example, in [1] a 1-OPT move involves exchanging the timing of a timber harvest for a single land unit, while a 2-OPT move involves swapping the harvesting times between two land units. A similar approach was used later in [2] and other publications on this problem.

At each iteration, the best solution x' in a subset $V(x) \subseteq N(x)$ is selected. The subset $V(x)$ is problem-dependent and will be specified shortly for our location problem. As long as x' is better than x (i.e., as long as the objective function increases), the behavior of the tabu search heuristic is the same as that of the standard ascent method. Otherwise, moving from x to x' induces no improvement or even a deterioration of the objective function, but allows to move out of a local minimum, and thus to search more extensively the feasible domain.

Since the value of the objective function is not necessarily strictly increasing, a safeguard against cycling is required. This is provided by including some kind of short term memory into the process to prevent returning to a solution already visited. Hence, a short term tabu list TL is used to keep track of some attributes or characteristics of the moves $m \in M$ that were recently used.

In our case, the tabu list TL includes the locations modified. Hence,

- If $x' = x \oplus m$ for some $m \in$ 1-OPT and $i \in T$ is the modified location, then i is inserted into TL.
- If $x' = x \oplus m$ for some $m \in$ 2-OPT and i_1, i_2 is the pair of modified locations, then the pair i_1, i_2 is inserted into TL.

Whenever a location is modified, it should keep the same status for several iterations. The tabu list TL can be represented as a circular list, in the sense that if it already includes the maximum number ρ of elements, then the oldest element of the list is removed before including a new element. Thus an element remains in TL for at most ρ iterations.

The size ρ of the tabu list is also called the tabu tenure. It is an important parameter regarding the efficiency of the heuristic, but its best value is not easy to be determined. Furthermore, it is often more efficient to use a variable size tabu list [14]. In this variant, the list size effectively used at each iteration varies to included the ρ' most recent elements, where ρ' is a random integer such that $\lfloor 0.8\rho \rfloor \leqslant \rho' \leqslant \rho$ ($\lfloor a \rfloor$ denotes the largest integer smaller than or equal to a).

Referring to the tabu list TL, a solution $x' = x \oplus m$ is declared tabu if

- $m \in$ 1-OPT and the location modified is in TL, or
- $m \in$ 2-OPT and either one of the modified locations i_1, i_2 is in TL.

The subset $V(x) \subseteq N(x)$ (referred to as the *effective neighborhood*) is generated by eliminating the tabu solutions in TL from $N(x)$.

The stopping criterion used in our implementation is specified in terms of a maximum number of iterations.

To improve the efficiency of the basic tabu search heuristic, additional features are introduced. They are described in the next section.

5. Additional tabu search features

To develop a more effective tabu search heuristic, we have to fully take advantage of the problem structure and appropriate data structures have to be used in the implementation. Furthermore, we should use improving strategies like intensification to search more deeply in the neighborhood of promising solutions, and diversification to search more extensively over the whole feasible domain.

5.1. Initial solution

To determine an initial solution, we first evaluate the profit of harvesting the reach zone of each potential location $i \in T$. This profit accounts for the revenue from the harvested timber, for the fixed installation cost of the machinery, and for the variable harvesting cost. Next, the locations are ranked in decreasing order of their profit.

We use the following procedure to specify an initial solution. At each iteration, we select the top ranking among the remaining locations. We eliminate the locations with reach zones intersecting that of the selected location. The procedure terminates when no location remains or if all cells have been reached. The reach zones of the selected locations in the initial solution do not intersect.

5.2. Reducing neighborhood size

Two different strategies are used to reduce the neighborhood size. The first one is related to the modifications in 2-OPT. We consider only the subset 2-OPT′ of modifications involving two locations that have intersecting reach zones. These modifications are the most relevant modifications in 2-OPT, since the other switching moves involving independent locations correspond to two different moves in 1-OPT.

The number of neighbors for both 1-OPT and 2-OPT moves is very large. To reduce this number, we use the strategy introduced in [2], where the set T of potential locations is partitioned into subsets or equivalence classes. Hence we generate p equivalence classes $[l] = \{i \in T : i \bmod p = l\}$, $l = 0, 1, \ldots, p-1$. Then, at iteration iter, we consider only locations i in the equivalent class [iter mod p] to generate moves in 1-OPT and pairs of locations i_1, i_2 such that i_1 or i_2 belonging to class [iter mod p] to generate those in 2-OPT′.

Note that, even if at each iteration we consider only a subset of moves involving locations in an equivalence class, each location is still used periodically.

5.3. Evaluating neighbor solutions

As mentioned earlier, evaluating a neighboring solution is complicated by the fact that if a machine location is opened or closed, or if there is an exchange of machine locations, then the corresponding road access network changes. To solve this complex problem exactly is very time consuming. Hence we develop a heuristic procedure to generate approximate solutions based on the computation of a spanning tree. For this purpose, a minimum spanning tree covering all potential locations is generated before the beginning of the tabu search procedure. This spanning tree is used to evaluate the neighbor solutions.

More specifically, for any neighbor solution, we consider the unique access road network including only arcs of the spanning tree linking the active machine locations to the exit (there is indeed a unique path in the spanning tree between any pair of nodes). This is very easy and fast to obtain. To improve even more the efficiency of this evaluation, we define an $M \times N$ matrix, where M is the number of potential locations and N the number of arcs in the spanning tree. An entry (i,j) of the matrix is equal to 1 if the timber harvested by a machine located in i goes through arc j of the spanning tree to reach the exit. Hence, by going through each column of the matrix, it is simple to determine which arcs have to be built in order to transport the harvested timber using the machines located at the current locations. Indeed, in any column j, if there is at least one entry equal to 1 in any row associated with a current location, then arc j needs to be built. To keep track of the arcs to be built for the current solution, we associate a vector of length N with the preceding matrix, where the value of the entry j associated with arc j is the sum of the entries in column j of the matrix associated with the current locations. Thus, the value of entry j in this vector corresponds to the number of current locations using this arc. This matrix is defined at the beginning of the process and the vector is easily updated each time a neighbor solution has to be evaluated. Indeed, in any neighbor solution, only one or two locations are modified. Whenever a current location i is closed in the neighbor solution, then the new vector is equal to the difference between the current vector and row i of the matrix. Similarly, whenever a new location i is opened, then the new vector is equal to the sum of the current vector to row i of the matrix.

This access road network is not optimal in general. Indeed, an optimal road network associated with the current locations usually includes other nodes as intermediate ones. A better road network solution can be determined, but at a higher computational cost. This is done for a small set of configurations that we consider as the best. For these configurations a Steiner tree heuristic is used to determine the associated access road network (see Section 6).

5.4. Intensification strategy

This strategy is often used in the implementation of tabu search heuristics. The basic idea is to periodically intensify the search in a seemingly promising region of the feasible domain. Hence, after executing the basic tabu search heuristic for a specified number max-iter of iterations, we use the current best feasible solution to reinitialize the same basic tabu search heuristic, where the size of the neighborhood is increased. This can be done by reducing the number of equivalence classes (thus enlarging each of them).

In our implementation we reduce this number to $\lfloor p/4 \rfloor + 1$. After a specified number of iterations, if the current best solution is better than the initial solution, then we move back to the basic tabu search heuristic with this initial solution. Otherwise, the current best solution is inserted into the set BESTSOL and we apply the diversification strategy.

5.5. Diversification strategy

Recall that this strategy is applied whenever the intensification strategy fails to improve the best current solution available at the end of the basic tabu search heuristic. Since the purpose of a diversification strategy is to search more extensively the whole feasible domain, one way of doing this is to open some locations that have been closed for a while. For this purpose, with each potential location $i \in T$ we associate the following information rc_i corresponding to the iteration number when location i has been closed for the last time (this value is equal to the current iteration number if the location is opened).

Let $\mathrm{REC} = \{\tau \in T : rc_\tau = \min_{i \in T}(rc_i)\}$ denote the set of locations that have been closed for the longest period of time. Then, to reinitialize the basic tabu search heuristic we use the current solution at the end of the intensification phase where we also open the additional locations in REC.

5.6. Additional procedures to select the current solution for the next iteration

At each iteration of the basic tabu search heuristic and of the intensification phase, we select the best solution in $V(x)$ to be the current solution for the next iteration. We now introduce two different groups of selection procedures using either a subset of the best neighbor solutions or all of them. Some of these procedures are deterministic and other probabilistic.

5.6.1. Using a subset of the best neighbor solutions

To implement these procedures, we first specify some value $\mathrm{PERCENT} \in (0, 100]$, and the set $B(x)$ including the $\lfloor \mathrm{PERCENT}\% \rfloor$ best solutions in $V(x)$.

(a) *GRASP* [6,11] *selection*: Select randomly a solution in $B(x)$ to be the current solution of the next iteration.
(b) *Proportional (roulette wheel) selection*: Associate with each solution in $B(x)$ a probability of being selected proportional to its value $f(x)$. Then select a solution randomly according to these probabilities.
(c) *Threshold selection* [2]: Rank the solutions in $B(x)$ in decreasing order of their objective function value $f(x)$. Specify also a THRESHOLD value in $(0, 1)$. At each iteration of the process, select the neighbor solution in the top of the list and a random number $r \in (0, 1)$. If $r \leqslant$ THRESHOLD, then the selected solution becomes the current solution for the next iteration. Otherwise, the selected solution is discarded and the process is repeated. In case all elements of $V(x)$ are discarded, then select the top ranking solution to become the current solution for the next iteration.

5.6.2. Using all solutions in $N(x)$

In order to implement these procedures, at each iteration of the basic tabu search heuristic or of the intensification phase, we first generate some permutation PERM of the potential locations belonging to the current equivalence class. Then, we consider each location according to its position in the permutation PERM in order to generate neighbor solutions using moves m in 1-OPT and 2-OPT. Each time a new solution x' is generated, we have to decide if it replaces the current solution x (and then move to the next iteration) or not, according to one of the following criteria:

(a) *Greedy selection*: If x' is not tabu and $f(x') > f(x)$, then x' replaces the current solution x.
(b) *Simulated Annealing type selection* [10]: If $\Delta f = f(x') - f(x) > 0$, then x' replaces the current solution x. If $\Delta f = f(x') - f(x) \leqslant 0$, then x' replaces the current solution x if $e^{\Delta f/\mathrm{TP}} > r$, where r is a random number in $(0, 1)$ and TP is a temperature factor decreasing with the number of iterations (note that the tabu list is not used in this selection procedure).

In case none of the solutions in $V(x)$ (resp. in $N(x)$) is selected by the *Greedy* procedure (resp. by the *Simulated Annealing* procedure), then select the best solution in $V(x)$ (resp. in $N(x)$).

5.7. Path relinking

Consider two feasible solutions x^1 and x^2. *Path relinking* [8,11] can be used to generate a sequence of other feasible solutions along a path linking the two solutions x^1 and x^2. First, we introduce a notion of distance $d(x^1, x^2)$ between x^1 and x^2. A commonly used distance for this operator is the Hamming distance, defined as the number of components of x^1 and x^2 that are different.

We also introduce the notion of a sub-neighborhood $N(x^1, x^2)$ of $N(x^1)$ "*closer*" to x^2:

$$N(x^1, x^2) = \{z \in N(x^1) : d(z, x^2) < d(x^1, x^2)\}.$$

Hence a solution $z \in N(x^1, x^2)$ has at least one more component in common with x^2.

For our problem,

- if $z = x^1 \oplus m \in N(x^1, x^2)$ and $m \in$ 1-OPT, then for some location $\tau \in T$, $x^1_\tau \neq x^2_\tau$ and $z_\tau = x^2_\tau$ while the rest of the components of z are equal to those of x^1;
- if $z = x^1 \oplus m \in N(x^1, x^2)$ and $m \in$ 2-OPT, then for some pair of locations $\mu, v \in T$, $x^1_\mu \neq x^2_\mu$, $x^1_v \neq x^2_v$, and $z_\mu = x^2_\mu$, $z_v = x^2_v$ while the rest of the components of z are equal to those of x^1.

The path relinking procedure can be summarized as follows. At the first iteration, let $ps^1 = x^1$ and $ps^2 = x^2$. Select an element $z \in N(ps^1, ps^2)$ according to the usual selection operator. If $f(z) > f(x^1)$ and $f(z) > f(x^2)$, then the path relinking procedure stops with z. Otherwise, ps^1 is replaced by ps^2, ps^2 by z, and the procedure moves to the next iteration. The procedure stops whenever $f(z) > f(x^1)$ and $f(z) > f(x^2)$, or $ps^1 = ps^2 = z$. Note that we move from x^1 and x^2 alternately.

The path relinking procedure is applied to pairs of solutions included in BESTSOL once the tabu search heuristic to solve the machinery location sub-problem is completed. For each solution in BESTSOL, the procedure is applied using the current best solution in BESTSOL as the second solution of the pair. If a better solution z is generated, then it is inserted into BESTSOL as the current best solution.

6. The road network design sub-problem

At the end of the tabu search method, a set BESTSOL of solutions is available for the machinery location sub-problem. Recall that during the resolution of this sub-problem, we use the road sections of the minimum spanning tree covering the potential locations and the exits of the forest to estimate the road network construction cost and the transportation cost. Now, to evaluate more exactly the cost of each solution in the set BESTSOL, we determine the best Steiner tree covering the opened locations using the procedure in [13] that can be summarized as follows.

The original problem can be viewed as a Steiner problem, where the terminal nodes are the cells with timber and the exits, while the Steiner points are the locations of machinery and road intersections. There are two types of arcs: those connecting timber cells with the cell where the corresponding harvesting machine is located and the roads. In the particular case of the algorithm presented in this section, the Steiner problem takes a different, simpler form. The terminal nodes are those where it has been decided a harvesting machine will be installed and the exit, while the Steiner points are road intersections and the non-chosen machine locations. The arcs are the roads.

An initial Steiner tree is built using the greedy construction algorithm of Takahashi and Matsuyama [15]. Its Steiner nodes characterize each Steiner tree. Neighbors of the current solution are defined by all sets of

Steiner nodes that can be obtained either by adding a new Steiner node (insertion moves) or by eliminating one of the current Steiner nodes (elimination moves). The tabu search procedure in [13] first investigates all insertion moves, since they can be evaluated faster than the elimination moves. To speedup the neighborhood search, the value of each possibly improving insertion move is first estimated by a very fast procedure running in linear time. Only a few insertion moves are exactly evaluated. In case the best insertion move improves the current solution, then it is selected and made effective by the local search procedure. Elimination moves are evaluated only in case no improving insertion move is found. Again, to speedup the search for improving elimination moves, the tabu search heuristic makes use of a quick procedure for the computation of lower bounds implementing a neighborhood reduction strategy. The move selected at each tabu search iteration is inserted into the tabu list. The classical aspiration criterion is used to override the forbidden status of a move. The heuristic stops after a certain number of iterations are performed without improvement in the best solution found.

7. Numerical results

Four different problems are used to analyze the different variants of the solution method introduced in this paper. The four problems are specified in Table 1. The two larger problems 3 and 4 are real life size problems. The first three were used in [16].

All the variants introduced in this paper were implemented in C. The numerical tests were completed on a 2000 MHz Pentium IV machine with 2048 Mbytes of RAM running under Linux. For each variant, preliminary numerical tests were used to set the values of the parameters. The different variants are compared numerically using these settings.

Each problem is solved using two different values δ of the unit revenue for the timber harvested: $\delta = 18$ and 50 US\$/m^3. Furthermore, in this study we analyze the impact of the tabu tenure and that of the neighborhood size. Three different tabu tenure values TL4, TL5, and TL6 were compared for problems 2, 3, and 4, where TLj = (total number of potential locations)/j. Also, to verify the impact of reducing the neighborhood size, the following numbers of equivalent classes are considered for the different problems:

- problem 1: 1,
- problem 2: 1, 4, 8, 12,
- problem 3: 1, 5, 10, 15, 20, 25, 30, 35, 40,
- problem 4: 1, 20, 40, 60, 80, 100, 120.

The other parameters of the different procedures are fixed according to preliminary tests or by referring to the literature. On the one hand, since our tabu search includes several successive major cycles, each of

Table 1
Problem data

Problem	1	2	3	4
Area (ha)	10	40	210	500
Number of cells	1000	4071	21,000	50,000
Number of potential tower locations	4	17	90	216
Number of potential skidder locations	6	41	150	398
Number of exit cells	1	1	5	11
Number of potential road sections	16	109	330	978
Number of existing road sections	0	45	36	102

which is composed by a basic search followed by an intensification phase and a diversification phase, the stopping criterion is specified in terms of a maximum overall number of iterations. This number increases with the problem size: 1000 iterations for problem 1, 2000 iterations for problem 2, 3000 iterations for problem 3, and 5000 iterations for problem 4.

On the other hand, within each major cycle, every time in which the basic tabu search or the intensification phase is completed, the stopping criterion is specified in terms of max-iter = 20 successive iterations without improvement in the objective function. The value of max-iter is small in order to increase the number of major cycles.

The rest of the parameters are related to the procedures to select the current solution for the next iteration. In our tests, we selected this solution among a small percentage (PERCENT = 5) of the best solutions in order to increase the chance of selecting better solutions. For the same reason, we use a large value THRESHOLD = 0.8 for this parameter. Note that similar values are used for this parameter in [2]. Furthermore, preliminary tests indicate that the solution time and the probability of selecting worst solutions increase with the value of PERCENT. Finally, for the *simulated annealing type selection*, the temperature is initialized with TP = 7 and is multiplied by 0.997 at each iteration.

The following notation is used to denote the different variants of the heuristic:

- Tabu: tabu search heuristic using *best solution selection*.
- Tabu G1: tabu search heuristic using *GRASP selection*.
- Tabu G2: tabu search heuristic using *Proportional selection*.
- Tabu G3: tabu search heuristic using *Threshold selection*.
- Tabu greedy: tabu search heuristic using *Greedy selection*.
- Tabu LV: tabu search heuristic using *best solution selection* and *variable* tabu tenure.
- Tabu SA: tabu search heuristic using *Simulated Annealing selection*.

For each variant, we also consider the corresponding variant where the *path relinking* procedure is applied to the set of solutions BESTSOL. This variant is denoted by appending +PR to the name of the corresponding original one (Tabu + PR, for instance).

In order to complete our numerical evaluation of the different options and variants, each problem is solved by each variant for each combination of values for δ, for TLj, and for the number of equivalence classes. Hence we have the following number of runs: 6 for problem 1, 304 for problem 2, 684 for problem 3, and 532 for problem 4.

Note that there are only six results for problem 1 because it is solved only with the variant Tabu. Furthermore, recall that variants Tabu SA and Tabu SA + PR do not use a tabu list.

Modifying the number of equivalence classes does not have a significant impact on the value of the objective function, but the solution times decrease significantly when the number of classes increases from one to larger values. On the other hand, the marginal gain in solution time decreases as the number of classes increases. Since this behavior is quite similar for all variants with different tabu tenures, we illustrate in Table 2 the typical behavior for the variant Tabu to solve problem 4 with $\delta = 50$ and the tabu tenure TL4. These results indicate that the relative difference between the largest and smallest values of the objective function is smaller than 0.05%, while the relative difference between the largest and the smallest values of the solution times is larger than 88.3%.

The impact of using different sizes (TL4, TL5, TL6) for the tabu tenure is not significant for any of the variants. The results in Table 3 when the variant Tabu is used to solve problem 4 with $\delta = 50$ and 60 equivalence classes illustrate the typical behavior observed for all variants. They indicate that the relative difference between the largest and the smallest values of the objective function and of the solution times are at most 0.093% and 10.9%, respectively.

Table 2
Impact of decreasing the neighborhood size for problem 4 using variant Tabu, with $\delta = 50$ and the tabu tenure TL4

Number of classes	Objective function (US$)	Time (seconds)
1	6,259,137.23	1478.22
20	6,260,059.00	335.11
40	6,259,999.31	288.81
60	6,259,090.23	228.61
80	6,260,358.12	210.70
100	6,260,390.13	178.82
120	6,257,406.24	172.67

Table 3
Impact of the different sizes (TL4, TL5, TL6) for the tabu tenure for problem 4 using variant Tabu, with $\delta = 50$ and 60 equivalence classes

Tabu tenure	$\delta = 18$		$\delta = 50$	
	Objective function (US$)	Time (seconds)	Objective function (US$)	Time (seconds)
TL4	1,515,081.53	223.24	6,259,090.23	228.61
TL5	1,516,496.63	238.43	6,259,932.31	240.89
TL6	1,516,001.93	244.11	6,258,389.58	256.55

When comparing the different variants, their efficiency is quite similar as far as solution quality and solution times are concerned. It seems difficult to identify some variants clearly dominating others. This is firstly illustrated in Table 4, where all variants are compared when solving problem 4 for $\delta = 18$ and $\delta = 50$, where the tabu tenure and the number of equivalence classes are TL4 and 60, respectively. As expected, on the one hand, the variants generating the best solution values (Tabu greedy + PR, Tabu LV + PR, and Tabu SA + PR) require twice more time than the fastest variant. On the other hand, the fastest variants (Tabu G2 and Tabu G1) obtain solutions that are roughly 0.2% worse than the best solutions.

Table 4
Comparing the efficiency of the variants when solving problem 4 for $\delta = 18$ and $\delta = 50$, where the tabu tenure and the number of equivalence classes are TL4 and 60, respectively

Variant	$\delta = 18$		$\delta = 50$	
	Objective function (US$)	Time (seconds)	Objective function (US$)	Time (seconds)
Tabu	1,515,081.53	223.24	6,259,090.23	228.61
Tabu PR	1,515,207.84	272.98	6,260,129.21	296.58
Tabu G1	1,513,035.69	159.19	6,256,638.49	154.38
Tabu G1 + PR	1,516,412.68	190.90	6,258,888.25	204.25
Tabu G2	1,514,180.13	157.17	6,256,888.84	156.59
Tabu G2 + PR	1,515,122.15	190.01	6,256,646.91	192.23
Tabu G3	1,516,574.40	238.11	6,260,032.68	231.59
Tabu G3 + PR	1,515,962.70	272.33	6,259,562.67	253.05
Tabu greedy	1,517,256.30	289.96	6,259,044.81	270.86
Tabu greedy + PR	1,518,095.28	374.10	6,260,428.37	314.14
Tabu LV	1,515,322.35	248.30	6,259,687.69	247.49
Tabu LV + PR	1,517,628.52	291.41	6,259,116.47	307.98
Tabu SA	1,517,694.42	309.65	6,260,342.61	335.44
Tabu SA + PR	1,516,613.50	361.06	6,259,373.80	375.15

The same overall behavior is observed in Tables 5–7, that report numerical results associated with the solution of problem 4 with $\delta = 18$ and $\delta = 50$ for tabu tenures TL4, TL5, and TL6, respectively. In these tables, for each tabu tenure value and each variant, we give the average value (mean) and the standard deviation (stand. dev.) of the objective function values and of the CPU times using the results for the six different numbers of equivalence classes (20, 40, 60, 80, 100, and 120). Note that since there is no tabu list in variants Tabu SA and Tabu SA + PR, then we have only one set of results for these variants in Table 5.

In Tables 8 and 9, we evaluate the effectiveness of the variant Tabu with respect to the solution of the integer programming model by a software package such as CPLEX 8.1. For each problem we use TL4 as the tabu tenure. The number of equivalence classes for problems 1, 2, 3, and 4 is set at 1, 8, 20, and 60, respectively. For each problem, we indicate the value of the objective function and the computation time

Table 5
Average behavior of the tabu search variants for solving problem 4 using TL4

	TL4 and $\delta = 18$				TL4 and $\delta = 50$			
	Objective function		Time		Objective function		Time	
	Mean	Std. dev.	Mean	Std. dev.	Mean	Std. dev.	Mean	Std. dev.
Tabu	1,516,143.17	1016.42	223.53	73.13	6,259,550.51	1151.26	235.79	64.16
Tabu PR	1,516,296.74	1001.34	298.62	91.47	6,259,835.31	1132.27	287.48	68.17
Tabu G1	1,514,276.08	2372.79	160.92	79.40	6,253,113.34	7171.47	169.07	83.87
Tabu G1 + PR	1,514,374.12	2312.36	189.98	85.11	6,258,181.44	1251.12	198.14	104.99
Tabu G2	1,514,661.10	2011.79	168.93	74.64	6,255,524.91	3256.77	161.44	88.70
Tabu G2 + PR	1,515,991.57	1151.66	197.88	100.66	6,254,168.29	5901.34	210.16	121.19
Tabu G3	1,515,642.16	1093.15	218.08	63.81	6,259,085.58	1247.64	221.17	61.93
Tabu G3 + PR	1,516,026.65	652.47	290.74	74.17	6,259,554.49	872.82	277.09	74.60
Tabu greedy	1,516,389.04	1131.11	294.78	76.10	6,258,908.98	739.01	272.57	63.72
Tabu greedy + PR	1,516,624.33	1272.80	354.92	53.10	6,259,726.49	620.34	337.25	76.07
Tabu LV	1,516,178.44	866.85	240.28	64.08	6,259,273.48	1136.34	243.44	72.63
Tabu LV + PR	1,516,343.64	1042.74	287.96	54.41	6,259,605.09	902.34	292.70	53.29
Tabu SA	1,516,313.13	719.72	304.83	109.42	6,259,227.46	918.74	317.06	125.62
Tabu SA + PR	1,516,351.34	695.19	361.00	90.99	6,259,817.19	708.45	397.08	105.71

Table 6
Average behavior of the tabu search variants for solving problem 4 using TL5

	TL5 and $\delta = 18$				TL5 and $\delta = 50$			
	Objective function		Time		Objective function		Time	
	Mean	Std. dev.	Mean	Std. dev.	Mean	Std. dev.	Mean	Std. dev.
Tabu	1,515,875.31	996.74	239.95	75.60	6,259,238.23	1066.26	240.60	76.67
Tabu PR	1,515,921.93	921.70	303.63	76.56	6,259,590.49	721.31	306.07	83.86
Tabu G1	1,511,476.76	7177.26	170.77	77.84	6,254,525.11	3584.61	165.38	92.49
Tabu G1 + PR	1,516,067.45	2529.60	203.43	101.18	6,257,180.00	3815.80	202.28	125.19
Tabu G2	1,514,707.90	1855.83	165.10	80.26	6,255,027.18	6359.03	158.81	86.34
Tabu G2 + PR	1,514,628.89	1624.67	193.21	96.88	6,257,692.09	2337.12	210.00	101.33
Tabu G3	1,515,425.16	1239.68	231.99	78.48	6,259,424.70	970.02	232.90	75.00
Tabu G3 + PR	1,516,771.88	1190.41	291.43	73.05	6,259,504.36	917.80	294.50	78.77
Tabu greedy	1,515,771.19	1185.17	292.65	87.14	6,259,738.06	888.72	283.14	74.17
Tabu greedy + PR	1,517,150.20	1141.07	354.26	71.76	6,259,678.63	918.44	352.82	78.38
Tabu LV	1,515,587.11	1226.39	261.92	64.38	6,259,589.52	962.16	253.86	86.93
Tabu LV + PR	1,516,035.77	842.06	298.66	52.64	6,259,491.93	719.19	324.72	77.91

Table 7
Average behavior of Tabu variants for solving problem 4 using TL6

	TL6 and $\delta = 18$				TL6 and $\delta = 50$			
	Objective function		Time		Objective function		Time	
	Mean	Std. dev.	Mean	Std. dev.	Mean	Std. dev.	Mean	Std. dev.
Tabu	1,515,990.71	599.44	238.00	61.21	6,259,292.94	918.22	245.06	87.71
Tabu PR	1,515,890.64	473.52	273.39	51.57	6,259,869.73	504.74	295.44	76.84
Tabu G1	1,513,362.68	2728.46	169.39	82.11	6,255,519.96	3459.85	167.85	92.71
Tabu G1 + PR	1,515,535.90	1344.69	194.60	98.69	6,257,378.01	2,944.09	209.86	100.66
Tabu G2	1,514,898.16	1378.30	162.02	80.69	6,256,087.58	1891.97	172.52	96.02
Tabu G2 + PR	1,515,293.01	2350.79	206.75	95.26	6,255,881.80	4545.92	209.37	124.34
Tabu G3	1,515,643.61	523.03	243.08	70.06	6,259,044.93	845.16	248.64	88.64
Tabu G3 + PR	1,516,643.47	839.96	278.99	84.84	6,259,384.67	581.91	299.22	65.28
Tabu greedy	1,516,087.65	951.79	273.25	79.97	6,259,232.82	1070.48	284.10	77.37
Tabu greedy + PR	1,516,010.76	713.65	334.74	70.30	6,259,486.90	835.49	346.88	93.64
Tabu LV	1,516,113.34	645.47	250.92	70.55	6,259,584.62	1404.83	246.16	80.14
Tabu LV + PR	1,515,713.15	969.20	309.29	42.94	6,259,605.86	971.68	300.50	60.59

Table 8
Results obtained with CPLEX 8.1 (processing time limited to 600 minutes) and with variant Tabu of the heuristic, with the unit revenue set at $\delta = 18$

		CPLEX 8.1		Tabu
$\delta = 18$ US$/m^3				
Problem 1	Objective function (US$)	10,704.40		10,704.40
	Upper bound US$		10,704.40	
	Gap		0%	
	Time (minutes)	0.020		0.001
Problem 2	Objective function (US$)	90,881.38		91,938.66
	Upper bound (US$)		100,725.58	
	Gap		10.83 %	
	Time (minutes)	600.00		0.12
Problem 3	Objective function (US$)	497,522.31		501,321.38
	Upper bound (US$)		559,349.56	
	Gap		12.43 %	
	Time (minutes)	600.00		1.09
Problem 4	Objective function (US$)	1,488,866.70		1,515,081.53
	Upper bound (US$)		1,680,473.60	
	Gap		12.87 %	
	Time (minutes)	600.00		3.72

for both approaches. The upper bound obtained with CPLEX 8.1 and the relative difference between the solution values obtained with each approach are also given. The solution time with CPLEX 8.1 is limited to 600 minutes for each problem. The solution of problem 4 with $\delta = 50$ using CPLEX 8.1 stopped with an error message indicating that all the memory has been used after 520.07 minutes. Results in these tables indicate that the variant Tabu can improve solution quality for all problems (except for problem 1, where both approaches found an optimal solution) with respect to CPLEX 8.1 within 10 hour of computations, with solution times significantly smaller.

Table 9
Results obtained with CPLEX 8.1 (processing time limited to 600 minutes) and with variant Tabu of the heuristic, with the unit revenue set at $\delta = 50$

		CPLEX 8.1		Tabu
$\delta = 50$ US$/$m^3$				
Problem 1	Objective function (US$)	85,992.56		85,992.56
	Upper bound (US$)		85,992.56	
	Gap		0%	
	Time (minutes)	0.020		0.001
Problem 2	Objective function (US$)	414,502.59		416,858.19
	Upper bound (US$)		427,966.33	
	Gap		3.25%	
	Time (minutes)	600.00		0.12
Problem 3	Objective function (US$)	2,040,319.79		2,041,777.38
	Upper bound (US$)		2,102,811.93	
	Gap		3.06%	
	Time (minutes)	600.00		1.04
Problem 4	Objective function (US$)	6,222,050.04		6,259,090.23
	Upper bound (US$)		6,420,475.28	
	Gap		3.19%	
	Time (minutes)	520.07		3.81

8. Conclusions

In this paper, we have proposed a tabu search heuristic to solve a relevant problem in forest management, namely the machine location and road access network design problem. Approaches to solve this problem using exact formulations have been able to solve only moderate size problems. Particular consideration is given to the evaluation of each neighbor solution, because an optimal access road network has to be determined for any given machine location configuration. In most of the iterations, to obtain rapid evaluations, approximate solutions were used, with the access road network being determined based on a spanning tree covering all potential machine locations. For the best configurations, a Steiner tree heuristic was used to determine the associated access road network.

Computational experiments were carried out to calibrate the parameters of the tabu search procedure and to test different variants. These tests showed that the only additional feature that significantly influenced solution quality and run times was the reduction of the neighborhood size. Other additional features tested did not have a significant impact on solution quality or run time. The proposed approach was tested on eight problems similar to real forest problems. Numerical results obtained with CPLEX 8.1 showed that the tabu search heuristic seems very attractive. Indeed, the heuristic performed considerably better than CPLEX 8.1, obtaining better solutions in solution times significantly smaller than the ten hours of computation times allowed to CPLEX 8.1. The gaps in solution value, favoring the heuristic with respect to CPLEX, ranged between 3% and 4% for $\delta = 50$ and between 10% and 13% for $\delta = 18$. The computation times observed for the tabu search heuristic increased almost linearly with problem complexity, measured either in terms of the number of cells or the number of potential roads to be built.

These promising results using a metaheuristic have practical applications, as this approach can be implemented into existing machine location systems used by forest companies, improving the quality of solutions over the simpler heuristics presently in use and keeping computer times very low. In the near future, it is planned to introduce this new solution approach and the basic tabu search variant with reduced neighborhoods into the machine location planning system PLANEX [5].

References

[1] P.K. Bettinger, K. Boston, J. Sessions, Intensifying a heuristic forest harvest scheduling search procedure with 2-opt decision choices, Canadian Journal Forestry Research 29 (1999) 1784–1792.
[2] F. Caro, M. Constantino, I. Martins, A. Weintraub, A 2-opt tabu search procedure for the multi-period forest harvesting problem with adjacency, green-up, old growth and even flow constraints, Forest Science 49 (2003) 1–14.
[3] A. Diaz Legües, Desarrollo de un algorithmo heuristico tabu para la localizacion de maquinaria de cosecha forestal, Memoria de Ingeniería Industrial, Universidad de Chile, Santiago, Chile, 2003.
[4] R. Epstein, R. Morales, R. Seron, A. Weintraub, OR models in the Chilean forest industry, Interfaces 29 (1999) 7–26.
[5] R. Epstein, A. Weintraub, P. Sapunar, E. Nieto, B. Sessions, J. Sessions, F. Bustamante, H. Musante, A combinatorial heuristic approach for solving real size machinery location and road design problems in forestry planning, Operations Research, in press.
[6] T. Feo, M.G.C. Resende, Greedy randomized adaptive search procedures, Journal of Global Optimization 2 (1995) 1–27.
[7] F. Glover, Future paths for integer programming and links to artificial intelligence, Computers & Operations Research 13 (1986) 533–549.
[8] F. Glover, M. Laguna, Tabu Search, Kluwer, Boston, 1997.
[9] P. Hansen, The Steepest–Ascent–Mildest–Descent heuristic for combinatorial programming, Congress on Numerical Methods in Combinatorial Optimization, Capri, Italy, 1986.
[10] D.E. Jeffcoat, R.L. Balfin, Simulated annealing for resource-constrained scheduling, European Journal of Operational Research 70 (1993) 43–51.
[11] M.G.C. Resende, C.C. Ribeiro, GRASP with path-relinking: Recent advances and applications, in: T. Ibaraki, K. Nonobe, M. Yagiura (Eds.), Metaheuristics: Progress as Real Problem Solvers, Kluwer, in press.
[12] M.G.C. Resende, C.C. Ribeiro, Greedy randomized adaptive search procedures, in: F. Glover, G. Kochenberger (Eds.), Handbook of Metaheuristics, Kluwer, 2002, pp. 219–249.
[13] C.C. Ribeiro, M.C. Souza, Tabu search for the Steiner problem in graphs, Networks 36 (2000) 138–146.
[14] E. Taillard, Robust tabu search for the quadratic assignment problem, Parallel Computing 17 (1991) 443–455.
[15] H. Takahashi, A. Matsuyama, An approximate solution for the Steiner problem in graphs, Mathematica Japonica 24 (1980) 573–577.
[16] J. Vera, A. Weintraub, M. Koenig, G. Bravo, M. Guignard, F. Barahona, A Lagrangian relaxation approach for a machinery location problem in forest harvesting, Pesquisa Operacional 23 (2003) 111–128.
[17] D. de Werra, A. Hertz, Tabu search techniques: A tutorial and an application to neural networks, OR Spektrum 11 (1989) 131–141.