Chapter 4

LOG-TRUCK SCHEDULING WITH A TABU SEARCH STRATEGY

Manfred Gronalt and Patrick Hirsch
Institute of Production Economics and Logistics, BOKU – University of Natural Resources and Applied Life Sciences, Vienna, Feistmantelstrasse 4, A-1180 Wien

Abstract: The Austrian forest sector has experienced extensive development in recent years. In 2003, approximately 27.9 million cubic meters of logs were processed in Austria. In order to enable a stable supply, an efficient and economical operation for round timber transport is necessary. In this paper, we present a Tabu Search based solution method for log-truck scheduling. A fleet of *m* log-trucks that are situated at the respective homes of the truck drivers must fulfill *n* transports of round timber between various wood storage locations and industrial sites. All of the transports are carried out as full truckloads. Since the full truck movements are known, our objective is to minimize the overall duration of empty truck movements. In addition to the standard VRP, we have to take into consideration weight constraints on the road network, multi-depots, and time windows at the industrial sites and homes of the truck drivers. We applied the Unified Tabu Search method and modified it by an oscillating change of the neighborhood size in some selected iteration steps. Our heuristics are verified with extensive numerical studies. The Tabu Search based heuristics are able to solve real-life problems within a reasonable timeframe by providing good solution quality.

Keywords: Log-truck scheduling; Timber Transport Vehicle Routing Problem; Tabu Search

1. INTRODUCTION AND PROBLEM DESCRIPTION

The Austrian forest sector has experienced extensive development in recent years. With respect to the Austrian economy, forest based industry is

in second place in terms of exporting goods and services. In 2003, Austria had 1,400 sawmills, 30 pulp mills, ground wood pulp mills, and paper mills, and 39 chipboard factories that processed approximately 27.9 million cubic meters of logs (Schwarzbauer, 2005). In order to enable a stable supply, an efficient and economical operation for round timber transport is necessary. A number of natural restrictions, such as regional topology, storms, and heavy snow may hinder a steady supply for industrial recipients. In this regard, the availability of the forest road network is of great importance. A number of research efforts mainly focus on information supply and the provision of GIS based applications for supporting truck drivers in finding storage locations and to further support wood transfer. Moreover, there exist numerous proposals for the efficient use of wood transportation systems. In order to sustain the competitiveness of the Austrian forest based industry improvements in transportation logistics are often considered an essential starting point. Our work focuses on the log-truck scheduling problem, which typically has many sources and few recipients. At the beginning of a planning period, the transportation orders are given. Here, we are considering full truckloads when it is that the truck moves from the wood storage location to a particular industrial site. When starting at the home location, and after unloading at the mill, we have to decide where the trucks should collect a new load in order to minimize the overall empty truck movements.

The problem we are discussing here is relevant for large forestry companies that serve a number of different mills; it describes the challenge of reducing the mileage of log-trucks. In our background forestry application, 10 trucks and approximately 30 trips are scheduled daily. In Austria, forest owners usually need to organize the log transport by employing forwarding companies. These forwarders usually serve several forest owners per day and aim to minimize their transportation costs. With this respect, the presented scheduling problem is highly relevant for log transport companies.

The emerging vehicle routing problem is denoted as a Timber Transport Vehicle Routing Problem (TTVRP) (see also Karanta et al., 2000 and Weintraub et al., 1996). It can be characterized as follows: a heterogeneous fleet of m log-trucks that are situated at the respective homes of truck drivers must fulfill n transports of logs between various wood storage locations and industrial sites, such as pulp mills and sawmills, during a specified timeframe. All of the transports are carried out as full truckloads; the vehicle is loaded at the wood storage location and unloaded at the industrial site. Each route commences at the home of the truck driver who leaves with an empty truck for loading round timber. Subsequently, he drives to the designated industrial site and completes the transport. The truck driver can

now finish his tour and return back home or start a new delivery. Due to the transportation orders, each wood storage location and industrial site can be visited more than once during the planning horizon.

Log transport has a few specific constraints to consider such as the fact that some parts of the forest road networks are unsuitable for larger trucks, as their weight occasionally damages the road. Therefore, some wood storage locations can only be reached by trucks with a certain capacity. We denote this as the route weight limits. Due to industry operating hours, time windows for unloading wood must be considered. Time windows also occur at the truck starting points since truck drivers are only on duty at certain times. Additionally, we have to observe tour length constraints and capacity constraints. According to the given transportation orders, the objective is to minimize empty truck movements.

In Figure 1, we present a small example to illustrate the planning problem. Two log-trucks have to perform eight transportation orders (*1*, …, *8*). The log-trucks are situated at the home-locations *A* and *B*, respectively. Wood is provided at six different wood storage locations (*P1*, …, *P6*) and must be transported to three industrial sites (*I1*, …, *I3*). The number of rectangles and triangles provides the number of visits at the respective location. *I1* receives three loads: two from *P1* and one from *P2*. Figure 1a) shows the required transports and demonstrates the problem of linking these transports in a cost-efficient manner, taking into account the above-mentioned constraints. Figure 1b) shows the cost-optimal solution for this problem. *A1* is the first trip taken from the log-truck situated at *A*, *A2* is the second one, etc.; the same is true for *B1* to *B11*. Altogether, we have scheduled 8 transports and 10 empty truck movements.

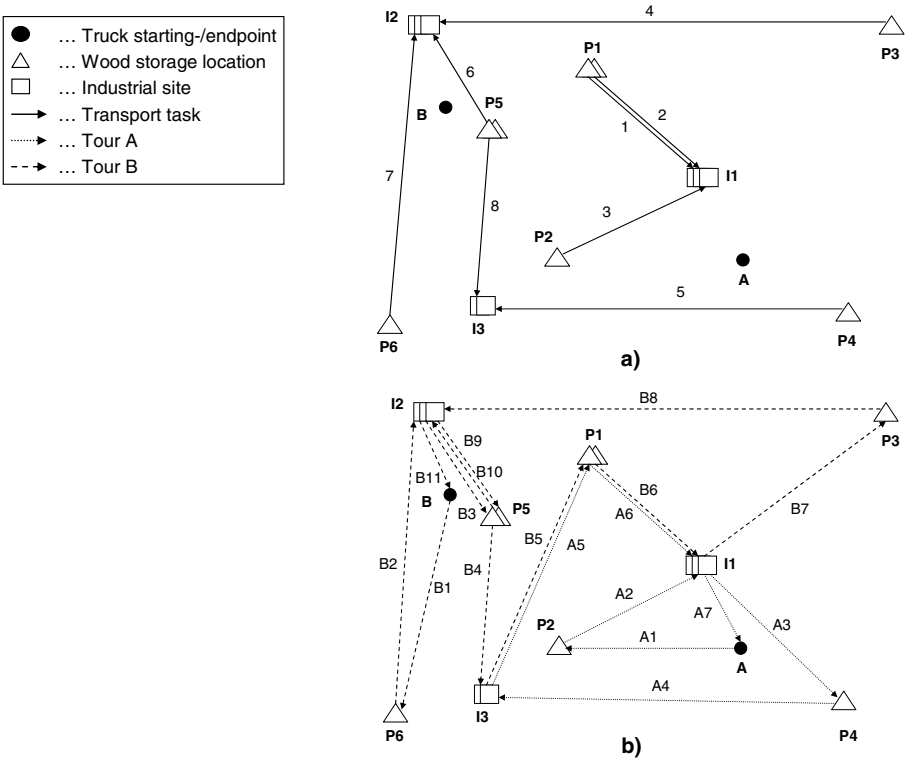

Figure 4-1. Conceptual formulation and solution for a TTVRP

The TTVRP is a special application of the full truckload vehicle routing problem (Gronalt et al., 2003). Murphy (2003) presents an approach that attempts to reduce the number of log-trucks that are used to perform transports of round timber. He developed a MIP model that minimizes the total transport costs. His approach does not take into account, however, time windows at industrial sites, availability times of the drivers, or route weight limits. He uses standard solver software to solve his problems but he only provides the best found solution after a certain computing time and not the global optimal solution. The approach of Palmgren et al. (2003) unites tactical and operational planning in wood transport. They provide a model formulation for the Log Truck Scheduling Problem (LTSP) and present a column generation based solution approach.

According to the established notation on VRPs, the TTVRP is related to the Multi Depot Vehicle Routing Problem with Pickup and Delivery, and Time Windows (MDVRPPDTW); supplementarily, one has to deal with specific route weight limits and full truckloads. An overview of the Vehicle Routing Problems can be found for example in Toth and Vigo (2002). The transport activities of the TTVRP have a similar structure to the Stacker

Crane Problem (SCP) (see Righini et al., 1999). Coja-Oghlan et al. (2004) provide an example of a SCP, which describes the scheduling of a delivery truck. Glover and Laguna (1997) provide a general introduction to the Tabu Search metaheuristic. For solving the TTVRP the Unified Tabu Search (Cordeau et al., 2001) is adapted and modified.

This present paper is organized in the following way: In Section 2, we present a model formulation of the TTVRP. The heuristic solution approach is outlined in the third section. We have developed three variants of the Tabu Search in order to obtain solutions for the TTVRP. Section 4 describes our numerical studies and the generation of test data. The results of the numerical experiments are provided in Section 5. We use different parameter sets for our heuristics and compare the three variants of the Tabu Search with each other and the best found feasible solution obtained with solver software. Finally, our conclusions are drawn in Section 6, in which an outlook on our future research is also provided.

2. MODEL FORMULATION

The transportation orders are predefined and can therefore be considered as tasks that must be fulfilled in order to obtain a feasible solution. A feasible solution must include all of the tasks that are represented by arcs. Figure 2 demonstrates the same problem as Figure 1, which is transformed into a special case of the SCP. We have two kinds of tasks, so-called artificial tasks (*A'*, *B'*) and transport tasks (*1*, …, *8*). The artificial tasks are introduced in order to connect the starting point and endpoint of a cycle. The direct connection between two vertices is always the shortest one. It is impossible to transport directly from one wood storage location to another or from one industrial site to another. This is because we have to deal with full truckloads. In Figure 2a) the tasks and vertices are displayed. Figure 2b) shows the corresponding optimal solution, using the same notation as in Figure 1b).

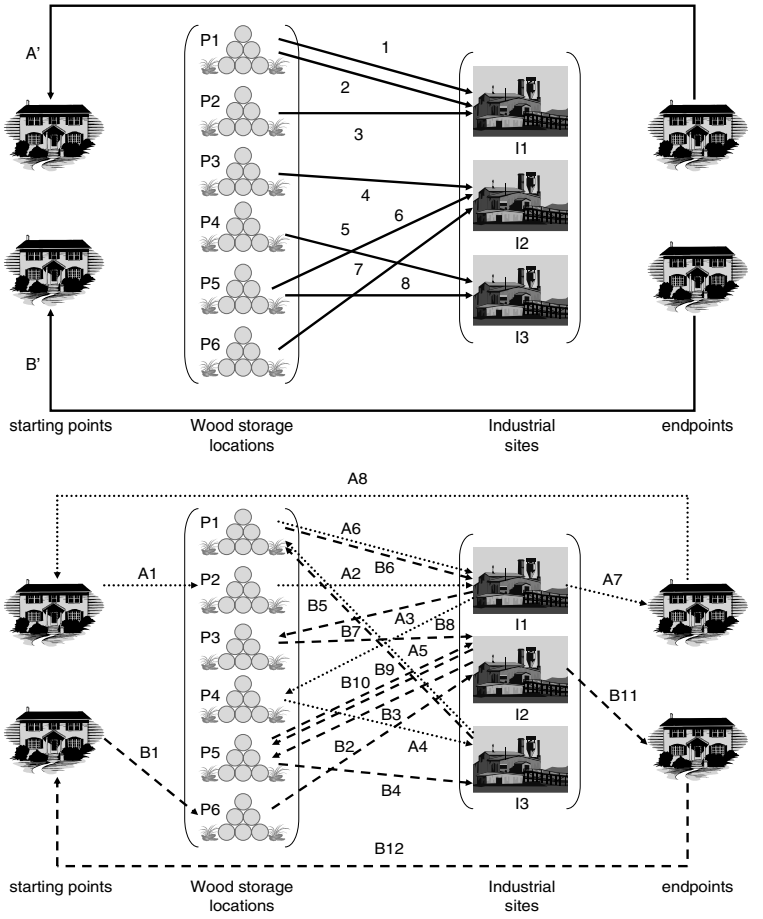

Figure 4-2. Example shown as special case of the SCP

In order to facilitate a further description the following notations are used:

- n-element set of transport tasks W,
- m-element set of artificial tasks V,
- and m-element set of trucks R.

The notation of the elements in V and R is identical. Truck $r \in R$ has a maximum capacity Q_r and a duration limit T_r. The availability time of a truck driver starts at e_r and ends at l_r. A specific route of a truck is named after this truck r.

Each transport task $i \in W$ has the following attributes:

- loading time a_i at the wood storage location,
- route weight limit k_i given in units of weight,
- order quantity q_i,

- unloading time s_i at the industrial site,
- time window $[e_i, l_i]$ at the industrial site,
- and traveling time u_i.

Each truck is allowed to arrive at an industrial site at a time $0 \le b_i \le l_i$; if the truck arrives at a time $b_i < e_i$ it must wait for the period $w_i = e_i - b_i$. t_{ij} represents the time that is needed to move from the endpoint of task i to the starting point of task j; this is the time needed for the empty truck movement.

The following binary decision variables are defined:

- $x_{ijr} = 1$, if task j is visited directly after task i with truck r; 0 otherwise.
- $y_{ir} = 1$, if task i is visited with truck r; 0 otherwise.

The set presented in (1) includes all of the tasks.

$$\tilde{W} = W \cup V \tag{1}$$

The objective function (2) of the model minimizes the duration of empty truck movements.

$$\min \sum_{r \in R} \sum_{i \in \tilde{W}} \sum_{j \in \tilde{W}} t_{ij} \cdot x_{ijr} \tag{2}$$

The following constraints have to be fulfilled:

$$\sum_{i \in \tilde{W}} x_{ihr} - \sum_{j \in \tilde{W}} x_{hjr} = 0 \quad \ldots \forall h \in \tilde{W}, r \in R \tag{3}$$

$$\sum_{r \in R} \sum_{j \in \tilde{W}} x_{ijr} = 1 \quad \ldots \forall i \in \tilde{W} \tag{4}$$

$$\sum_{j \in W} x_{ijr} + x_{iir} = 1 \quad \ldots \forall i \in V, r \in R \quad (i = r) \tag{5}$$

$$\sum_{i \in W} x_{ijr} + x_{jjr} = 1 \quad \ldots \forall j \in V, r \in R \quad (j = r) \tag{6}$$

$$y_{ir} = \sum_{j \in \tilde{W}} x_{ijr} \quad \ldots \forall i \in \tilde{W}, r \in R \tag{7}$$

$$q_i \cdot y_{ir} \le Q_r \quad \ldots \forall i \in W, r \in R \tag{8}$$

$$Q_r \cdot y_{ir} \le k_i \quad \ldots \forall i \in W, r \in R \tag{9}$$

$$\sum_{i \in \tilde{W}} \sum_{j \in \tilde{W}} t_{ij} \cdot x_{ijr} + \sum_{i \in W} u_i \cdot y_{ir} \le T_r \quad \ldots \forall r \in R \tag{10}$$

$$b_i + w_i + s_i + t_{ij} + a_j + u_j - M \cdot (1 - x_{ijr}) \le b_j \quad \ldots \forall i \in \tilde{W}, j \in W, r \in R \tag{11}$$

$$b_i + w_i \geq e_i \quad \ldots \forall i \in \tilde{W} \tag{12}$$

$$b_i \leq l_i \quad \ldots \forall i \in \tilde{W} \tag{13}$$

$$b_i + w_i + s_i + t_{ij} - M \cdot (1 - x_{ijr}) \leq l_r \quad \ldots \forall i \in \tilde{W}, j \in V, r \in R \quad (j = r) \tag{14}$$

$$x_{ijr} \in \{0,1\} \quad \ldots \forall i \in \tilde{W}, j \in \tilde{W}, r \in R \tag{15}$$

$$y_{ir} \in \{0,1\} \quad \ldots \forall i \in \tilde{W}, r \in R \tag{16}$$

$$w_i \geq 0 \quad \ldots \forall i \in \tilde{W} \tag{17}$$

$$b_i \geq 0 \quad \ldots \forall i \in \tilde{W} \tag{18}$$

Constraints (3) guarantee a tour, (4) to (6) define the predecessor and successor relationships, and (7) links the binary variables. Constraints (8) to (10) guarantee the observance of the truck capacity, route weight limits, and maximum travel times. (11) to (14) deal with the time windows at the industrial sites and truck starting points. (15) and (16) define the binary variables. (17) and (18) are non-negativity constraints. We validated our model for small instances, using Xpress-MP software. For real-life problems, it is necessary to develop a customized heuristic.

3. SOLUTION APPROACH

The solution approach consists of the following steps:

1. Restrict the solution space.
2. Find an initial solution with a greedy heuristic.
3. Find an improved solution by applying one of the following Tabu Search procedures:
   a. Standard Tabu Search
   b. Tabu Search with a limited neighborhood
   c. Tabu Search with an alternating strategy
4. Apply a post-optimization heuristic based on 2opt.

The Unified Tabu Search heuristic served as a starting point for our solution procedures. Three variants that differ with respect to the size of their solution space in each iteration step are developed and subsequently discussed.

3.1 Tabu Search

3.1.1 Solution space and initial solution

The overall heuristic commences with a reduction of the solution space. Looking at the problem characteristic, we see that some transport tasks can only be executed by certain truck types. On the one hand, this is because it is impossible to split transport tasks; therefore a truck r with capacity Q_r cannot perform a transport task with an order quantity q_i if $Q_r < q_i$. On the other hand, we have wood storage locations that cannot be reached by each truck type because of the route weight limits. A truck r with capacity Q_r cannot perform a transport task with a route weight limit k_i if $Q_r > k_i$. In the first step, it is guaranteed that a truck r is only assigned to a task that can be handled by this truck with respect to the truck capacity and the route weight limit.

To construct an initial solution we use a regret-heuristic. The gained solution may violate the duration- and time window constraints. The regret-heuristic works in the following way:

- Initialization:
  - For every artificial task $i \in V$: find the closest transport task j and the second-closest transport task z.
  - Calculate a regret-value $REG_i = t_{iz} - t_{ij}$.
  - Sort the regret-values in descending order.
  - Allocate the closest transport tasks to the artificial tasks according to this order; if a transport task is the closest to two or more artificial tasks, it is assigned to the one with the highest regret value.
- Continue with the same procedure until all of the transport tasks are assigned to a tour. Always find the closest and second-closest transport task to the last included task.

3.1.2 Parameter setup

Based on the initial solution a rank indicator B_{ir} with $i \in W$ and $r \in R$ is defined. If $B_{ir} = 0$ this means that transport task i is not on tour (of truck) r. If for example $B_{ir} = 3$ this means that transport task i is ranked third on tour r. While traversing the solution space we apply different notations for marking the solutions: current solution s, a neighbor solution $s°$, the best neighbor solution s', and the best found feasible solution $s*$. The costs associated with a solution are given by $c(s)$ and are equal to the total travel time of the empty trucks. The Tabu Search permits infeasible intermediate solutions. The total violation of tour duration constraints and time windows is denoted by $d(s)$

and $h(s)$, respectively. The variables α and β are used to weight the total violation of constraints. Their values are updated in each iteration step with the help of a parameter δ. α and β are used in order to guide the search process. If we are gaining feasible solutions for a number of iterations, these variables encourage the search process to move to areas with infeasible solutions. If the search process stays in an area with infeasible solutions for a longer time, the search process is driven to areas with feasible solutions. The parameter λ is used to weight the penalizing factor for deteriorating neighbor solutions.

The array ρ_{ir} is used to store how often a transport task i was part of a tour r in a solution s. The tabu status is stored in the array τ_{ir}. We save the information up to which iteration step a task i may not be part of a tour r. An aspiration criterion is used to permit the bypassing of the tabu status. We use fixed tabu durations that are dependent on the number of log-trucks and transport tasks, in which the tabu duration is given by θ. The array σ_{ir} saves the value of the best found feasible solution, in which transport task i was part of tour r. The parameter η gives the number of iteration steps. The function $f(s)$ is equal to the cost function $c(s)$ plus the weighted violations of constraints. The decision function $g(s)$ is used to determine which neighbor solution is chosen; it is equal to $f(s)$ plus a possible penalty function $p(s)$.

3.1.3 Search procedure

The Tabu Search algorithm works as follows:

- Initialization
  - If the initial solution s is feasible set $s^* := s$ and $c(s^*) := c(s)$; else set $s^* := \{\ \}$and $c(s^*) := \infty$.
  - Initialize α and β.
  - For all attributes (i,r):
    - Set $\tau_{ir} := 0$ and $\rho_{ir} := 0$.
    - If the initial solution s is feasible and $B_{ir} > 0$ then set $\sigma_{ir} := c(s)$; else set $\sigma_{ir} := \infty$.
  - Set the parameters δ and λ. We use the following values for these parameters:
    - $\delta \in [0.1, 0.9]$
    - $\lambda \in [0.010, 0.025]$
- For $\kappa = 1$ To η do
  - Determine all neighbor solutions s° of s and their costs $c(s^\circ)$. A neighbor solution s° is generated by moving a transport task i from a tour r to a tour o (move-operator).

- Each transport task i is taken out of its current tour r and tentatively inserted into all of the other tours that fulfill the capacity- and route weight limits.
- If a transport task i is eliminated in a tour r, the direct predecessor and direct successor of i are connected. In its new tour o transport task i is inserted at the position with the least additional costs.
- For each attribute (i,r) that is part of a neighbor solution $s°$, but was not part of solution s, the procedure checks if τ_{ir} is smaller than κ. This means that the attribute is checked as to whether it is tabu or not. If the attribute (i,r) is tabu the algorithm checks if $s°$ is a feasible solution and $c(s°) < \sigma_{ir}$. In this case, the aspiration criterion is fulfilled, in which it is permitted to use this neighbor solution $s°$ despite its tabu status. If the tabu status remains, the value of the decision function to choose a neighbor solution must be set to $g(s°) := \infty$.
- For all neighbor solutions $s°$ that are not tabu or meet the aspiration criterion, the algorithm computes $f(s°)$ and $g(s°)$. If $f(s°) < f(s)$, then set $g(s°) := f(s°)$; otherwise set $g(s°) := f(s°) + p(s°)$. Equation (19) shows the calculation of $f(s°)$; (20) shows the computation of the penalty function $p(s°)$.

$$f(s°) = c(s°) + \alpha \cdot d(s°) + \beta \cdot h(s°) \tag{19}$$

$$p(s°) = \lambda \cdot c(s°) \cdot \sqrt{n \cdot m} \cdot \sum_{i \in W} \sum_{r \in R} \rho_{ir} \qquad ...\forall (i,r) \in s° \tag{20}$$

  The penalty function $p(s°)$ penalizes the neighbor solutions $s°$ for having the same or a higher function value $f(s°)$ as the current solution s. The parameter λ is predefined; n is the number of transport tasks and m the number of log-trucks. The sums over ρ_{ir} count how often attributes (i,r) that are element of $s°$ were part of a solution s.
- The neighbor solution $s°$ that has the lowest value of $g(s°)$ is chosen and called s'.

– After having found the best neighbor solution s' the algorithm continues with the following steps:
  - For each attribute (i,r), which was part of solution s but is not part of s' set $\tau_{ir} := \kappa + \theta$. The tabu duration θ is calculated with Equation (21).

$$\theta = \left\lceil (\log(n \cdot m))^2 \cdot 4 \right\rceil \tag{21}$$

We obtained this formula after a number of parameterization approaches for θ. The value of θ is dependent on the size of the problem according to this formula.

- For each attribute (i,r) that is part of the best neighbor solution s' set $\rho_{ir} := \rho_{ir} + 1$.
- If s' is a feasible solution and $c(s') < c(s^*)$ set $s^* := s'$ and $c(s^*) := c(s')$; otherwise, leave the values of $c(s^*)$ and s^* unchanged.
- If s' is a feasible solution do: for each attribute (i,r) which is part of s' set $\sigma_{ir} := \min\{\sigma_{ir}, c(s')\}$.
- Adjustment of α and β:
  - If $d(s') > 0$ set $\alpha := \alpha \cdot (1 + \delta)$, else set $\alpha := \alpha / (1 + \delta)$.
  - If $h(s') > 0$ set $\beta := \beta \cdot (1 + \delta)$, else set $\beta := \beta / (1 + \delta)$.

– Set $\kappa := \kappa + 1$ and $s := s'$.

- End For

3.1.4 Post-optimization heuristic

A 2-opt based heuristic is applied as a post-optimization procedure after each iteration step of the Tabu Search algorithm. The algorithm attempts to improve single tours by changing the position of two transport tasks. If improvement is attained, the tour is rebuilt accordingly and the same procedure restarts until no further improvement can be found. Per definition, an improvement of a solution is only tolerated if the solution is feasible. The post-optimization procedure does not influence the Tabu Search algorithm; the input data for the next Tabu Search iteration step remains unchanged even if improvement is attained. Only s^* and $c(s^*)$ are updated if the costs $c(s')$ of the post-optimized solution s' are lower than the current best found costs $c(s^*)$.

3.2 New search strategies

The Tabu Search strategy described in Section 3.1.3 implies a search of the entire neighborhood of a solution in each iteration step. We call this strategy hereafter a Standard Tabu Search. This is a very time-consuming procedure since there are no rules to restrict the search space. Therefore, we developed a search strategy that concentrates on the elimination of bad connections between tasks. Toth and Vigo (2003) proposed the Granular Tabu Search in order to restrict the neighborhood of solutions drastically and reduce computing times. They attempt to limit moves that insert “long” arcs in the current solution. Our approach concentrates on a certain fraction of empty truck movements in the current solution s; only these links are to be

removed in neighbor solutions. Other links can only be modified if a task from a removed link is inserted between their starting and ending points.

The procedure functions in the following way: The links are first sorted according to their duration in descending order. Then, a predefined number of links is chosen starting from the one with the longest duration. The number of used links is calculated as a fraction of all the existing empty truck movements; the divider D is set as a parameter. If $D = 4$ this means that one fourth of all the links of a solution s is taken away for being removed in neighborhood solutions.

We call this strategy a Tabu Search with a limited neighborhood. This strategy seems to be myopic since "shorter" links are unaffected directly. To overcome this we merge the Standard Tabu Search and Tabu Search with a limited neighborhood in a new algorithm called a Tabu Search with an alternating strategy. After a predefined number of iteration steps with a restricted neighborhood, an iteration step with a full neighborhood search is set. The parameter A is used to define which iteration steps will be computed with a full neighborhood search. For example, a setting of $A = 8$ means that in every eighth iteration step a full neighborhood search is performed. These new strategies lead to drastic reductions of the computing time. As shown in Section 5, there are also no, or only minimal, losses in the solution quality if the Tabu Search with an alternating strategy is used.

4. NUMERICAL EXPERIMENTS

The small introductory example with eight transport tasks and two trucks can be solved with standard solver software within seconds. Unfortunately, real-life problems have far more trucks and trips to consider. We have observed that regional forest enterprises have to perform approximately 30 transport tasks per day and on average, they operate 10 log-trucks. In the course of a year up to 600 pick-up locations are visited to supply five industrial sites. A large wood processing company in the area operates four sites. In order to ensure a smooth wood supply, up to 250 transport tasks and 80 trucks per day are on order. We estimate their overall yearly number of pick-up locations as 2,500. Murphy (2003) presents a case study with an average of 9 trucks and 35 transport tasks per day for a company situated on the Southern Island of New Zealand. Palmgren et al. (2003) present two case studies for Sweden: one with six trucks and 39 transport tasks, and one with 28 trucks and approximately 85 transport tasks.

In order to test the algorithmic approach for real-life sized problem instances we have developed a random problem generator. Two sets of problem instances have been generated. Each set consists of 20 instances

with 30 transport tasks and 10 trucks. The first set of instances has weaker constraints than the second one in terms of the average task duration and the traveling times between the tasks. In the first instance set, the same 10 truck starting points are used for all of the instances. There are three different industrial sites and 560 possible wood storage locations. In the second instance set, the same 10 truck starting points are also used for all of the instances; but they are different from those of instance set 1. In instance set 2 there are four different industrial sites and 560 possible wood storage locations. In instance set 1, we chose the three industrial sites with the lowest average distance to the 560 wood storage locations out of a set of nine industrial sites; whereas in instance set 2 we use four industrial sites out of this set, which belong to one company and are situated less centrally. This is the reason why we have longer distances in instance set 2.

The model formulation was implemented with the software Xpress-MP. The heuristic solution approach was programmed with Visual Basic 6. We tested the algorithm in the following variants: Standard Tabu Search, Tabu Search with a limited neighborhood, and Tabu Search with an alternating strategy. The post-optimization strategy is only applied in some test runs.

All of the computers used are equipped with a Pentium IV processor with 2.52 GHz and 512 MB RAM; their operating system is Windows XP.

The values of the following parameters were varied in the test runs:

- weighting factor λ for the penalty function $p(s)$,
- parameter δ to update α and β,
- number of iteration steps η,
- divider D,
- and parameter A.

The variables α and β are initialized with the value 1. We use the best found solutions and lower bounds computed with Xpress-MP after a certain computing time as a benchmark for the heuristic solutions. It is also necessary to compare the different variants of Tabu Search with respect to computing times and solution quality. Section 5 shows the results of the numerical studies.

5. RESULTS

The optimal solution for the introductory example with two log-trucks and eight transport tasks can be found within a few iteration steps for all of the variants of the heuristic. The numerical studies were started with a Standard Tabu Search variant, which forbids log-trucks to stay at home and uses no post-optimization strategy. The first test case of each instance set was taken to find the best values for the parameters

$\lambda \in [0.010, 0.025]$ and $\delta \in [0.1, 0.9]$. The resulting values were taken for further computations. Tables 1 and 2 show the deviation from the best found solution for different parameter values. The algorithm is executed for 10,000 iteration steps. In total, we tested 36 parameter variants. In Table 1 and 2, the first row shows the different values for λ and the first column shows the different values for δ. The highest deviation is written in cursive; the shaded cell marks the best found parameterization.

	0.010	**0.015**	**0.020**	**0.025**
0.1	0.1616%	0.1269%	0.0952%	0.0744%
0.2	0.0744%	0.1578%	0.1269%	0.1371%
0.3	0.2492%	0.2175%	0.1688%	0.2492%
0.4	0.1896%	0.1371%	0.1341%	0.2609%
0.5	0.0744%	0.1269%	0.1325%	0.0627%
0.6	0.3338%	0.1269%	0.3965%	0.0000%
0.7	0.3701%	0.4139%	0.3761%	0.0310%
0.8	0.5422%	*0.7416%*	0.6638%	0.1325%
0.9	0.5460%	0.6442%	0.6967%	0.5234%

Table 4-1. Deviation from the best found solution for test case 1 of instance set 1

	0.010	**0.015**	**0.020**	**0.025**
0.1	0.0554%	0.2979%	0.2440%	0.2281%
0.2	0.2042%	0.1332%	0.2245%	0.1781%
0.3	0.0000%	0.2799%	0.0914%	0.0462%
0.4	0.1411%	0.2220%	0.0554%	0.0500%
0.5	0.2973%	0.2695%	0.2339%	0.2440%
0.6	0.1518%	0.5159%	0.3205%	0.0481%
0.7	0.3606%	0.2822%	0.4143%	0.1424%
0.8	0.5824%	0.1518%	0.1424%	0.0941%
0.9	0.3205%	*0.9748%*	0.1054%	0.6251%

Table 4-2. Deviation from the best found solution for test case 1 of instance set 2

Table 3 shows the deviation from the best found solution for all test cases of instance set 1 depending on the number of iteration steps. The best found solution is obtained after 1,000,000 iteration steps. If there is no deviation from the best found solution, the cell is shaded. We have adopted the best found parameter values for test instance 1 with $\lambda = 0.025$ and $\delta = 0.6$. Table 3 provides insight into the speed of convergence. The first row shows the number of iteration steps and the first column shows the different test instances.

	10	100	1,000	10,000	100,000
T1	no sol.	1.5116%	0.4928%	**0.0000%**	**0.0000%**
T2	no sol.	0.4799%	0.4799%	0.0838%	**0.0000%**
T3	no sol.	0.6693%	0.6693%	0.1570%	0.1570%
T4	no sol.	0.5808%	0.5808%	0.0474%	0.0147%
T5	no sol.	1.1163%	0.7436%	0.2793%	0.1573%
T6	no sol.	0.4273%	0.4273%	**0.0000%**	**0.0000%**
T7	no sol.	3.5075%	0.7752%	0.5607%	0.2204%
T8	no sol.	1.1864%	0.8319%	0.2374%	**0.0000%**
T9	no sol.	0.5149%	0.5149%	0.3448%	0.0127%
T10	no sol.	0.6192%	0.6192%	0.4733%	0.1605%
T11	no sol.	0.4182%	0.4182%	0.4182%	0.0210%
T12	no sol.	0.8338%	0.8338%	0.2614%	0.0546%
T13	no sol.	1.0375%	0.2896%	0.2127%	**0.0000%**
T14	no sol.	0.2859%	0.2859%	0.0868%	0.0868%
T15	no sol.	0.1770%	0.1770%	0.1770%	**0.0000%**
T16	no sol.	0.5266%	0.1911%	0.1911%	0.1499%
T17	no sol.	0.3748%	0.3748%	0.2933%	0.1484%
T18	no sol.	1.6211%	0.9737%	0.2782%	0.1476%
T19	no sol.	0.2550%	0.2550%	0.1807%	0.0464%
T20	no sol.	0.4012%	0.4012%	**0.0000%**	**0.0000%**

Table 4-3. Deviation from the best found solution after 1,000,000 iteration steps depending on the number of performed iteration steps for instance set 1

With 1,000 iteration steps the solution values of all test instances of set 1 are less than 1% worse than the best found solution. It takes approximately 150 seconds to perform 1,000 iteration steps with the Standard Tabu Search. Since we can estimate a linear relationship between computing times and the number of iteration steps we only need about a tenth part of the computing time for 10,000 iteration steps.

Table 4 shows the same data as Table 3 for all of the test instances of instance set 2 depending on the number of iteration steps. We also used the best found parameterization for test instance 1 with $\lambda = 0.010$ and $\delta = 0.3$.

	10	100	1,000	10,000	100,000
T1	no sol.	2.2515%	0.4010%	**0.0000%**	**0.0000%**
T2	no sol.	1.8074%	1.8074%	1.8074%	0.8177%
T3	no sol.	5.9537%	2.5884%	0.1172%	**0.0000%**
T4	no sol.	3.2998%	0.0548%	0.0438%	**0.0000%**
T5	no sol.	2.6279%	0.9880%	0.1037%	**0.0000%**
T6	no sol.	5.4487%	2.1683%	1.2378%	0.5272%
T7	no sol.	5.5435%	**0.0000%**	**0.0000%**	**0.0000%**
T8	no sol.	0.1990%	0.1990%	0.1990%	0.1944%
T9	no sol.	3.4008%	0.9153%	0.8786%	0.0002%
T10	no sol.	1.0430%	1.0430%	0.8509%	**0.0000%**
T11	no sol.	0.8382%	0.8382%	0.4553%	0.1801%
T12	no sol.	1.3593%	1.2256%	0.2803%	0.2803%
T13	no sol.	1.7042%	1.4113%	0.1400%	0.1400%
T14	no sol.	2.4080%	1.3224%	0.8963%	0.0270%
T15	no sol.	3.1242%	0.9137%	0.9137%	0.0113%
T16	no sol.	9.7840%	2.5547%	1.6086%	0.2714%
T17	no sol.	1.3800%	1.3800%	1.1039%	**0.0000%**
T18	no sol.	6.1360%	2.5321%	1.2930%	0.0285%
T19	no sol.	3.1343%	0.7194%	0.1164%	0.0589%
T20	no sol.	1.5204%	0.7800%	0.1743%	0.0223%

Table 4-4. Deviation from the best found solution after 1,000,000 iteration steps depending on the number of performed iteration steps for instance set 2

Table 5 compares the average deviation from the best found solution for all test cases of instance set 1 and 2 in the range of 100 to 100,000 iteration steps. It summarizes the results of Table 3 and Table 4. One can observe that it is possible to find solutions of good quality in less computing time, for instance set 1. We assume that the tighter constraints of instance set 2 make it more difficult to find feasible and good quality solutions.

	100	1,000	10,000	100,000
instance set 1	0.8272%	0.5168%	0.2141%	0.0689%
instance set 2	3.1482%	1.1921%	0.6110%	0.1280%

Table 4-5. Average deviation from the best found solution after 1,000,000 iteration steps depending on the number of performed iteration steps for instance sets 1 and 2

Furthermore, we compared the different variants of Tabu Search with respect to computing times and solution quality. For performing this, we used test case 1 of instance set 1 to compute 10,000 iteration steps with the different Tabu Search variants. In all of the Tabu Search variants, we did not permit unemployed log-trucks. We also applied different parameter values of the divider D and varied the sequence of full neighborhood search iteration steps. The parameter λ was set to 0.025; δ was set to 0.6. Table 6 shows in its

first column the used Tabu Search variant and the respective parameterization, in the second, the time deviation from the lowest computing time, and in the third, the deviation of the solution value from the best found solution value is displayed. All Tabu Search variants in Table 6 are computed without a post-optimization strategy. We also applied the post-optimization heuristic to the Tabu Search with an alternating strategy. It turned out that the post-optimization was able to improve the best found solution within the first iteration steps, but after 10,000 iteration steps we obtained the same results as when not using it. The additional computing time for the post-optimization can only be determined empirically for each test instance; roughly spoken, one can expect an increase of approximately 10%.

The following abbreviations are used for the Tabu Search variants in the below-mentioned text: Standard Tabu Search (TS), Tabu Search with a limited neighborhood (TSLN), and Tabu Search with an alternating strategy (TSAS).

One can observe that the TS has the highest computing time of all the variants and offers a solution quality that is close to the best found solution. We tested four parameterizations of the TSLN. The divider D determines which portion of the connections between the transport tasks is removed in neighboring solutions. The results show that the lowest computing time (203 seconds) is reached with a TSLN and a divider $D = 8$. However, this method also offers the worst solution quality, which is in turn unacceptable. When the TSLN is used with $D = 2$, a quite good solution quality is obtained in reasonable computing time. Nevertheless, the TSLN is a myopic strategy. Some parts of the neighborhood are excluded permanently. The TSAS seems to be a good way to overcome this problem. A look at the results shows that it is able to reduce computing times drastically with little or no loss in solution quality. As the results show, it is sufficient to search the full neighborhood in every eighth iteration step.

Method	Time deviation	Solution value deviation
TS	673.40%	0.18%
TSLN D = 2	227.59%	0.27%
TSLN D = 4	90.64%	2.09%
TSLN D = 6	33.00%	7.31%
TSLN D = 8	0.00% (203 s)	7.69%
TSAS D = 2 A = 2	444.33%	0.29%
TSAS D = 4 A = 2	373.89%	0.00%
TSAS D = 6 A = 2	348.77%	0.08%
TSAS D = 8 A = 2	326.11%	0.03%
TSAS D = 2 A = 8	287.68%	0.05%
TSAS D = 4 A = 8	159.11%	0.00%
TSAS D = 6 A = 8	110.84%	0.06%
TSAS D = 8 A = 8	78.82%	0.00%

Table 4-6. Comparison of the different Tabu Search variants for test case 1 of instance set 1 for 10,000 iteration steps

We also compared the results of the different Tabu Search variants after a fixed computing time. Table 7 shows the results for the same Tabu Search variants and parameterizations as Table 6. The running time was chosen as the average of the running times of the different Tabu Search variants for 10,000 iteration steps (696 seconds). The second column of Table 7 shows the number of iteration steps in this time, and the third, the deviation from the best found solution. The best found solution has the same value in both comparisons.

Method	Iteration steps	Solution value deviation
TS	4,430	0.18%
TSLN D = 2	10,459	0.27%
TSLN D = 4	17,973	2.09%
TSLN D = 6	25,762	7.31%
TSLN D = 8	34,263	7.69%
TSAS D = 2 A = 2	6,295	0.29%
TSAS D = 4 A = 2	7,230	0.15%
TSAS D = 6 A = 2	7,635	0.08%
TSAS D = 8 A = 2	8,041	0.03%
TSAS D = 2 A = 8	8,838	0.05%
TSAS D = 4 A = 8	13,223	0.00%
TSAS D = 6 A = 8	16,251	0.06%
TSAS D = 8 A = 8	19,161	0.00%

Table 4-7. Comparison of the different Tabu Search variants for test case 1 of instance set 1 after 696 seconds running time

In the following the mean solution value of three arbitrary test cases namely 1, 10, and 20 of instance set 1 for 10 to 1,000,000 iteration steps is displayed. We forbade unemployed log-trucks and applied no post-optimization. In Figure 3, the abscissa represents the number of iteration steps; the ordinate shows the deviation from the best found solution; if the deviation is equal to 10%, this means that no feasible solution was found for one or more test cases up to this iteration step. The solution quality does not improve significantly for the TSLN with a divider D equal to 6 and 8 if more than 100 iteration steps are computed; the same is true with more than 10,000 iteration steps for a divider D equal to 2 and 4. With the TSAS and the TS solutions of good quality can be obtained. Even with a very fast Tabu Search variant (TSAS with $D = 8$, full neighborhood search in every eighth iteration step) the deviations from the best found solution are far less than 1% after 1,000 iteration steps. The bars in figures 3 and 4 are ordered in the same way as the sequence of the legend.

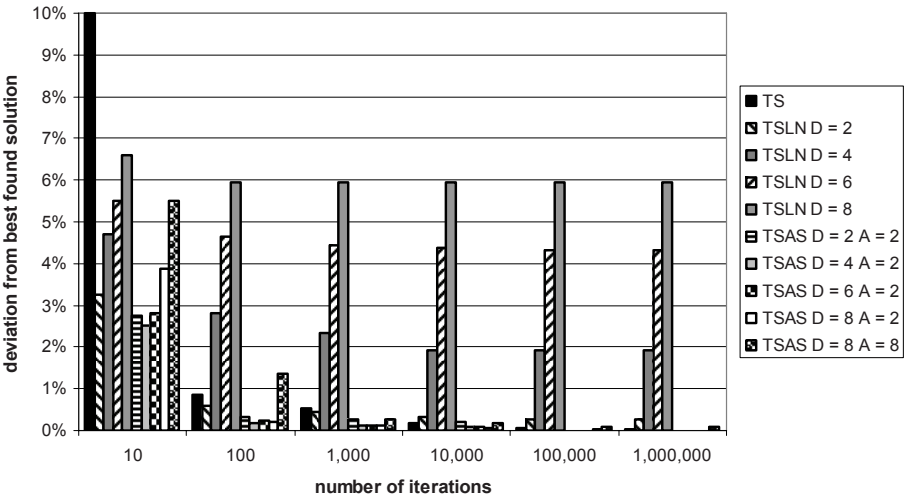

Figure 4-3. Average deviation from the best found solution for test cases 1, 10, and 20 of instance set 1

Figure 4 shows the average deviation from the best found solution for test cases 1, 10, and 20 of instance set 2. Since instance set 2 has tighter constraints, it is more difficult to find feasible solutions. Even after 1,000,000 iteration steps with a TSLN no feasible solution for the dividers $D = 4$, $D = 6$, and $D = 8$ can be obtained. The TSLN achieves a solution of good quality only for a divider $D = 2$. The TS and the TSAS are able to find

solutions that are equal to the best found solution with one exception; the TSAS with a divider $D = 8$ and a full neighborhood search in every eighth iteration step seems to be improper for solving problems with tight constraints. However, if the divider is reduced it is also possible to improve the solution quality in this case.

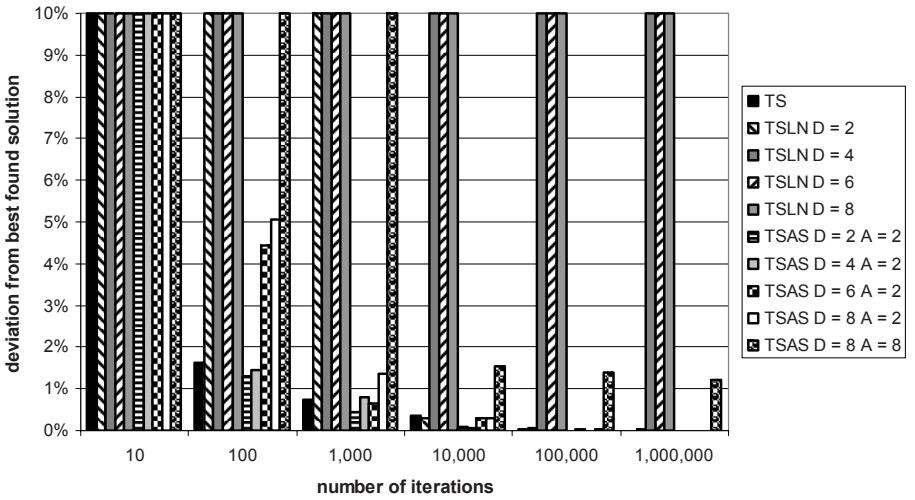

Figure 4- 4. Average deviation from the best found solution for test cases 1, 10, and 20 of instance set 2

Due to the computational complexity of the TTVRP, it is clear that standard solver software is unsuitable for these problems. However, to benchmark the heuristics, we compared in Tables 8 and 9 their best found solutions after 10,000 iteration steps with the best found solution obtained with the solver software Xpress-MP for the same computing time. In this comparison, we permitted log-trucks to stay at home and applied a post-optimization heuristic to the Tabu Search variants. Table 8 shows the results for test case 1 of instance set 1. In Table 8 for example the TSAS with $D = 8$ and a full neighborhood search in every second iteration needs 825 seconds for 10,000 iteration steps. This provides a solution value of 2,603.52. For the same timeframe, the Xpress solver provides a value of 2,640.13. Even after a computing time of 24 hours, the solver software obtains a solution value (2,603.12) that is worse than most heuristic solution values. The lower bound after 24 hours is equal to 2,593.37; but this solution may represent an infeasible solution to the problem.

Method	Time [sec]	sol. val.	sol. val. Xpress Solver	Deviation
TS	1,447	2,597.52	2,616.16	0.7125%
TSAS D = 8 A = 2	825	2,603.52	2,640.13	1.3867%
TSAS D = 4 A = 8	503	2,598.33	2,702.59	3.8578%
TSAS D = 6 A = 8	413	2,599.72	2,767.19	6.0520%
TSAS D = 8 A = 8	328	2,602.79	no solution	no solution

Table 4-8. Comparison of solution values after certain computing times for test case 1 of instance set 1

Table 9 shows the results for test case 1 of instance set 2. The best found parameter values of Tabu Search variants that allow log-trucks to stay at home and variants that do not permit this differ in instance set 2. Therefore the parameter λ was set to 0.015; δ was set to 0.6. Since there are tighter constraints, the solver software could not find feasible solutions within the computing times needed by heuristics. Even after a computing time of 24 hours, the solver software obtains only one feasible solution with a value of 5,617.76 that is much worse than the heuristic solutions. The lower bound after 24 hours is equal to 3,786.93.

Method	Time [sec]	sol. val.	sol. val. Xpress Solver	Deviation
TS	1,440	4,749.97	no solution	no solution
TSAS D = 8 A = 2	820	4,773.07	no solution	no solution
TSAS D = 4 A = 8	497	4,784.72	no solution	no solution
TSAS D = 6 A = 8	438	4,774.76	no solution	no solution
TSAS D = 8 A = 8	344	4,772.40	no solution	no solution

Table 4-9. Comparison of solution values after certain computing times for test case 1 of instance set 2

Additional comparisons were made for other test cases also. It turned out that all of the heuristic solutions were better than the best found solutions obtained with Xpress-MP after the same computing time.

6. CONCLUSION

In this paper, we presented a formal description of the TTVRP, which was only described verbally in the literature up to now. We have also developed a heuristic solution approach based on a Tabu Search with different neighborhood structures. The numerical studies show that the proposed heuristics are able to solve real-life problem instances in reasonable computing times with good solution quality. The heuristics

perform rather well if they are compared to the best found solutions of solver software as a benchmark.

The TSAS is a good method to reduce computing times and keep the solution quality nearly constant. This approach can also be enhanced with a dynamic component; instead of fixing the iteration steps with a full neighborhood search, one can also make the neighborhood structure dependent on the solution quality. If the solution quality does not improve for a certain number of iteration steps with a limited neighborhood (this number could be a function of the total number of iteration steps) one can set an iteration step with a full neighborhood search. The development of further variants of the TSAS could also bring forth benefits for other research areas that use a Tabu Search as a solution method. The TS is recommendable if there are tight constraints, in which feasible solutions are difficult to find; but it is also worth attempting to use the TSAS with frequent iteration steps with a full neighborhood search for such problem instances. Even though the TSLN offers a reduction in computing times compared to the TS, it is not recommendable since it is myopic, and therefore, the search process is locked very often in local optima for a large number of iteration steps.

We can also observe that the improvements in solution quality have not been significantly compared to the additional computing times after 10,000 iteration steps. We can conclude that the heuristic solution approaches quickly converge to solutions with good quality. This fast speed of convergence may be an indication for being locked in local optima; but if we look at the intermediate solutions, we can notice that this is not the case, and the diversification strategy of the Tabu Search heuristics is working well.

Future research will concentrate on an extension of the planning horizon of this scheduling problem. The current method is able to optimize the routing of log-trucks during a given timeframe, which is generally one day. When the planning horizon is extended to one week, an evenly distributed workload among the days of that week cannot be assured. Since this is an important factor for industrial sites in the wood industry, it is necessary to introduce a model formulation that first performs an optimal allocation of the transport tasks to single days of the week. Subsequently, the current method can be reused. As the results show, it also makes sense to put forth additional effort toward the enhancement of the TSAS.

Acknowledgements:
The authors would like to thank the reviewers and the auditorium at the MIC 2005 for their useful comments on this contribution.

REFERENCES

Coja-Oghlan, A., Krumke, S. O., Nierhoff, T. (2004): A Heuristic for the Stacker Crane Problem on Trees which is Almost Surely Exact. Journal of Algorithms, In Press.

Cordeau, J. F., Laporte, G., Mercier, A. (2001): A unified tabu search heuristic for vehicle routing problems with time windows. Journal of the Operational Research Society **52**, 928-936.

Glover, F. and Laguna, M. (1997): Tabu Search. Kluwer Academic Publishers, Boston, USA.

Gronalt, M., Hartl, R., Reimann, M. (2003): New savings based algorithms for time constrained pickup and delivery of full truckloads. European Journal of Operational Research **151(3)**, 520-535.

Karanta, I., Jokinen, O., Mikkola, T., Savola, J., Bounsaythip, C. (2000): Requirements for a Vehicle Routing and Scheduling System in Timber Transport. Proceedings of the IUFRO International Conference: Logistics in the forest sector, 235-250.

Murphy, G. (2003): Reducing Trucks on the Road through Optimal Route Scheduling and Shared Log Transport Services. Southern Journal of Applied Forestry **27(3)**, 198-205.

Palmgren, M., Rönnqvist, M., Värbrand, P. (2003): A solution approach for log truck scheduling based on composite pricing and branch and bound. International Transactions in Operational Research **10**, 433-447.

Righini, G. and Trubian, M. (1999): Data-dependent bounds for the General and the Asymmetric Stacker-Crane problems. Discrete Applied Mathematics **91**, 235-242.

Schwarzbauer, P. (2005): Die österreichischen Holzmärkte: Größenordnungen – Strukturen – Veränderungen. Lignovisionen Band 8, Universität für Bodenkultur Wien.

Toth, P. and Vigo, D. (eds.) (2002): The Vehicle Routing Problem. SIAM, Philadelphia, USA.

Toth, P. and Vigo, D. (2003): The Granular Tabu Search and Its Application to the Vehicle-Routing Problem. INFORMS Journal on Computing **15(4)**, 333-346.

Weintraub, A., Epstein, R., Morales, R., Seron, J., Traverso, P. (1996): A Truck Scheduling System Improves Efficiency in the Forest Industries. Interfaces **26(4)**, 1-12.