Intelligent Deployment of Forest Road Graders

Matthew Thompson
Kevin Boston
Jeff Arthur
*John Sessions**

ABSTRACT

Road grading is the most common maintenance activity performed on forest roads. Reducing grading cost could release resources for other maintenance needs, ideally resulting in a better maintained road system. A combinatorial optimization method, tabu search, is combined with two local search procedures to generate efficient grading routes. Determining the optimal grading route is modeled as an extension of the Mixed Rural Postman Problem (MRPP), adapted to include a daily operating time limit and different traversal/service times. The objective was to minimize total operating time, a proxy for grading cost. The heuristic was tested on both artificial and actual forest road networks, and computational results are presented. The heuristic demonstrates the ability to generate efficient and feasible grader routes.

Keywords: *road maintenance, grading, optimization, tabu search*

Introduction

Road Maintenance and Grading

Road grading is the most common maintenance activity performed on forest roads. A grader operates by traversing the road with a lowered blade, mixing and redistributing the aggregate surfacing material to provide a smooth, pothole-, rut-, and washboard-free surface. As with other forms of road maintenance, grading has both economical and environmental benefits. A well-graded road reduces haul costs by allowing vehicles to travel at design speeds, and reduces vehicle ownership and operating costs by reducing the amount of wear and tear incurred while traveling. Insurance costs could potentially be reduced due to enhanced driver safety. Proper grading also serves to maintain the road's designed drainage system, which reduces environmental degradation associated with erosion and sedimentation.

The transportation manager is often responsible for ensuring that the road network meets some minimum acceptable levels for driver safety, vehicle traffic capacity, and drainage. Budgetary constraints limit the amount of maintenance a planner can implement in a given period. Identifying cost-saving measures, therefore, could release these resources for other maintenance projects, ideally resulting in a better maintained road network. As grading cost for a road system is driven largely by total operating time (Dave Young, pers. comm.), emphasis should be placed on minimizing grader operating time for that system. Additionally, careful selection of road segments that are actually in need of maintenance can reduce overall costs by reducing the number of road segments treated. In a study performed in eastern Canada, for example, it was found that identifying segments in poor condition and scheduling grading accordingly could reduce grading costs by 30 percent over a policy of grading at fixed time intervals (Provencher 1995). Limiting unnecessary grading can also limit potential environmental degradation from grading, which is associated with increased sediment yields due to disruption of the road surface (Luce and Black 1999).

In this paper, a novel method for assisting grader deployment decisions is proposed, the goal of which is to determine minimal cost tours that visit all road segments requiring grading. The study will demonstrate that use of computerized decision aids to determine routes for graders to take when maintaining a road network could result in increased efficiency. Also, a "monitor and manage" approach to road grading scheduling is more appropriate than adhering to unstructured or uniform predetermined grading schedules.

Grading Decisions

First, the manager must decide which road segments require grading. Typically the decision to grade is based upon weather and traffic patterns, as well as observations and complaints from drivers. Rarely are these decisions based upon quantifiable information regarding road roughness or water content. Recent technological developments, however, provide for improved data collection and analysis, which could inform and improve the grading decision process. A good example of progress in this field is the Opti-Grade® package, developed by the Forest Engineering Research Institute of Canada (FERIC 2007). Opti-Grade includes a device to be installed on a vehicle that can measure and record road rough-

The authors are, respectively, Graduate Student (Matthew.thompson@oregonstate.edu), Dept of Forest Engineering, Oregon State University; Assistant Professor, Dept of Forest Engineering, Oregon State University; Professor, Dept of Statistics, Oregon State University; and Distinguished Professor, Dept of Forest Engineering, Oregon State University, Corvallis, OR.This paper was received for publication in September 2006.
*Forest Products Society Member.

International Journal of Forest Engineering 18(2): 15-23.

ness values as the vehicle traverses a road network. The roughness data is paired with global positioning system (GPS) time and location data to provide an objective, quantitative measure of the road network's roughness levels, which is then analyzed by the package's software to determine which segments require grading. Opti-Grade determines grading requirements according to roughness thresholds identified by managers, but similar technology could implement additional decision criteria, such as road category, season of use, and volume of traffic.

Grader Routing

After the subset of road segments that require grading has been established, the transportation manager must then determine the route the grader will take over the road network. This route, or tour, must visit all road segments that the manager has identified as requiring grading. In practice, knowledge about local conditions, experience, and a roadmap are the primary drivers of this decision process (Dave Young, pers. comm.). As previously mentioned, the manager seeks to minimize total project cost, or as a proxy, total grader operating time. Numerous possible routes exist that traverse all of the segments requiring grading, and the manager must select an efficient tour from among them. For networks of substantial size, the pool of possible solutions is far too vast to be evaluated by hand, and the manager must decide upon the route according to some heuristic process.

In mathematical terms, the generic problem facing the transportation manager is known as the Rural Postman Problem (RPP), part of a broader class of Arc Routing Problems (ARPs). The aim of the RPP, and ARPs in general, is to determine a minimal cost closed walk of a subset of some arcs of a graph (Eiselt et al. 1995a, 1995b). A closed walk is a sequence of arc traversal over a network, with the sequence beginning and ending at the same point. The rural postman, for instance, leaves the post office in the morning, visits some subset of county roads to deliver mail, and returns to the post office at the end of the day. The RPP formulation provides a basis for many other industrial applications, including street sweeping, school bus routing, snow plowing, and garbage collection (Corberán et al. 2000, Eiselt et al. 1995b). In this context, road segments represent arcs and road intersections represent vertices on a graph. To accurately reflect most real forest road networks, the underlying graphs are considered mixed, meaning they contain both arcs (one-way) and edges (two-way). The manager's problem is, therefore, an instance of the Mixed Rural Postman Problem (MRPP), defined as below (Eiselt et al. 1995b):

> Let G = (V, E, A) be a connected mixed graph, where V is the vertex set, A is the arc set, E is the edge set, and a nonnegative cost is associated with all arcs and edges. Define $A_R \subseteq A$ and $E_R \subseteq E$ to be the subsets of arcs and edges, respectively, that require grading. The aim of the MRPP is to find a minimal cost closed walk that traverses each required arc and edge at least once.

The RPP and MRPP are proven to be NP-Hard; there is no nondeterministic polynomial-timed algorithm able to solve this class of problems. These problems types generally require a heuristic method to generate near optimal solutions (Corberán et al. 2000, Eiselt et al. 1995b). In a survey of ARPs, Eiselt et al. (1995b) conclude that arc routing theory could benefit from additional research, and they identify tabu search as a direction for future research. The success of tabu search applied to Vehicle Routing Problems (VRPs) is cited as an example. Gendreau et al. (1994) developed TABUROUTE, a tabu search heuristic for VRP with capacity and route length restrictions, which upon testing outperformed existing heuristics. Tabu search was later applied to the Capacitated Arc Routing Problem (CARP) with similar success (Hertz et al. 2000). Most relevant, a tabu search heuristic was implemented in concert with a construction heuristic for the MRPP, and the authors (Corberán et al. 2000) considered tabu search to have performed "remarkably well."

Tabu search is a commonly used and effective combinatorial heuristic algorithm. Developed by Glover (1989, 1990), tabu search uses memory structures to characterize and prioritize candidate solutions. It is essentially an improvement method in which successive solutions, which are perturbations of previous solutions, are examined and the best is selected. Ideally, an improving solution will be found, but where none of the perturbations would result in an improvement the best solution is still selected. Thus, in a given iteration, the objective function may worsen, but it is hoped that doing so will allow the heuristic to escape local optima. In addition, a tabu list is used that identifies forbidden moves; this prevents cycling and ideally prevents the heuristic from keying in on local optima. Essentially, tabu search can be viewed as a metaheuristic that can guide any local search procedure toward a global optimum. Utilization of heuristics, such as tabu search, provide the opportunity to evaluate increasingly large pools of data and alternative scenarios and to seek optimal solutions to complex problems.

Tabu search has been successfully implemented in various vehicle routing and arc routing problems (Amberg et al. 2000, Corberán et al. 2000, Hertz et al. 2000, Gendreau et al. 1994). Additionally, tabu search has been widely implemented in the field of forest management (Richards and Gunn 2003). Examples include harvest scheduling (Boston and Bettinger 2002, Bettinger et al. 1999, Bettinger et al. 1997), wildlife planning (Bettinger et al. 2002), landscape planning with environmental goals (Bettinger et al. 1998), road optimization (Aruga 2005), and logging crew assignment (Murphy 1998).

Like most applications, the transportation manager's problem includes additional characteristics that require modifications from the original MRPP formulation. In this case, a daily maximum operable time limit is included, which has the potential to greatly increase the complexity of the problem. In excessively steep terrain, the grader may be limited to unidirectional travel or may be forced to reduce speeds due to safety and sliding concerns (Caterpillar Inc. 2002). These lim-

its were incorporated into our model by assuming the extreme case of different grading speeds for uphill and downhill travel. This assumption introduces elements of the Windy Postman Problem (WPP), in which the cost of traversal depends on the direction of travel; the WPP is also proven to be NP-Hard (Eiselt et al. 1995a). Where not appropriate, the formulation can easily be amended to include uniform grading speeds, which in fact reduces the complexity of the problem by reducing the size of the solution space. Traversal over road segments without grading (deadheading) is assumed at a constant speed.

Thus, the solution methodology can draw from MRPP theory but will necessarily require a tailored approach. A tabu search metaheuristic algorithm was proposed, combined with two improvement procedures that seek efficient grader routes across the road network while incorporating the operational constraints discussed. Upon obtaining an acceptable solution, the manager can deploy the grader to begin operation.

In practice grading time depends on the number of passes required to service the road segment, a function largely of road width and standard. For this paper, all road segments were assumed to require a single pass, but variable pass requirements for different road segments can easily be included – grading time is a model parameter that can be updated appropriately. Where an even number of passes is required, the algorithm would need to be updated to continue traversal from the beginning of the recently graded road segment rather than the end. Including number of passes, however, would not fundamentally change problem complexity or our solution approach, and thus its consideration was omitted from the example application presented below. Rather, considerations such as an operating time limit and directional-dependent speeds were chosen to increase problem complexity and challenge the solution method.

Notation and Definitions

Edges henceforth generically refer to both arcs and edges and are defined by a vertex pair (*i, j*). Here *i* is the "from" vertex, and *j* is the "to" vertex defining the edge. If an edge is undirected, the edge defined by (*j, i*) also exists. Edges that require grading are said to require service, are denoted as s*ervice edges*, and are represented as [i, j]. A *tour* is the order in which service edges are visited. A given tour may be comprised of multiple *sub-tours*, which are necessary when the overall tour exceeds the daily time capacity. All tours (including subtours) originate and end at the *depot* vertex, where the grader is stored when not in use. A grader traversing an edge without grading is said to be *deadheading*. A *spur* is an edge only connected to the network via one vertex, i.e., the edge comes to a deadend.

Tabu Search

Our tabu search heuristic iteratively generates alternative solutions (tours) by re-ordering the traversal of service edges. Total tour time is calculated as the sum of all grading times plus the time to traverse between service edges. Note this calculation includes time to return to the depot at the end of each sub-tour, if required due to operating time constraints. When traveling between edges requiring service, the grader is assumed to follow the shortest path between service edges. Specifically, this shortest path originates at the "to" vertex of the service edge graded prior and ends at the "from" vertex of the next service edge where grading will begin. When traversing between two service edges, the grader may deadhead over a third service edge, and failing to grade that service edge along the way may result in an inefficient solution. The Shortest Path Service Edge Insertion (SPSEI) heuristic, described below, addresses this concern. When a tour length extends beyond the maximum allowable time, it must be divided into sub-tours that are feasible. We developed the Tour Partition Heuristic to address this concern, and it is described below. The tabu search seeks an efficient ordering of service edges that results in an overall minimal closed-walk time.

Neighborhood Definition

Each instance of traversal re-ordering is considered a "swap." The neighborhood of the search procedure includes both 1-edge and 2-edge swaps. One-edge swaps are only applicable for undirected service edges; the direction in which it is graded is simply reversed. A 2-edge swap switches the location of two service edges in the tour. Service edges that are undirected can be inserted into their new locations in either possible traversal direction; the procedure evaluates both possibilities. All possible combinations of service edge swaps are considered per iteration of the tabu search, to provide as broad a scope as possible.

To illustrate the allowable swaps, consider the example network below in **Figure 1**. The network contains 5 vertices (depot, A, B, C, D) and seven undirected edges. The black dot represents the depot. Two edges require service: [A, B] and [C, D]. **Table 1** displays the average deadhead and grading times for these edges. Note that deadhead times are the same for either direction, whereas grading times vary with direction.

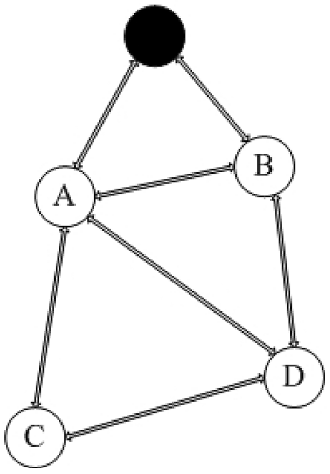

Figure 1. ~ *Example network.*

Table 1. ~ *Example deadhead and grade times.*

Deadhead		Grade	
Edge(s)	Time	Edge	Time
(DEP, A) and (A, DEP)	1.7	[A, B]	4.5
(DEP, B) and (B, DEP)	1.3	[B, A]	3
(A, B) and (B, A)	1.5	[C, D]	8.4
(A, C) and (C, A)	2.3	[D, C]	5.6
(A, D) and (D, A)	2.4		
(B, D) and (D, B)	1.8		
(C, D) and (D, C)	2.8		

Assume the current solution is to traverse the service edges in the order [A, B] → [C, D]. **Figure 2a** displays the resulting tour the grader would traverse over the network for this solution. Bold lines represent service edges; arrowheads indicate direction of traversal.

Current Solution: [A, B] → [C, D]

Total Tour: (DEP, A) → [A, B] → (B, A) → (A, C) → [C, D] → (D, B) → (B, DEP)

Total Time: 21.5

Table 2 lists all possible 1- and 2-edge swaps for this original candidate solution.

Figure 2b through 2d present a subset of eligible swaps, with the resulting tours and associated total times listed below. Because predetermined shortest paths are traversed between grading service edges, the tour can substantially change when swaps are performed.

1-edge swap: [B, A] → [C, D]

Total Tour: (DEP, B) → [B, A] → (A, C) → [C, D] → (D, B) → (B, DEP)

Total Time: 18.1

2-edge swap: [C, D] → [A, B]

Total Tour: (DEP, A) → (A, C) → [C, D] → (D, A) → [A, B] → (B, DEP)

Total Time: 20.6

2-edge swap: [D, C] → [A, B]

Total Tour: (DEP, B) → (B, D) → [D, C] → (C, A) → [A, B] → (B, DEP)

Total Time: 16.8

Tabu Search Implementation Details

Initial solutions are generated as a random permutation of service edge ordering. The tabu search procedure is then invoked to improve upon the random initial solution. The maximum number of iterations was set at 10,000, where per iteration a neighborhood search is performed to select a new solution. The neighborhood search consists of an exhaustive investigation into all possible 1-and 2-edge swaps, as detailed above. The shortest path service edge insertion (SPSEI) heuristic is invoked for each candidate move to investigate the potential to reduce tour redundancy. A redundancy, as defined here, would be to deadhead over a service edge while traversing the shortest path between two other service edges. If necessary, the Tour Partition heuristic is also invoked to divide the candidate tour into feasible sub-tours. The inclusion of the two sub-heuristics makes calculations of the change in solution quality rather difficult, so the decision was made to

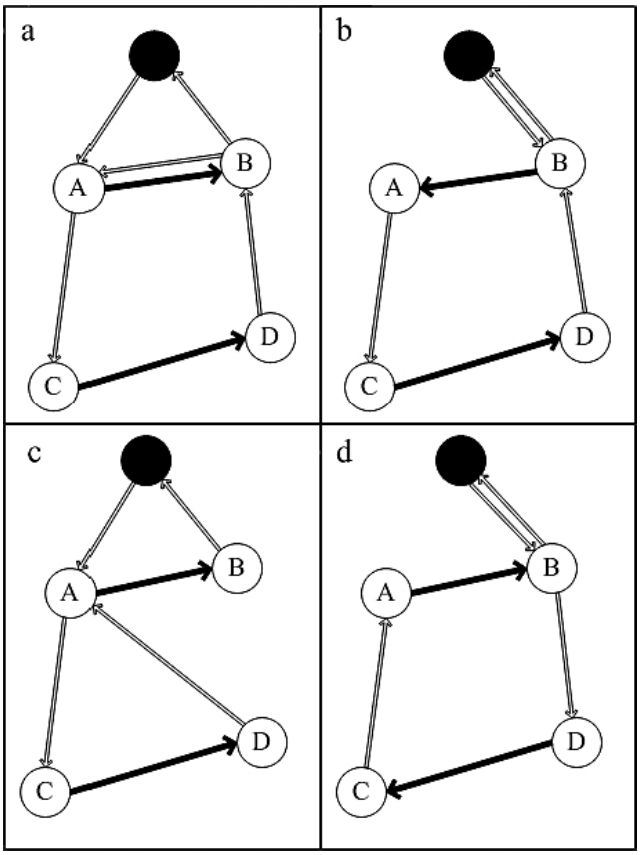

Figure 2. ~ *Example solution and possible swaps. Bold lines represent service edges; arrowheads indicate direction of traversal.*
a) Current solution: [A, B] → [C, D],
b) 1-edge swap: [B, A] → [C, D],
c) 2-edge swap: [C, D] → [A, B], and
d) 2-edge swap: [D, C] → [A, B].

Table 2. ~ *Possible candidate solutions from swaps.*

1-edge swaps	2-edge swaps
[B, A] → [C, D]	[C, D] → [A, B]
[A, B] → [D, C]	[C, D] → [B, A]
	[D, C] → [A, B]
	[D, C] → [B, A]

recalculate total tour time for each candidate move after invoking the two sub-heuristics.

Diversification criteria included both recency and frequency memory structures. The recency memory structure is a tabu list of fixed length, which varied depending on problem size. Values of 15 through 25 were used, which were determined by initial experimentation and analysis of solution quality. The frequency memory structure maintained the number of times moves, or swaps, were selected – the idea being to penalize moves that are selected more frequently to avoid returning to previously visited local optima too quickly. The aspiration criterion stipulated that moves considered tabu can only be selected if their solution time is the best time achieved yet.

Per iteration the swap providing the greatest improvement in solution is selected, provided it is not currently on the tabu list. Pursuant to the aspiration criteria, a tabu move must yield the best solution to be permissible. The selected move is then entered into the tabu list (if not already there), replacing the least recently added move. The selected swap's frequency count is also updated. In situations where no improving moves exist, all solutions are penalized according to their frequency and the move with the best penalty-adjusted time is accepted. The penalty function multiplies the grading time by a factor of 0.5, and this is added to the total solution time.

SPSEI Heuristic

This heuristic seeks to rearrange the order of service edge traversal in order to improve overall tour efficiency. When evaluating candidate solutions, the heuristic investigates the shortest paths that would be inserted into the tour as a result of the swap under consideration. Specifically the routine is searching for instances where these shortest paths create redundancies in service edge traversal. If such an instance is found, the solution is re-ordered so that any service edge located along a shortest path between two other service edges is graded along the way.

Consider the solution as depicted in **Figure 2d**, and additionally assume the edge connecting vertices B and D also requires service. When traversing between service edges [A, B] and [D, C], the grader will take the shortest path between vertices B and D, which is the edge (B, D). Clearly it makes no sense to traverse this edge (B, D), grade [D, C] and then return later to grade [B, D]. A more economical option would be to grade the service edge [B, D] en route to [D, C], which would result in the candidate solution [A, B] $\rightarrow$ [B, D] $\rightarrow$ [D, C].

Tour Partition Heuristic

If a tour's total time exceeds the allowable limit, it must be partitioned into some subset of feasible sub-tours. This heuristic attempts to partition the tour into sub-tours in such a way as to minimize total operating time. A simple rule is to continue grading for as long as possible and stop with enough time to return to the depot. Cases could exist where it would be more efficient to return early. For example consider the case where elapsed time is near the limit, and the remaining service edges are clustered in some area far away from the depot. Rather than travel the far distance with only enough time to service one or two edges, it might be more economical to return to the depot early and finish all of the remaining service edges in the next day's sub-tour. This heuristic considers multiple partition locations to seek the best location to partition the overall tour into sub-tours. The tour is initially partitioned at the last possible location, and then a neighborhood around that initial partition is evaluated by reducing the current sub-tour by one service edge at a time.

Computational Results

All procedures were programmed in C and compiled in Microsoft Visual C++ 6.0. A variety of road networks were tested, including networks with cycle and directed networks that are more commonly found in steep terrain. Initially, the tabu search heuristic model was tested on four artificially generated networks, purposefully created small enough so the optimum solutions could be found through complete enumeration. Then the model was applied to a real data set from a research forest owned by Oregon State University (OSU). The following parameters are used to generically define an instance of a road network: the number of vertices (|V|) and the number of edges (|E|). Density, d, is computed as $|E| / |V|^2$. Typically forest road networks have a very sparse density, and the generated data sets reflect that. Additionally, |SE| represents the number of edges requiring service in a particular network.

Each generated network was tested with nine problem instances, where a problem instance consists of a |SE| value and a randomly generated combination of that many service edges. A total of 36 instances were, therefore, solved on artificial networks, and for each instance the heuristic algorithm generated 100 results. For the Oregon State Research Forest, a total of six problem instances were tested, and for each instance the heuristic algorithm generated 50 results.

For the generated networks, a range of |SE| values were tested, and three random combinations of service edges were generated for each value of |SE|. Appendix A contains a listing of the service edges associated with each random instance. On the small networks, the model was run 100 times for each unique |V|, |E|, and |SE| combination. For the Oregon State Research Forest, fewer problems were tested but included a range of possible |SE| values; the model ran 50 times for each |SE| value.

Initially the tests were run absent the maximum operating time constraint in order to evaluate the baseline performance of the model. After it was determined the model could provide satisfactory results, the time constraint was incorporated, but only for one |SE| value per network, due in part to the increased computational difficulty of determining the optimal solution when incorporating such a constraint. The values used for "Max Time" were chosen based upon the obtained unconstrained solutions, and should not be construed to have any real world meaning. Because the optimal solution was

found through enumeration, for Networks 1 through 4 the values in the "Num Best" column represent the number of times the model determined the actual optimal solution. A value of 75, for example, in this column means that the optimum solution was reached by the tabu search heuristic in 75 out of 100 runs. This will not be the case for the Oregon State research forest network, for which the optimal solution is unknown. The "Avg Dev" column represents the average percent deviation from the best solution found.

The results obtained and reported in this paper were performed on a machine equipped with Optiplex GX280 3.3 GHz with 1 GB RAM. CPU time was not recorded, although observationally solutions were generated in less than a second for all but the real forest network. CPU times for that network ranged from seconds to over 2 minutes, depending on the value of |SE|.

Network 1: |V| = 10, |E| = 10, d = 10%

Our model was tested on Network 1, which was dendritic and contained one directed edge and five spurs. A tabu list size of 15 was used for this network. Nine problem instances were tested, three of which included operational time constraints. The heuristic obtained the known optimal solution 100 percent of the time.

Network 2: |V| = 10, |E| = 20, d = 20%

Network 2 is an extension of Network 1. Ten edges were added, two of which were directed. Network 2 was cyclical, and thus contained no spurs. A tabu list size of 15 was used for this network. **Table 3** presents the results for Network 2. Instances 2.7, 2.8, and 2.9 are comprised of the same service edges as instances 2.1, 2.2, and 2.3, respectively, but with the inclusion of operational time constraints.

Network 3: |V| = 20, |E| = 20, d = 5%

Network 3 is shown in **Figure 3**. Two edges are one-way: (15, 16) and (16, 10). Twelve spurs were contained in Network 3. A tabu list size of 15 was used for this network. **Table 4** presents the results for Network 3. Instances 3.7, 3.8, and 3.9 are comprised of the same service edges as instances 3.1, 3.2, and 3.3, respectively, but with the inclusion of operational time constraints.

Network 4: |V| = 20, |E| = 40, d = 10%

Network 4 is shown in **Figure 4**. Twenty edges were added to Network 3 to create this network, including four one-way edges: (4, 7), (7, 8), (8, 13), and (18, 16). This network, like Network 2, contains no spurs. A tabu list size of 20 was used. **Table 5** presents the results for Network 4. Instances 4.7, 4.8,

Table 3. ~ *Results for Network 2.*

Instance	|SE|	Max time	Num Best	Avg Dev (%)
2.1	4	--	100	0.00
2.2	4	--	100	0.00
2.3	4	--	98	0.06
2.4	8	--	100	0.00
2.5	8	--	99	0.09
2.6	8	--	79	0.35
2.7	4	1.5	100	0.00
2.8	4	1.5	100	0.00
2.9	4	1.5	38	3.44

Table 4. ~ *Results for Network 3.*

Instance	|SE|	Max time	Num Best	Avg Dev (%)
3.1	6	--	100	0.00
3.2	6	--	87	0.67
3.3	6	--	78	1.12
3.4	8	--	79	0.97
3.5	8	--	72	1.17
3.6	8	--	44	2.40
3.7	6	2	100	0.00
3.8	6	2	90	1.11
3.9	6	2	68	1.48

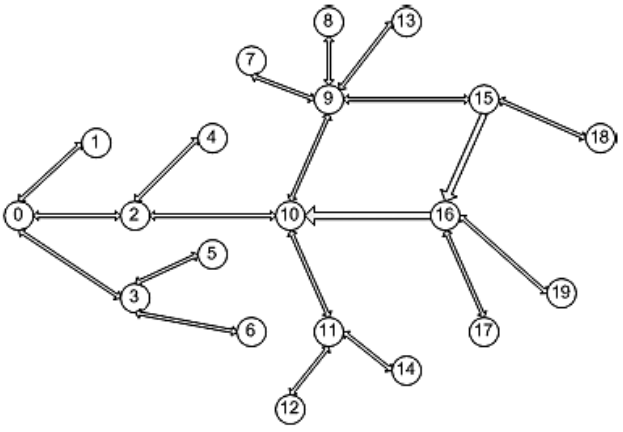

Figure 3. ~ *Network 3.*

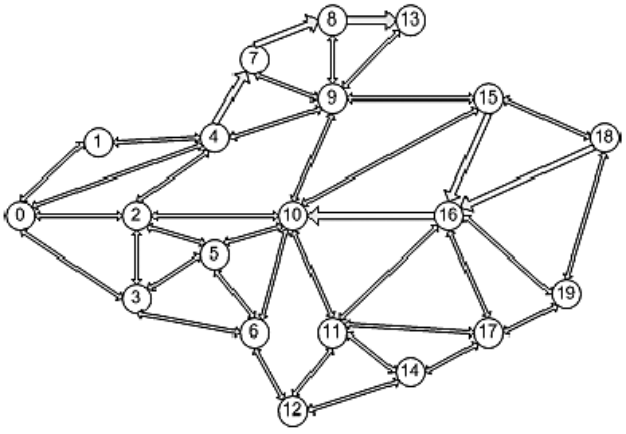

Figure 4. ~ *Network 4.*

Table 5. ~ *Results for Network 4.*

Instance	|SE|	Max time	Num Best	Avg Dev
				(%)
4.1	8	--	96	0.17
4.2	8	--	88	0.29
4.3	8	--	55	3.84
4.4	9	--	100	0.00
4.5	9	--	81	0.71
4.6	9	--	51	0.74
4.7	8	2	93	0.46
4.8	8	2	84	0.36
4.9	8	2	64	1.62

and 4.9 are comprised of the same service edges as instances 4.1, 4.2, and 4.3, respectively, but with the inclusion of operational time constraints.

Oregon State Research Forest: |V| = 127, |E| = 141, d = 0.9%

Road segment data for this network was obtained from GIS data published by OSU. Specifically, a portion of the roads in the northwest corner of the Dunn Forest was selected. Of the 141 edges, 123 were undirected and 18 were directed. The network contained a total of 38 spurs.

Data on segment roughness and maintenance need were unavailable and were, therefore, artificially generated. Arbitrary roughness index levels (1 through 5, 5 = highest level of roughness) and road categories (1 through 3, 3 = least trafficked road) were randomly assigned to various road segments. Road segment classifications were done according to assigned probabilities that mimic the distribution of roads on the network in the OSU forest, whereas roughness levels were assigned in such a way as to have approximately 10 percent of the segments requiring service. As a proxy for managerial decisions, a framework was established in which road segments of a certain category require grading if the roughness level is above a threshold. The following rubric was imposed to create a quantitative framework for determining when to grade a road segment.

Grading rules

Category 1: Grade if RI > 2

Category 2: Grade if RI > 3

Category 3: Grade if RI > 4

A tabu list size of 25 was used. Note that only 50 runs were performed over this larger network. **Table 6** presents the results for the research forest network.

Table 6. ~ *Results for Oregon State Research Forest Network.*

Instance	|SE|	Max time	Num Best	Avg Dev
				(%)
RF.1	14	--	19	1.45
RF.2	15	--	4	2.28
RF.3	15	--	1	3.71
RF.4	14	3	10	2.63
RF.5	13	3	8	2.29
RF.6	15	3	2	5.18

Discussion

For the smaller networks, the heuristic performs quite well, even when operational time constraints are included. In summary, aggregate over the artificial networks, the optimal solution was reached in 2,107 of 2,400 runs (87.79%) for problem instances with no operating constraints, and in 1,037 of 1,200 runs (86.42%) for instances with operational time constraints. Aggregate percent deviation was 0.52 percent and 0.71 percent, for the constraint-free scenario and operating time limit scenario, respectively.

In instances where the optimal solution was achieved a relatively small proportion of the time, the heuristic selected the known second or third best solution a high proportion of the time. Instance 3.6 of Network 3 (**Table 4**), for example, only locates the optimal solution 44 times. But, the tabu search procedure located the known second-best solution an additional 38 times, meaning that at least 82 percent of the obtained solutions are of extremely high quality.

For the much larger forest road network, the low average deviation suggests that the heuristic consistently obtains good results, although there is no optimal solution as a basis for comparison. One possible method to estimate the quality of our solutions would be to compare against solutions from relaxations of integer linear programs, although Eiselt et al. (1995b) state that in practice this method has not been very successful for ARPs. Alternatively a global optimum may be estimated using extreme value theory, although Boston and Bettinger (1999) demonstrated that estimates of optima were unreliable, and Bettinger et al. (2002) found the technique of limited usefulness.

Since it is difficult to compare our solution against a known or estimated optimal value, it might be informative to learn if the heuristic would yield improvements over current practices. To validate our model, a greedy heuristic was developed, which might mimic how a grader operator would traverse the network to grade the segments. Starting from the depot, the grader first travels to the closest segment requiring grading and services that segment. From there, the grader travels to the next nearest segment requiring grading, and so on until all segments requiring grading have been serviced. The greedy heuristic was tested on RF.1. The tabu search heuristic achieved a 12.5 percent reduction in total grading time as compared to the greedy heuristic. Given that in practice an operator may not know with certainty which is the nearest segment requiring service, it is possible savings could be even higher. Further, given that in practice many grading decisions

are assigned not based on road condition but on time since the last grading, the overall decision framework presented here may yield substantial reductions in overall maintenance cost.

As road density increases, so does the complexity of the problem. But, it is important to note that in forest road networks density is usually very small, and thus this solution framework may prove quite acceptable. Additionally, maintenance needs rarely constitute a large proportion of the road network, suggesting the instances generated above may be quite representative of the real-world problems faced by industry. Certainly the results demonstrate the gains possible from using computational methods to evaluate complex decisions, especially in light of the current management practices, which cannot or do not account for the combinatorial nature of the networks.

Solution quality appears to be associated with two factors: number of directed service edges and, to a lesser degree, number of service edges that are spurs. On average, as the number of directed service edges or spur service edges increases, solution quality decreases. Additional work with this data set could include testing a variety of service edge requirements to determine what impact, if any, the number of directed and spur service edges have upon solution quality for larger networks.

Future research should focus on both the solution methodology and the problem definition. Improvements might be achievable by increasing the complexity of the tabu search heuristic, such as utilizing a dynamic tabu list size or implementing strategic oscillation. The tour partition heuristic could also be expanded and possibly implemented as a combinatorial heuristic, such as tabu search or simulated annealing. This method would be of great benefit in cases where the initial tour's length is so great as to require multiple partitions; the world of possible partitions increases exponentially with respect to a unit increase in the number of partitions required. Incorporation of time-windows would create a more realistic model, applicable perhaps to a high volume network experiencing simultaneous haul and maintenance traffic. Some industrial applications might involve a fleet of graders, operating from different depots or perhaps without an assigned depot. Operational time constraints may not be quite so limiting, allowing for a small exceedance in order to gain overall tour efficiency.

Future work could also be devoted to creating a broader decision framework for determining which segments to grade. Rather than utilize a binary decision rule, priorities could be established for segments requiring service. Thus the heuristic would ensure that some segments of the highest priority are graded, but in addition would seek opportunities to grade other high priority segments. If successfully implemented, this framework could free up future resources for other maintenance activities, hopefully preventing future drainage and transportation problems. Perhaps this method would best be employed in scenarios where the manager's goal is to achieve the highest level of maintenance possible, rather than a goal of cost minimization.

It is the intention of this paper to demonstrate the potential for cost savings in grader scheduling and deployment. The heuristic in this study performed well on both cyclical and dendritic networks, the latter being more common in mountainous terrain. The tabu search heuristic presented can provide for improved decision making with regard to routing graders and could improve overall road maintenance.

Acknowledgments

The authors would like to thank Dave Young, the Roads and Trails Specialist of the Oregon State University College Forests Office, for his input regarding operational road maintenance practice and grading in particular. Jeff Hamann, a graduate student in the OSU Forest Engineering Department, provided invaluable assistance in computer programming. We would also like to thank the editor and anonymous reviewers for their helpful comments.

Literature Cited

Amberg, A., W. Domschke, and S. Voß. 2000. Multiple center capacitated arc routing problems: A tabu search algorithm using capacitated trees. European J. of Operational Research. 124(2): 360-376.

Aruga, K. 2005. Tabu search optimization of horizontal and vertical alignments of forest roads. J. of Forest Research. 10(4): 275-284.

Bettinger, P., D. Graetz, K. Boston, J. Sessions, and W. Chung. 2002. Eight heuristic planning techniques applied to three increasingly difficult wildlife-planning problems. Silva Fennica. 36(2): 561-584.

_____, K. Boston, and J. Sessions. 1999. Intensifying a heuristic forest harvest scheduling procedure with paired attribute decision choices. Canadian J. of Forest Research. 29: 1784-1792.

_____, J. Sessions, and K.N. Johnson. 1998. Ensuring the compatibility of aquatic habitat and commodity production goals in Eastern Oregon with a tabu search procedure. Forest Sci. 44(1): 96-112.

_____, _____, and K. Boston. 1997. Using tabu search to schedule timber harvests subject to spatial wildlife goals for big game. Ecological Modelling. 94: 111-123.

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

_____ and _____. 1999. An analysis of Monte Carlo integer programming, simulated annealing, and tabu search heuristics for solving spatial harvest scheduling problems. Forest Sci. 45(2): 292-301.

Caterpillar, Inc. 2002. Caterpillar Performance Handbook. Version 33. Caterpillar, Inc., Peoria, IL.

Corberán, A., R. Martí, and A. Romero. 2000. Heuristics for the mixed rural postman problem. Computers and Operations Research. 27(2): 183-203.

Eiselt, H.A., M. Gendreau, and G. Laporte. 1995a. Arc routing problems, Part I: The Chinese postman problem. Operations Research. 43(2): 231-242.

_____, _____, and _____. 1995b. Arc routing problems, Part II: The rural postman problem. Operations Research. 43(3): 399-414.

Forest Engineering Research Institute of Canada (FERIC). 2007. Opti-Grade. www.feric.ca/index.cfm?objectid=2DBAACB8-E081-222F-A48CDBA6DAA20324. Last accessed 11 April 2007.

Gendreau, M., A. Hertz, and G. Laporte. 1994. A tabu search heuristic for the vehicle routing problem. Management Sci. 40(10): 1276-1290.

Glover, F. 1990. Tabu search – Part II. ORSA J. of Computing. 2(1): 4-32.

_____. 1989. Tabu search – Part I. ORSA J. of Computing. 1(3): 190-206.

Hertz, A., G. Laporte, and M. Mittaz. 2000. A tabu search heuristic for the capacitated arc routing problem. Operations Research. 48(1): 129-135.

Luce, C.H. and T.A. Black. 1999. Sediment production from forest roads in western Oregon. Water Resources Research. 35(8): 2561-2570.

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

Provencher, Y. 1995. Optimizing road maintenance intervals. *In*: Proc. of the Sixth International Conf. on Low-Volume Roads, June 25-29, Minneapolis, MN. Vol. I, pp. 199-207.

Richards, E.W. and E.A. Gunn. 2003. Tabu search design for difficult forest management optimization problems. Canadian J. of Forest Research. 33(6): 1126-1133.

APPENDIX A:
Listing of Randomly Generated Service Edges

Network 1:

1.1: [1,3], [4,7]

1.2: [2,5], [5,6]

1.3: [0,2], [4,2]

1.4: [4,2], [4,7], [4,8], [5,9]

1.5: [0,1], [1,3], [4,2], [5,6]

1.6: [0,2], [1,3], [1,4], [4,7]

Network 2:

2.1: [0,1], [2,5], [5,9], [4,5]

2.2: [1,4], [5,6], [4,5], [8,9]

2.3: [4,2], [4,7], [3,4], [7,9]

2.4: [0,2], [1,4], [4,2], [4,7], [0,4], [3,4], [7,9], [8,9]

2.5: [0,1], [1,3], [4,2], [4,8], [0,4], [2,6], [3,4], [6,9]

2.6: [1,4], [2,5], [4,8], [5,9], [3,4], [4,5], [6,9], [8,5]

Network 3:

3.1: [0,2], [0,3], [3,6], [7,9], [10,11], [11,12]

3.2: [2,4], [7,9], [9,10], [10,11], [15,16], [16,17]

3.3: [2,10], [8,9], [9,13], [11,14], [15,16], [16,10]

3.4: [0,2], [2,4], [2,10], [8,9], [11,12], [11,14], [15,18], [16,10]

3.5: [0,1], [0,3], [3,5], [7,9], [9,13], [11,12], [15,16], [16,17]

3.6: [2,4], [3,5], [8,9], [9,15], [10,11], [11,14], [15,18], [16,19]

Network 4:

4.1: [0,1], [9,10], [9,13], [9,15], [10,11], [11,17], [12,14], [14,17]

4.2: [1,4], [2,5], [3,5], [3,6], [5,10], [6,12], [7,8], [16,17]

4.3: [0,3], [0,4], [4,7], [7,8], [9,13], [11,12], [15,18], [16,17]

4.4: [0,1], [0,2], [0,3], [0,4], [1,4], [2,3], [2,4], [2,5], [2,10]

4.5: [0,4], [5,6], [6,12], [8,9], [8,13], [10,15], [12,14], [16,19], [18,19]

4.6: [2,4], [5,10], [6,10], [7,9], [8,13], [9,13], [10,15], [11,16], [16,10]