Forest Road Construction: heuristic and interactive approach

David Meignan, Postdoctoral fellow
Jean-Marc Frayret, Ing. Jr., PhD., supervisor
Gilles Pesant, Ph.D., co-supervisor
Presentation Outline

- Mixed Integer Model
- Base Algorithms
  - Greedy construction
  - Local search
- Interactive Optimization
  - GRASP
  - User preferences
  - Partial re-optimization
- Future Work
Mixed Integer Model

Modeling elements

\[ y_{ij} = 1 \]

\[ x_{ik} = 1 \]

\[ y_{ij} = 1 \]

\[ x_{ik} = 1 \]
**Mixed Integer Model**

**Sets and parameters:**
- \(G(V, E, A):\) bi-level graph
- \(V:\) set of vertices
- \(E:\) set of potential route edges
- \(A:\) set of service (harvesting) edges
- \(R:\) set of root vertices (existing roads or exit locations)
- \(B:\) set of demands (cell to harvest)
- \(w_i:\) quantity of demand (timber volume) at vertex \(i\)
- \(c_{ij}:\) building cost along the edge \((i, j)\) for the set of potential route edges \(E\)
- \(d_{ij}:\) unit cost to serve (harvest) the demand point \(j\) from vertex \(i\)

**Variables:**
- \(x_{ij} = \begin{cases} 1 & \text{if edge } (i, j) \text{ is part of the solution} \\ 0 & \text{otherwise} \end{cases}\)
- \(y_{ij} = \begin{cases} 1 & \text{if vertex } i \text{ serves the demand point } j \\ 0 & \text{otherwise} \end{cases}\)
- \(z_{ij}:\) flow variable, total demand that pass through the route edge \((i, j)\)

Minimize \(\sum_{(i,j)\in E} c_{ij} x_{ij} + \sum_{(i,j)\in A} d_{ij} y_{ij} w_j\)

Subject to:
1. \(\sum_{i\in V} z_{ij} = \sum_{k\in V} z_{jk} + \sum_{k\in V,(j,k)\in A} y_{jk} w_k \quad \forall j \in V \setminus R\)
2. \(\sum_{i\in R} \sum_{j\in V \setminus R} z_{ij} + \sum_{i\in R} \sum_{j\in V \setminus R,(i,j)\in A} y_{ij} w_j \geq T\)
3. \(r_j + \sum_{i\in V,(i,j)\in E} x_{ij} \geq y_{jk} \quad \forall (j, k) \in A\)
4. \(\sum_{i\in V,(i,j)\in A} y_{ij} \geq b_j \quad \forall j \in V\)
5. \(y_{ij} d_{ij} \leq D \quad \forall (i, j) \in A\)
6. \(z_{ij} \leq x_{ij} T \quad \forall (i, j) \in E\)
7. \(x_{ij} \in \{0,1\} \quad \forall (i, j) \in E\)
8. \(y_{ij} \in \{0,1\} \quad \forall (i, j) \in A\)
9. \(z_{ij} \in \{0, \ldots, T\} \quad \forall (i, j) \in E\)
Base algorithm > Greedy construction

Principle

Select a set of covering vertices
(the set is iteratively constructed)

Step 1: Location of covering vertices

Procedure Location of covering vertices

1. coveringVertices ← ø
2. toCover ← demands – VERTICESCOVERED(rootVertices)
3. while toCover ≠ ø do
4. vertex ← MAXIMUMCOVERAGE(toCover)
5. coveringVertices ← coveringVertices U {vertex}
6. toCover ← toCover – VERTICESCOVERED(vertex)
Base algorithm > Greedy construction

- **Principle**
  - Select a set of covering vertices
    (the set is iteratively constructed)
Base algorithm

Base algorithm > Greedy construction

Principle

Select a set of covering vertices
(the set is iteratively constructed)

Generate of a set of trees that span the covering vertices
(minimum path algorithm)

Step 2: Spanning covering vertices

Procedure Span covering vertices

1. terminals ← coveringVertices
2. forest ← φ
3. forestVertices ← rootVertices
4. while terminals ≠ φ do
   5. path ← MINIMUMPATH(forestVertices, terminals)
   6. forest ← forest U path
   7. forestVertices ← forestVertices U VERTICES(path)
   8. terminals ← terminals – VERTICES(path)
Base algorithm > Greedy construction

- **Principle**
  - Select a set of covering vertices
    (the set is iteratively constructed)
  - Generate a set of trees that span the covering vertices
    (minimum path algorithm)

Step 2: Spanning covering vertices
Base algorithm

Base algorithm > Local search

- **Variable neighborhood descent**
  - Three neighborhood structures are used
  - The solution is improved until a local optimum with regard to the three neighborhood is reached

---

**Procedure** Variable neighborhood descent

1. $s \leftarrow $ INITIALSOLUTION()
2. $k \leftarrow 1$
3. while $k \neq k_{\text{max}}$ do
4. \hspace{1em} $s' \leftarrow $ FINDBEST(N_k, s)
5. \hspace{1em} if $f(s) > f(s')$ then
6. \hspace{2em} $s \leftarrow s'$
7. \hspace{1em} $k \leftarrow 1$
8. \hspace{1em} else
9. \hspace{2em} $k \leftarrow k + 1$

---

Movement in neighborhood structures

insertion move

removal move

swap move
Interactive optimization

The decision support system offers different ways to run and interact with the optimization procedure:

- **Fully automated optimization:**
  - The user indicates a time limit or a maximum number of iterations to run a GRASP (Greedy Randomized Adaptive Search Procedure).

- **Iterative addition of preferences:**
  - An initial (greedy and deterministic) solution is generated.
  - The user iteratively adds some preferences (mandatory/penalized points). The greedy procedure is run to take into account preferences.

- **Partial re-optimization with preferences:**
  - An initial (greedy and deterministic) solution is generated.
  - The user possibly adds some preferences, and selects the set of road sections to re-optimize. An optimization procedure is run only on the selected sections.
Fully automated optimization > GRASP

- **GRASP Principle:**
  - Iterative procedure with two phases
  1. Greedy randomized construction procedure (Insert elements in a partial solution until a complete solution is obtained. The choice of elements combines a greedy function and a part of randomness)
  2. Local search (the solution is improved until a local optimum)

- **GRASP implementation for our problem:**
  1. Greedy randomized selection of covering vertices
  2. Computation of a set of trees that span the covering vertices
  3. Variable neighborhood descent using the three neighborhood structures
Interactive optimization

Iterative addition of preferences > Greedy construction + local descent

- **User preferences**
  - Mandatory points
  - Penalized points

- **Greedy construction with preferences**
  - Pure greedy construction procedure (selection of covering vertices, then spanning)
  - Mandatory points are initially added to the set of covering vertices
  - User penalties are taken into account for the computation of spanning trees

- **Local descent with preferences**
  - Variable neighborhood descent using three neighborhood structures
  - Mandatory points are not involved in local search
  - User penalties are taken into account for costs computation
Interactive optimization

Partial re-optimization with preferences > Reconstruction + local descent

- **User preferences**
  - Selected sections
  - Mandatory points
  - Penalized points

- **Reconstruction with preferences**
  1. Sections selected are removed from the solution
  2. The reconstruction procedure reconnects the partial solution while taking into account mandatory point and user penalties

- **Partial local descent with preferences**
  - Local descent is applied only on new parts of the solution
  - User preferences are taking into account
Interactive optimization

Future Work

- **Take into account flows on edges**
  - Solutions may lead to longer hauling distances
- **Complete connectivity with FPInterface**
  - Enable knowledge transfer
  - Enable on-site performance evaluation
- **Improve user interface**
  - Improve user experience and man-machine interaction
  - Enable the development of an intelligent interface agent
Interactive optimization

Thank you…