The dial a ride problem (DARP)
The dial a ride problem (DARP)

Work in progress! Most of the material presented in this talk is from:


Some material is new and is based on joint work with Jean-François Cordeau and Gilbert Laporte (Canada Research Chair in Distribution Management, HEC Montréal).

**About myself**

- Stefan Røpke (sropke@diku.dk)
- PhD student at DIKU
- Interested in solving routing problems using metaheuristics and exact optimization methods.
The dial a ride problem

The problem

- Door-to-door transportation of elderly and disabled persons (the users).

- Several users are transported in the same vehicle (think of a mini-bus).

- The users specify when they wish to be picked up and when they have to be at their destination. Such a transportation task is denoted a request.

- The users do not specify an exact time of day, but a time window. Example: Instead of requesting a pickup at 9:11 the users request a pickup between 9:00 and 9:30.

- Often the user only specify either the pickup or delivery time window. An operator would assign the other time window.
The dial a ride problem

- Users don’t like to be taken on long detours even if it helps the overall performance of the transportation system. Consequently a maximum ride time constraint is specified for each request.

- Time windows are not enough for ensuring that the maximum ride time constraint is enforced. Example: pickup [8:00; 8:15], delivery [8:45; 9:00], max ride time 45 minutes. Pickup at 8:00 and delivery at 9:00 violates max ride time constraint. Pickup time window could be shrunk to [8:15; 8:15]. This would ensure that ride time constraint is enforced, but it rules out perfectly good solutions like pickup at 8:05 and delivery at 8:45.

- Each vehicle has a certain capacity (only a limited amount of seats).

- The vehicles have to start and end their tours at a given start and end terminal.

- Objective: minimize driving cost subject to the constraints mentioned above.

- Problem is NP-Hard.
DARP example

Possible solution:
Branch and Bound

• Minimization problem. Main ingredients: lower and upper bound.

• High level algorithm:

1. Set of subproblems ($SoS$) = \{ Entire problem \}
2. Remove subproblem $S$ from $SoS$
3. Find lower and upper bound ($LB$ and $UB$) for $S$
4. if $UB < \text{global UB (GUB)}$ then $\text{GUB} = UB$
5. if $LB < \text{GUB}$ then split $S$ into two subproblems and add them to $SoS$
6. if $SoS \neq \emptyset$ then goto step 2, else return $GUB$
DARP formal definition (Graph problem)

Notation:

- \( n \) Number of requests.
- \( P = \{1, \ldots, n\} \) Pickup locations
- \( D = \{n + 1, \ldots, 2n\} \) Delivery locations
- \( N = P \cup D \cup \{0, 2n + 1\} \) The set of all nodes in the graph. 0 and \( 2n + 1 \) are the start and end terminal respectively. Request \( i \) consist of pickup \( i \) and delivery \( n + i \).
- \( K \) Set of vehicles
- \( G = (N, A) \) Directed graph on which the problem is defined. \( A \) is the set of edges.
- \( Q \) Capacity of a vehicle
- \( q_i \) Amount loaded onto vehicle at node \( i \). \( q_i = q_{n+i} \).
- \( [e_i, l_i] \) time window of node \( i \)
- \( d_i > 0 \) duration of service at node \( i \)
- \( L \) Max ride time of a request.
- \( c_{ij} \) Cost of traveling from node \( i \) to node \( j \). It is assumed that \( c_{ij} \) satisfies the triangle inequality.
- \( t_{ij} \) Time needed for going from node \( i \) to node \( j \). It is assumed that \( t_{ij} \) satisfies the triangle inequality.
Standard model (DARP1)

Decision variables

Binary variables
\[ x_{ij}^k \quad 1 \text{ iff the } k\text{th vehicle goes straight from node } i \text{ to node } j. \]

Fractional variables
\[ B_i^k \quad \text{When vehicle } k \text{ starts visiting node } i \]
\[ Q_i^k \quad \text{The load of vehicle } k \text{ after visiting node } i. \]
\[ L_i^k \quad \text{The ride time of request } i \text{ on vehicle } k. \]
Standard model (DARP1)

Objective:

\[ \min \sum_{k \in K} \sum_{i \in N} \sum_{j \in N} c_{ij} x_{ij}^k \]

Every request is served exactly once:

\[ \sum_{k \in K} \sum_{j \in N} x_{ij}^k = 1 \quad \forall i \in P \]

Same vehicle services pickup and delivery:

\[ \sum_{j \in N} x_{ij}^k - \sum_{j \in N} x_{n+i,j}^k = 0 \quad \forall i \in P, k \in K \]

Every vehicle leaves the start terminal:

\[ \sum_{j \in N} x_{0j}^k = 1 \quad \forall k \in K \]

The same vehicle that enters a node leaves the node:

\[ \sum_{j \in N} x_{ji}^k - \sum_{j \in N} x_{ij}^k = 0 \quad \forall i \in P \cup D, k \in K \]

Every vehicle enters the end terminal:

\[ \sum_{i \in N} x_{i,2n+1}^k = 1 \quad \forall k \in K \]
Standard model (DARP1)

Setting and checking visit time:

\[ B^k_j \geq (B^k_i + d_i + t_{ij})x^k_{ij} \quad \forall i \in N, j \in N, k \in K \]
\[ e_i \leq B^k_i \leq l_i \quad \forall i \in N, k \in K \]

Linearization of first equation (\( M^k_{ij} \) is a large constant):

\[ B^k_j \geq B^k_i + d_i + t_{ij} - M^k_{ij}(1 - x^k_{ij}) \quad \forall i \in N, j \in N, k \in K \]

Setting and checking ride time:

\[ L^k_i = B^k_{n+i} - (B^k_i + d_i) \quad \forall i \in P, k \in K \]
\[ L^k_i \leq L \quad \forall i \in N, k \in K \]

Setting and checking vehicle load:

\[ Q^k_j \geq (Q^k_i + q_j)x^k_{ij} \quad \forall i \in N, j \in N, k \in K \]
\[ Q^k_i \leq Q \quad \forall i \in N, k \in K \]

Binary variables:

\[ x^k_{ij} \in \{0, 1\} \quad \forall i \in N, j \in N, k \in K \]
Preprocessing

- Shrink time windows. For example if $l_i = 10, l_{n+i} = 20, d_i = 2$ and $t_{i,n+i} = 12$ then $l_i$ can be reduced to 6.

- Remove edges from $G$ that cannot be part of a feasible solution.

- Some examples
  - Edges that are impossible because of time windows
  - Edges of the type $(n + i, i) \forall i \in P$
  - Edges of the type $(0, n+i) \forall i \in P$ and $(i, 2n+1) \forall i \in P$
  - Edges that are impossible because of ride time constraints. Edge $(i, j)$ can be removed if $j \neq n + i$ and the trip $i \rightarrow j \rightarrow n + i$ violates the ride time constraint of request $i$

- Preprocessing is fast and easy to do, but can have a significant impact on the running time of the algorithm.
Some results

- Solving (DARP1) using CPLEX 8.0 on 2.5 Ghz Pentium 4

<table>
<thead>
<tr>
<th>Instance</th>
<th>Bound</th>
<th>CPU (min)</th>
<th>Nodes</th>
<th>Opt</th>
</tr>
</thead>
<tbody>
<tr>
<td>a2-16</td>
<td>294.25</td>
<td>0.01</td>
<td>23</td>
<td>294.25</td>
</tr>
<tr>
<td>a2-20</td>
<td>344.83</td>
<td>0.05</td>
<td>313</td>
<td>344.83</td>
</tr>
<tr>
<td>a2-24</td>
<td>431.12</td>
<td>1.42</td>
<td>10,868</td>
<td>431.12</td>
</tr>
<tr>
<td>a3-18</td>
<td>300.48</td>
<td>0.41</td>
<td>3,596</td>
<td>300.48</td>
</tr>
<tr>
<td>a3-24</td>
<td>344.83</td>
<td>76.59</td>
<td>310,667</td>
<td>344.83</td>
</tr>
<tr>
<td>a3-30</td>
<td>472.17</td>
<td>240.00</td>
<td>515,931</td>
<td>494.85</td>
</tr>
<tr>
<td>a3-36</td>
<td>570.26</td>
<td>240.00</td>
<td>504,553</td>
<td>583.19</td>
</tr>
<tr>
<td>a4-16</td>
<td>282.68</td>
<td>21.49</td>
<td>145,680</td>
<td>282.68</td>
</tr>
<tr>
<td>a4-24</td>
<td>359.52</td>
<td>240.00</td>
<td>442,000</td>
<td>375.02</td>
</tr>
<tr>
<td>a4-32</td>
<td>427.65</td>
<td>240.00</td>
<td>189,900</td>
<td>485.50</td>
</tr>
<tr>
<td>a4-40</td>
<td>462.21</td>
<td>240.00</td>
<td>65,000</td>
<td>557.69</td>
</tr>
<tr>
<td>a4-48</td>
<td>466.7</td>
<td>240.00</td>
<td>40,400</td>
<td>668.82</td>
</tr>
<tr>
<td>b2-16</td>
<td>309.41</td>
<td>0.21</td>
<td>5,815</td>
<td>309.41</td>
</tr>
<tr>
<td>b2-20</td>
<td>332.64</td>
<td>0.01</td>
<td>26</td>
<td>332.64</td>
</tr>
<tr>
<td>b2-24</td>
<td>444.71</td>
<td>2.76</td>
<td>32,399</td>
<td>444.71</td>
</tr>
<tr>
<td>b3-18</td>
<td>301.64</td>
<td>1.29</td>
<td>12,223</td>
<td>301.65</td>
</tr>
<tr>
<td>b3-24</td>
<td>394.51</td>
<td>7.27</td>
<td>42,950</td>
<td>394.51</td>
</tr>
<tr>
<td>b3-30</td>
<td>531.44</td>
<td>189.74</td>
<td>574,281</td>
<td>531.45</td>
</tr>
<tr>
<td>b3-36</td>
<td>588.44</td>
<td>240.00</td>
<td>447,474</td>
<td>603.79</td>
</tr>
<tr>
<td>b4-16</td>
<td>296.96</td>
<td>2.44</td>
<td>20,189</td>
<td>296.96</td>
</tr>
<tr>
<td>b4-24</td>
<td>369.36</td>
<td>59.31</td>
<td>175,495</td>
<td>369.36</td>
</tr>
<tr>
<td>b4-32</td>
<td>460.78</td>
<td>240.00</td>
<td>222,600</td>
<td>494.82</td>
</tr>
<tr>
<td>b4-40</td>
<td>570.37</td>
<td>240.00</td>
<td>153,400</td>
<td>656.63</td>
</tr>
<tr>
<td>b4-48</td>
<td>577.64</td>
<td>240.00</td>
<td>50,000</td>
<td>673.81</td>
</tr>
</tbody>
</table>
Preprocessing pays off

- Some examples (Cplex 9.0 on 3.0Ghz Pentium 4).

<table>
<thead>
<tr>
<th></th>
<th>Preprocessing</th>
<th>No preprocessing</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CPU (min)</td>
<td>Nodes</td>
</tr>
<tr>
<td>a2-20</td>
<td>0.07</td>
<td>332</td>
</tr>
<tr>
<td>b3-24</td>
<td>3.91</td>
<td>16773</td>
</tr>
</tbody>
</table>

- (DARP1) had $O(|N|^2|K|)$ binary variables. If we could get rid of the $k$ index on the $x^{k}_{ij}$ variables then the number of binary variables could be reduced to $O(|N|^2)$, which hopefully would make the problem easier to solve.

- (DARP2) - model where the $k$ index is stripped from all variables. The variables have the same meaning as in (DARP1), they are just no longer associated with a specific vehicle.
Compact model (DARP2)

Objective:

\[
\min \sum_{i \in N} \sum_{j \in N} c_{ij} x_{ij}
\]

One vehicle enters every user node and one vehicle leaves every user node:

\[
\sum_{j \in N} x_{ij} = 1 \quad \forall i \in P
\]

\[
\sum_{i \in N} x_{ij} = 1 \quad \forall j \in P
\]

Setting and checking visit time:

\[
B_j \geq (B_i + d_i + t_{ij}) x_{ij} \quad \forall i \in N, j \in N
\]

\[
e_i \leq B_i \leq l_i \quad \forall i \in N
\]

Setting and checking ride time:

\[
L_i = B_{n+i} - (B_i + d_i) \quad \forall i \in P
\]

\[
L_i \leq L \quad \forall i \in N
\]

Setting and checking vehicle load:

\[
Q_j \geq (Q_i + q_j) x_{ij} \quad \forall i \in N, j \in N
\]

\[
Q_i \leq Q \quad \forall i \in N
\]  

(1)

Binary variables:

\[
x_{ij} \in \{0, 1\} \quad \forall i \in N, j \in N
\]
Compact model (DARP2)

- Problem: The model does not guarantee that the pickup and delivery of a request are performed by the same vehicle. To ensure this we first define the set $S$ consisting of all node subsets $S \subset N$ such that there is at least one request $i$ for which $i \in S$ but $n + i \notin S$.

- Now the following set of equations (precedence constraints) ensure that each pickup/delivery pair is served by the same vehicle.

\[
\sum_{i \in S} \sum_{j \in N \setminus S} x_{ij} \geq 1 \quad \forall S \in S
\]

The equation simply express that one edge should leave the set (we have to leave the set in order to visit $n + i$).

Example 1:
Compact model (DARP2)

Example 2 (precedence is also ensured by the constraint):

- New problem: $S$ grows exponentially with $n$. Constraints must be generated dynamically.

- Given fractional solution $\bar{x}$ a violated precedence constraint can be found using the following algorithm.

1. Construct a weighted graph $\bar{G} = (N, \bar{A})$ where $\bar{A} = \{(i, j) \in A; \bar{x}_{ij} > 0\}$. Each edge $(i,j)$ in $\bar{A}$ has an associated weight $w_{ij} = \bar{x}_{ij}$

2. for all $i$ in $P$ do
   (a) Find the minimum cut between $i$ and $n + i$ in $\bar{G}$
   (b) If the weight of the minimum cut is less than 1 then a violated inequality has been found

- The correctness of the algorithm follows easily
  - If the weight of minimum cut is less than 1 then the cut identifies a set $S$ that violates the inequality
  - If the weight of minimum cut is greater than or equal to 1 for all $i$ then we can show by contradiction that no precedence constraint will be violated.
Comparing DARP1 to DARP2

- DARP1: Certain extra constraints are easier to represent like:
  - Heterogenous fleet
  - Route duration constraints

- DARP1 can be solved directly using CPLEX, DARP2 needs special implementation.

- DARP2 is expected to solve problems faster
Valid inequalities

Objective: minimize y
Valid inequalities - some examples

Subtour elimination constraints

\[ x_{ij} + x_{ji} + x_{jk} + x_{kj} + x_{ki} + x_{ik} \leq 2 \]

Lifting for directed case:

\[ x_{ij} + 2x_{ji} + x_{jk} + x_{ki} \leq 2 \]

Lifting for DARP case:

\[ x_{ij} + 2x_{ji} + x_{jk} + x_{ki} + x_{n+j,i} + x_{n+k,i} \leq 2 \]

General expression and more liftings described in paper.

Separation algorithms?
Generalized order constraints

\[ x_{i,n+j} + x_{n+j,i} + x_{j,n+k} + x_{n+k,j} + x_{k,n+i} + x_{n+i,k} \leq 2 \]

Lifting for directed case:

\[ x_{i,n+j} + x_{n+j,i} + x_{j,n+k} + x_{n+k,j} + x_{k,n+i} + x_{n+i,k} + x_{ij} + x_{i,n+k} \leq 2 \]

Separation algorithms?
Capacity constraints

\[
\sum_{i \in S} \sum_{j \in N \setminus S} x_{ij} \geq \left\lceil \frac{q(S)}{Q} \right\rceil \quad \forall S \subseteq P \cup D
\]

\[q(S) = \sum_{i \in S} q_i\]

Separation algorithms?
Infeasible path constraints

If the path $i_1 \rightarrow i_2 \rightarrow \ldots \rightarrow i_h$ is infeasible because of time window or ride time constraints (or a combination) then the following inequality is valid:

$$\sum_{i=1}^{h-1} x_{i,i+1} \leq h - 2$$

Can be separated in polynomial time.
Even more compact model (DARP3)

Using some of the inequalities just presented, we can get rid of the $B_i, Q_i$ and $L_i$ variables.

\[
\min \sum_{i \in N} \sum_{j \in N} c_{ij} x_{ij}
\]

\[
\sum_{j \in N} x_{ij} = 1 \quad \forall i \in P
\]

\[
\sum_{i \in N} x_{ij} = 1 \quad \forall j \in P
\]

\[
\sum_{i \in S} \sum_{j \in N \setminus S} x_{ij} \geq 1 \quad \forall S \in S
\]

Infeasible path inequality that ensures that time window, capacities and ride time constraints are obeyed. \(\mathcal{P}\) is the set of all infeasible paths. Each path in \(\mathcal{P}\) is stored as a set of edges.

\[
\sum_{(i,j) \in E^*} x_{ij} \leq |E^*| - 1 \quad \forall E^* \in \mathcal{P}
\]

\[
x_{ij} \in \{0, 1\} \quad \forall i \in N, j \in N
\]
Computational results

See other slide
Conclusion

• A more compact model in terms of number of binary variables was profitable.

• Getting rid of the “superflous” fractional variables didn’t improve running time.

• We have just scratched the surface. There are more to tell, and even more to discover.

• Plenty of open algorithmic questions - how to design good separation routines?