A Joint Routing and Speed Optimization Problem

Fernando Santos - UNIFEI (Brazil)
joint work with Ricardo Fukasawa (U. Waterloo),
Qie He (U. Minnesota) and Yongjia Song (Virginia C. U.)

Column Generation 2016
Búzios
Most of vehicle routing problems assume that vehicles travel at a constant speed.

In the Vehicle Routing Problem with Time Windows (VRPTW) is assumed that vehicles leave the depot at time $t = 0$ and should visit each node $i$ in the interval $[a_i, b_i]$

The Joint Routing and Speed Optimization Problem (JRSOP) proposes to relax the assumption of the VRPTW on which vehicles should travel at a constant over arcs.

Instead, JRSOP introduces new decision variables $v_{ij} \geq 0 : \forall (i, j) \in A$ to determine which speed vehicles should travel over each arc.

Usually the speed is bounded on each arc $l_{ij} \leq v_{ij} \leq u_{ij}$

This way, more feasible solutions are allowed.
Another motivation for the JRSOP is about costs

Usually vehicles fuel consumption are affected by:
- Vehicle weight
- Vehicle speed
- Distance travelled

Also, JRSOP includes driver’s labour costs into the routing solution cost

This way the JRSOP provides a better estimative of costs than the traditional models considering constant speed
### JRSOP cost function

\[
\sum_{(i,j)} F_{ij}(v_{ij}) + \text{Hourly salary } \times \text{Total travel time}
\]

Fuel cost  

Driver’s labour cost

### Fuel costs

\[
F_{ij}(v_{ij}) = \alpha_1 \frac{d_{ij}}{v_{ij}} + \alpha_2 d_{ij} + \alpha_3 d_{ij} f_{ij} + \alpha_4 d_{ij} v_{ij}^2,
\]

where vehicle load \( f_{ij} \) depends on the routing decision
Literature Review

Speed Optimization Problem
- Quadratic time exact algorithm (Kramer et al. 2015)

Pollution Routing Problem - PRP
- MILP formulation for the PRP (Bektaş and Laporte 2011)
- Branch-and-cut for the PRP (Fukasawa et al. 2015)
- Branch-and-price for the PRP (Dabia et al. 2015)
- Adaptive large neighbourhood search (Demir et al. 2012)
- Iterated local search-based heuristics (Kramer et al. 2015)
SP Formulation

Classical SP Formulation

\begin{align*}
\text{min} & \quad \sum_{r \in \Omega} c_r z_r \quad \cdots (1) \\
\text{s.t.} & \quad \sum_{r \in \Omega} a_{ir} z_r = 1, \; i \in V_0, \quad \cdots (2) \\
& \quad \sum_{r \in \Omega} z_r = K, \quad \cdots (3) \\
& \quad z_r \in \mathbb{Z}_+, \; r \in \Omega. \quad \cdots (4)
\end{align*}

- Usually set of routes $\Omega$ stores all feasible vehicle routes
- To adapt this approach for the JRSOP we must consider speeds over arcs
  1) To consider infinity elements in set $\Omega$
  2) To evaluate the optimal speed for each $r \in \Omega$
Proposition 1:
Let \((i, j), (j, k) \in A_r\) be arcs of a given route \(r \in R\). Assuming a convex cost function, if node \(j\) is visited in the interval \((a_j, b_j)\) then optimal speeds \(v_{ij} = v_{jk}\).
Proposition 1:

Let \((i, j), (j, k) \in A_r\) be arcs of a given route \(r \in R\). Assuming a convex cost function, if node \(j\) is visited in the interval \((a_j, b_j)\) then optimal speeds \(v_{ij} = v_{jk}\).
JRSOP SP Formulation

\[
\begin{align*}
\min & \quad \sum_{(r, I, s) \in \Omega} c_{r, I, s} z_{r, I, s} \quad (5a) \\
\text{s.t.} & \quad \sum_{(r, I, s) \in \Omega} a_{ir} z_{r, I, s} = 1, \ i \in V_0, \quad (5b) \\
& \quad \sum_{(r, I, s) \in \Omega} z_{r, I, s} = K, \quad (5c) \\
& \quad z_{r, I, s} \in \mathbb{Z}_+, (r, I, s) \in \Omega. \quad (5d)
\end{align*}
\]

- Elements of set \( \Omega \) are now defined by a triple \((r, I, s)\)
  - \( r \) is the route performed by the vehicle
  - \( I \) is the set of \text{active} nodes of \( r \)
  - \( s \) is the service start time of each node \( j \in I \) \((s_j = \{a_j, b_j\})\)
Assumptions

1. \( t_0 = 0 \)
   Vehicles always leave the depot at time 0

2. \( l_{ij} = l, u_{ij} = u, \forall (i, j) \in A \)
   all arcs have the same lower and upper speed limit

3. Fuel cost function is convex
Labelling Algorithm

For each label extension to a given node $i \in C$, we create 3 new labels:

1. visits $i$ at time $a_i$
2. visits $i$ at time $b_i$
3. visits $i$ in the interval $(a_i, b_i)$
Labelling Algorithm

- For each label extension to a given node $i \in C$, we create 3 new labels:
  1. visits $i$ at time $a_i$
  2. visits $i$ at time $b_i$
  3. visits $i$ in the interval $(a_i, b_i)$
Labelling Algorithm

For each label extension to a given node $i \in C$, we create 3 new labels

1. visits $i$ at time $a_i$
2. visits $i$ at time $b_i$
3. visits $i$ in the interval $(a_i, b_i)$

When a label $L$ terminates it defines recursively a route $r$, a set of active nodes $I$ and a set of service times $s$ of a given triple $(r, I, s) \in \Omega$
The algorithm is able to evaluate the optimal fuel cost of the labels created using extension 1 or 2.

Each label stores 3 extra parameters:
- $w_L$ stores the last node in the path on which a label hit a time window.
- $s_L$ stores the time on which node $w_L$ is visited: $a_{w_L}$ or $b_{w_L}$.
- $C_L$ stores the optimal fuel cost up until node $w_L$. 
Labelling Algorithm

Using Proposition 1 and Assumptions 1 and 2, the algorithm is able to evaluate and store the fuel cost of labels created in extensions 1 and 2.

Labels created by extension 3 store the coefficients of time and cost functions

\[ F_L(v_{WL}) = \alpha_4 d_{WL} v_{WL}^2 + \alpha_1 \frac{d_{WL}}{v_{WL}} + p_L \]

\[ T_L(v_{WL}) = s_L + \frac{d_{WL}}{v_{WL}} \]
Labelling Algorithm

Label attributes
- $i$: last node visited
- $\bar{N}$: set of visited nodes
- $w$: last node in the path on which a label hit a time windows
- $s$: time on which node $w_L$ is visited: $a_{w_L}$ or $b_{w_L}$
- $V$: interval with feasible speeds after $L$ visits $w_L$
- $T(v)$: arriving time on node $i$
- $F(v)$: fuel cost from node 0 to $i$

Remark
If $i \neq w$, $T(v)$ and $F(v)$ store the coefficients of functions in terms of $v \in V$. Otherwise, they store constant values.
Labelling Algorithm

### General Dominance Rule
Given labels $L_A$ and $L_B$, we say that $L_A$ dominates $L_B$ if

1. $i_A = i_B$
2. $E(L_B) \subseteq E(L_A)$
3. $\forall L \in E(L_B) : c(L_A \oplus L) \leq c(L_B \oplus L)$

### Dominance Rule for JRSOP
Given labels $L_A$ and $L_B$, we say that $L_A$ dominates $L_B$ if

1. $i_A = i_B$
2. $\tilde{N}_A \subseteq \tilde{N}_B$
3. $T_A(v) \leq T_B(v)$
4. $F_A(v) - \sum_{n \in \tilde{N}_A} \delta_n \leq F_B(v) - \sum_{n \in \tilde{N}_B} \delta_n$
if \( i_A = i_B = w_A = w_B \), all 4 conditions can be checked by comparing constant values
Otherwise, $L_A$ and/or $L_B$ visit $i$ on the interval $(a_i, b_i)$. 

\[ i \quad \left[ \begin{array}{c} L_A \\ L_B \end{array} \right] \]

Conditions 3 and 4 are checked as following 

For any $v \in V_B$, there exists $v' \in V_A$ such that

\[ T_A(v') \leq T_B(v) \quad \text{and} \]

\[ F_A(v') - \sum_{n \in \bar{N}_A} \delta_n \leq F_B(v) - \sum_{n \in \bar{N}_B} \delta_n \]
Labelling Algorithm

\[ g(v) = \min_{v' \in V_A} \{ F(v') - \sum_{n \in \tilde{N}_A} \delta_n \mid T_A(v') \leq T_B(v) \} \]

\[ g(v) = \min_{v' \in V_A} \{ F(v') - \sum_{n \in \tilde{N}_A} \delta_n \mid s_A + \frac{d_{wA_i}}{v'} \leq s_B + \frac{d_{wB_i}}{v} \} \]

\[ g(v) = \min\{ F(v') - \sum_{n \in \tilde{N}_A} \delta_n \mid s_A + \frac{d_{wA_i}}{v'} \leq s_B + \frac{d_{wB_i}}{v}, V_A^{min} \leq v' \leq V_A^{max} \} \]

\[ g(v) = \min\{ F(v') - \sum_{n \in \tilde{N}_A} \delta_n \mid \max\{ \frac{d_{wA_i}v}{(s_b - s_A)v + d_{wB_i}}, V_A^{min} \} \leq v' \leq V_A^{max} \} \]

Resulting optimization problem

\[ D = \max_{v \in V_B} \{ g(v) - F(v) - \sum_{n \in \tilde{N}_B} \delta_n \} \]  \hspace{1cm} (6)
Solution of (10) is obtained by solving a continuous differentiable problem

- Optimal solution is among KKT points (points with derivative zero or boundary points)
- Finding stationary points amounts to solving roots of a degree 4 polynomial.
  - If $D < 0$ on the interval $V_B$, $L_A$ dominates $L_B$
  - Otherwise, no dominance is allowed.
Implementation Details

Cuts

- Inequalities introduced on model (5)-(8) to improve the LP lower bounds

[box]

Capacity Cuts - derived from VRP

\[ \sum_{i \in S} \sum_{j \notin S} x_{ij} \geq \pi(S) : S \subseteq C \]  

- \( x_{ij} \) is the value assumed by arc \((i, j) \in A\) in a given LP solution
- \( \pi(S) \) is the minimum number of vehicles to attend customers of subset \( S \)

- Cuts are separated via a heuristic from Lysgaard (2004)
Q-routes

- Elementary routes are harder to evaluate
  - Label $L_A$ dominates $L_B$ if $\tilde{N}_A \subseteq \tilde{N}_B$
  - This condition prevents many labels to be discarded
- Christofides et al. (1981) proposed Q-routes
  - Q-route is a walk on the Graph that respect the vehicle’s capacity
  - Relax that condition imposing customers are visited at most once
  - Condition $\tilde{N}_A \subseteq \tilde{N}_B$ is replaced by $q_A \leq q_B$
- Set of all feasible Q-routes include all elementary routes
Test instances (Bektaş and Laporte 2011, Demir et al. 2012, Kramer et al. 2014)
- Based on UK cities, 10-city instances to 25-city instances
- Three series: UK-A, UK-B, UK-C.
- Widths of time windows: UK-A > UK-C > UK-B

SCIP as the framework for branch-cut-and-price

All code implemented in C++

One-hour time limit
Computational Results

The graph displays computational results for different scenarios.

- **Percentage of instances (x 100%)**

- **Times slower than fastest**

Scenarios include:
- Elementary
- q-routes
- Elementary + cuts
- q-routes + cuts
- 2-cycle-free
- 2-cycle-free + cuts

The graph illustrates the performance comparison among these scenarios over a range of times slower than the fastest.
## Computational Results

<table>
<thead>
<tr>
<th>instance</th>
<th>Branch-and-cut Algorithm</th>
<th>Branch-and-cut-and-price</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>optimal</td>
<td>time(s)</td>
</tr>
<tr>
<td>UK10A-1</td>
<td>170.64</td>
<td>1354.4</td>
</tr>
<tr>
<td>UK10A-2</td>
<td>204.88</td>
<td>813.7</td>
</tr>
<tr>
<td>UK10A-3</td>
<td>200.34</td>
<td>1708.3</td>
</tr>
<tr>
<td>UK10A-4</td>
<td>189.88</td>
<td>844.9</td>
</tr>
<tr>
<td>UK10A-5</td>
<td>175.59</td>
<td>2649.2</td>
</tr>
<tr>
<td>UK10A-6</td>
<td>214.48</td>
<td>1472.8</td>
</tr>
<tr>
<td>UK10A-7</td>
<td>190.14</td>
<td>882.5</td>
</tr>
<tr>
<td>UK10A-8</td>
<td>222.17</td>
<td>564.3</td>
</tr>
<tr>
<td>UK10A-9</td>
<td>174.54</td>
<td>352.0</td>
</tr>
<tr>
<td>UK10A-10</td>
<td>189.82</td>
<td>211.1</td>
</tr>
</tbody>
</table>
## Computational Results

<table>
<thead>
<tr>
<th>instance</th>
<th>Branch-and-cut Algorithm</th>
<th>Branch-and-cut-and-price</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>optimal</td>
<td>time(s)</td>
</tr>
<tr>
<td>UK10A-1</td>
<td>170.64</td>
<td>1354.4</td>
</tr>
<tr>
<td>UK10A-2</td>
<td>204.88</td>
<td>813.7</td>
</tr>
<tr>
<td>UK10A-3</td>
<td>200.34</td>
<td>1708.3</td>
</tr>
<tr>
<td>UK10A-4</td>
<td>189.88</td>
<td>844.9</td>
</tr>
<tr>
<td>UK10A-5</td>
<td>175.59</td>
<td>2649.2</td>
</tr>
<tr>
<td>UK10A-6</td>
<td>214.48</td>
<td>1472.8</td>
</tr>
<tr>
<td>UK10A-7</td>
<td>190.14</td>
<td>882.5</td>
</tr>
<tr>
<td>UK10A-8</td>
<td>222.17</td>
<td>564.3</td>
</tr>
<tr>
<td>UK10A-9</td>
<td>174.54</td>
<td>352.0</td>
</tr>
<tr>
<td>UK10A-10</td>
<td>189.82</td>
<td>211.1</td>
</tr>
</tbody>
</table>
## Computational Results

<table>
<thead>
<tr>
<th>instance</th>
<th>Branch-and-cut Algorithm</th>
<th>Branch-and-cut-and-price</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>optimal</td>
<td>time(s)</td>
</tr>
<tr>
<td>UK20A-1</td>
<td>352.45</td>
<td>3600</td>
</tr>
<tr>
<td>UK20A-2</td>
<td>365.77</td>
<td>3600</td>
</tr>
<tr>
<td>UK20A-3</td>
<td>230.49</td>
<td>3600</td>
</tr>
<tr>
<td>UK20A-4</td>
<td>347.04</td>
<td>3600</td>
</tr>
<tr>
<td>UK20A-5</td>
<td>329.63</td>
<td>3600</td>
</tr>
<tr>
<td>UK20A-6</td>
<td>367.73</td>
<td>3600</td>
</tr>
<tr>
<td>UK20A-7</td>
<td>258.75</td>
<td>3600</td>
</tr>
<tr>
<td>UK20A-8</td>
<td>303.17</td>
<td>3600</td>
</tr>
<tr>
<td>UK20A-9</td>
<td>362.56</td>
<td>3600</td>
</tr>
<tr>
<td>UK20A-10</td>
<td>317.79</td>
<td>3600</td>
</tr>
</tbody>
</table>
### Computational Results

<table>
<thead>
<tr>
<th>instance</th>
<th>Branch-and-cut Algorithm</th>
<th>Branch-and-cut-and-price</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>optimal</td>
<td>time(s)</td>
</tr>
<tr>
<td>UK20A-1</td>
<td>352.45</td>
<td>3600</td>
</tr>
<tr>
<td>UK20A-2</td>
<td>365.77</td>
<td>3600</td>
</tr>
<tr>
<td>UK20A-3</td>
<td>230.49</td>
<td>3600</td>
</tr>
<tr>
<td>UK20A-4</td>
<td>347.04</td>
<td>3600</td>
</tr>
<tr>
<td>UK20A-5</td>
<td>329.63</td>
<td>3600</td>
</tr>
<tr>
<td>UK20A-6</td>
<td>367.73</td>
<td>3600</td>
</tr>
<tr>
<td>UK20A-7</td>
<td>258.75</td>
<td>3600</td>
</tr>
<tr>
<td>UK20A-8</td>
<td>303.17</td>
<td>3600</td>
</tr>
<tr>
<td>UK20A-9</td>
<td>362.56</td>
<td>3600</td>
</tr>
<tr>
<td>UK20A-10</td>
<td>317.79</td>
<td>3600</td>
</tr>
</tbody>
</table>
Concluding Remarks

- The proposed algorithm outperforms the previous approach for all instances of our test set.
- The framework can be applied to any JRSOP variants as long as the cost is convex in the speed.
- As future work, we suggest allowing variable departure time at the depot.