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

Introduction

- 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

Introduction

- 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

Cost Function

JRSOP cost function

$$\underbrace{\sum_{(i,j)} F_{ij}(v_{ij})}_{\text{Fuel cost}} + \underbrace{\text{Hourly salary} \times \text{Total travel time}}_{\text{Driver's labour cost}}$$

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

$$\min \sum_{r \in \Omega} c_r z_r \tag{1}$$

$$\text{s.t.} \sum_{r \in \Omega} a_{ir} z_r = 1, i \in V_0, \tag{2}$$

$$\sum_{r \in \Omega} z_r = K, \tag{3}$$

$$z_r \in \mathbb{Z}_+, r \in \Omega. \tag{4}$$

- Usually set of routes Ω 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 Ω
  2) To evaluate the optimal speed for each $r \in \Omega$

JRSOP speed property

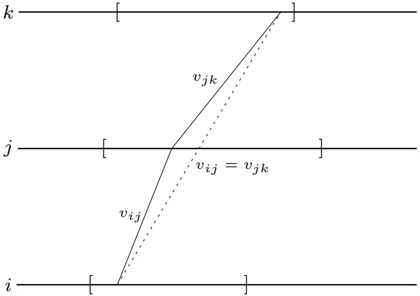
Proposition 1:

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 speed property

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

JRSOP SP Formulation

$$\min \sum_{(r,l,\mathbf{s})\in\Omega} c_{r,l,\mathbf{s}} z_{r,l,\mathbf{s}} \tag{5a}$$

$$\text{s.t.} \sum_{(r,l,\mathbf{s})\in\Omega} a_{ir} z_{r,l,\mathbf{s}} = 1, i \in V_0, \tag{5b}$$

$$\sum_{(r,l,\mathbf{s})\in\Omega} z_{r,l,\mathbf{s}} = K, \tag{5c}$$

$$z_{r,l,\mathbf{s}} \in \mathbb{Z}_+, (r,l,\mathbf{s}) \in \Omega. \tag{5d}$$

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

JRSOP SP Formulation

Assumptions

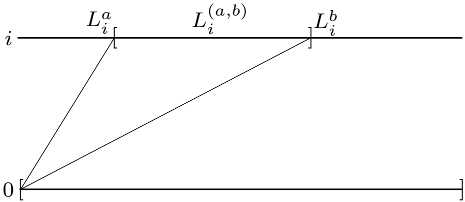
1. $t_0 = 0$
   Vehicles always leave the depot at time 0
2. $l_{ij} = l, u_{ij} = u, \ \forall (i,j) \in A$
   all arcs have the same lower and upper speed limit
3. Fuel cost function is convex

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)

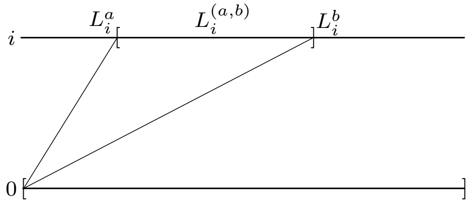
Labelling Algorithm

- 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 l and a set of service times s of a given triple $(r, l, s) \in \Omega$

Labelling Algorithm

- 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 windows
  - 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

$$F_L(v_{w_L i}) = \alpha_4 d_{w_L i} v_{w_L i}^2 + \alpha_1 \frac{d_{w_L i}}{v_{w_L i}} + p_L$$

$$T_L(v_{w_L i}) = s_L + \frac{d_{w_L i}}{v_{w_L i}}$$

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. $\forall L \in E(L_B) : c(L_A \oplus L) \leq c(L_B \oplus L)$

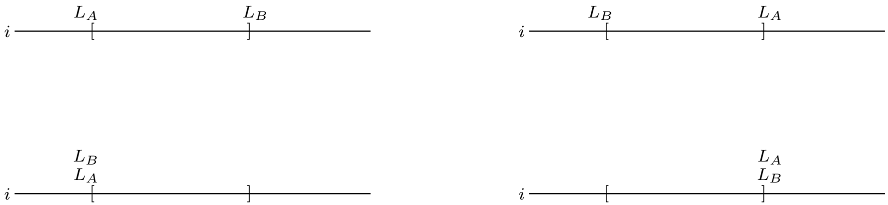
Dominance Rule for JRSOP

Given labels L_A and L_B, we say that L_A dominates L_B if

1. $i_A = i_B$
2. $\bar{N}_A \subseteq \bar{N}_B$
3. $T_A(v) \leq T_B(v)$
4. $F_A(v) - \sum_{n \in \bar{N}_A} \delta_n \leq F_B(v) - \sum_{n \in \bar{N}_B} \delta_n$

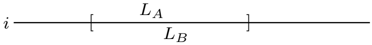
Labelling Algorithm

- if $i_A = i_B = w_A = w_B$, all 4 conditions can be checked by comparing constant values



Labelling Algorithm

- Otherwise, L_A and/or L_B visit i on the interval (a_i, b_i)



- Conditions 3 and 4 are checked as following

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

$$T_A(v') \leq T_B(v) \text{ and}$$

$$F_A(v') - \sum_{n \in \bar{N}_A} \delta_n \leq F_B(v) - \sum_{n \in \bar{N}_B} \delta_n$$

Labelling Algorithm

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

$$g(v) = \min_{v' \in V_A} \{F(v') - \sum_{n \in \bar{N}_A} \delta_n \mid s_A + \frac{d_{w_A i}}{v'} \leq s_B + \frac{d_{w_B i}}{v}\}$$

$$g(v) = \min\{F(v') - \sum_{n \in \bar{N}_A} \delta_n \mid s_A + \frac{d_{w_A i}}{v'} \leq s_B + \frac{d_{w_B i}}{v}, V_A^{\min} \leq v' \leq V_A^{\max}\}$$

$$g(v) = \min\{F(v') - \sum_{n \in \bar{N}_A} \delta_n \mid \max\{\frac{d_{w_A i} v}{(s_b - s_A)v + d_{w_B i}}, V_A^{\min}\} \leq v' \leq V_A^{\max}\}$$

Resulting optimization problem

$$D = \max_{v \in V_B}\{g(v) - F(v) - \sum_{n \in \bar{N}_B} \delta_n\} \tag{6}$$

Labelling Algorithm

- 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

Capacity Cuts - derived from VRP

$$\sum_{i \in S} \sum_{j \notin S} x_{ij} \geq \pi(S) : S \subseteq C \quad (7)$$

- 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)

Implementation Details

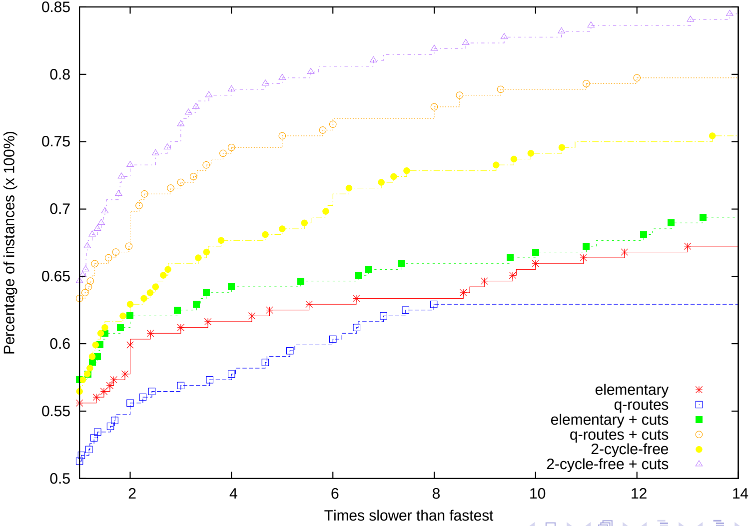
Q-routes

- Elementary routes are harder to evaluate
  - Label L_A dominates L_B if $\bar{N}_A \subseteq \bar{N}_B$
  - 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 $\bar{N}_A \subseteq \bar{N}_B$ is replaced by $q_A \leq q_B$
- Set of all feasible Q-routes include all elementary routes

Computational Settings

- 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



Computational Results

	Branch-and-cut Algorithm			Branch-and-cut-and-price		
instance	optimal	time(s)	gap	optimal	time(s)	gap(%)
UK10A-1	170.64	1354.4	0.0%	170.64	3.3	0.0%
UK10A-2	204.88	813.7	0.0%	204.88	1.5	0.0%
UK10A-3	200.34	1708.3	0.0%	200.34	0.6	0.0%
UK10A-4	189.88	844.9	0.0%	189.88	3.9	0.0%
UK10A-5	175.59	2649.2	0.0%	175.59	3.5	0.0%
UK10A-6	214.48	1472.8	0.0%	214.48	0.7	0.0%
UK10A-7	190.14	882.5	0.0%	190.14	4.0	0.0%
UK10A-8	222.17	564.3	0.0%	222.17	0.2	0.0%
UK10A-9	174.54	352.0	0.0%	174.54	5.4	0.0%
UK10A-10	189.82	211.1	0.0%	189.82	0.5	0.0%

Computational Results

	Branch-and-cut Algorithm			Branch-and-cut-and-price		
instance	optimal	time(s)	gap	optimal	time(s)	gap(%)
UK10A-1	170.64	1354.4	0.0%	170.64	3.3	0.0%
UK10A-2	204.88	813.7	0.0%	204.88	1.5	0.0%
UK10A-3	200.34	1708.3	0.0%	200.34	0.6	0.0%
UK10A-4	189.88	844.9	0.0%	189.88	3.9	0.0%
UK10A-5	175.59	2649.2	0.0%	175.59	3.5	0.0%
UK10A-6	214.48	1472.8	0.0%	214.48	0.7	0.0%
UK10A-7	190.14	882.5	0.0%	190.14	4.0	0.0%
UK10A-8	222.17	564.3	0.0%	222.17	0.2	0.0%
UK10A-9	174.54	352.0	0.0%	174.54	5.4	0.0%
UK10A-10	189.82	211.1	0.0%	189.82	0.5	0.0%

Computational Results

	Branch-and-cut Algorithm			Branch-and-cut-and-price		
instance	optimal	time(s)	gap	optimal	time(s)	gap(%)
UK20A-1	352.45	3600	22.9%	351.82	24.1	0.0%
UK20A-2	365.77	3600	20.7%	365.77	3.8	0.0%
UK20A-3	230.49	3600	23.6%	230.49	25.1	0.0%
UK20A-4	347.04	3600	21.2%	347.04	109	0.0%
UK20A-5	329.63	3600	24.3%	323.44	26.3	0.0%
UK20A-6	367.73	3600	25.0%	364.23	27.2	0.0%
UK20A-7	258.75	3600	23.3%	258.75	3600	7.3%
UK20A-8	303.17	3600	23.0%	301.51	19.5	0.0%
UK20A-9	362.56	3600	19.5%	362.56	17.4	0.0%
UK20A-10	317.79	3600	26.3%	313.33	20.1	0.0%

Computational Results

	Branch-and-cut Algorithm			Branch-and-cut-and-price		
instance	optimal	time(s)	gap	optimal	time(s)	gap(%)
UK20A-1	352.45	3600	22.9%	351.82	24.1	0.0%
UK20A-2	365.77	3600	20.7%	365.77	3.8	0.0%
UK20A-3	230.49	3600	23.6%	230.49	25.1	0.0%
UK20A-4	347.04	3600	21.2%	347.04	109	0.0%
UK20A-5	329.63	3600	24.3%	323.44	26.3	0.0%
UK20A-6	367.73	3600	25.0%	364.23	27.2	0.0%
UK20A-7	258.75	3600	23.3%	258.75	3600	7.3%
UK20A-8	303.17	3600	23.0%	301.51	19.5	0.0%
UK20A-9	362.56	3600	19.5%	362.56	17.4	0.0%
UK20A-10	317.79	3600	26.3%	313.33	20.1	0.0%

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 allow variable departure time at the depot