Branch-and-Price-and-Cut for the Split Delivery Vehicle Routing Problem with Time Windows

Guy Desaulniers

École Polytechnique de Montréal and GERAD

Column Generation 2008 Aussois, France
Introduction

- Problem definition
- Literature review
- Main difficulty with branch-and-price for SDVRPTW

Formulation

- Master problem
- Subproblem

Branch-and-price-and-cut method

- Column generation
- Cutting and branching

Computational results

Conclusions
Outline

1. Introduction
   - Problem definition
   - Literature review
   - Main difficulty with branch-and-price for SDVRPTW

2. Formulation
   - Master problem
   - Subproblem

   - Column generation
   - Cutting and branching

4. Computational results

5. Conclusions
Vehicle Routing Problem with Time Windows (VRPTW)

- **Given**
  - unlimited number of identical vehicles with a given capacity, housed in a single depot
  - set of customers with known demands
  - a time window for each customer

- **Find** vehicle routes such that
  - all customer demands are met
  - each customer is visited by a single vehicle
  - each route starts and ends at the depot
  - each route satisfies vehicle capacity and time windows
  - total distance is minimized
Given

- unlimited number of identical vehicles with a given capacity, housed in a single depot
- set of customers with known demands
- a time window for each customer

Find vehicle routes such that

- all customer demands are met
- each customer is visited by a single vehicle
- each route starts and ends at the depot
- each route satisfies vehicle capacity and time windows
- total distance is minimized
The Split Delivery VRPTW (SDVRPTW)

Same as the VRPTW except
- several vehicles can visit each customer
- demand can be split (split deliveries)
- demand can be greater than vehicle capacity

SDVRP was introduced by Dror and Trudeau (1989, 1990)

SDVRPTW was studied first by Frizzell and Giffin (1995)
Literature on the SDVRP

Not well-studied problem

- Close to 10 heuristics (2 based on branch-and-price)
- A few exact methods
  - Branch-and-cut method by Dror et al. (1994)
  - Cutting plane method by Belenguer et al. (2000)
  - Dynamic programming method by Lee et al. (2006)
  - Iterative two-stage method by Jin et al. (2007)
Literature on the SDVRPTW

Very few papers

- Construction and improvement heuristics by Frizzell and Giffin (1995) and Mullaseril et al. (1997)
- One exact branch-and-price method by Gendreau et al. (2006)
Branch-and-price

Leading methodology for the VRPTW

- Each column in the master problem corresponds to a feasible route
- Subproblem is an elementary shortest path problem with resource constraints
Main difficulty with branch-and-price for SDVRPTW

Delivered quantities are decision variables

- Branch-and-price heuristics of Sierksma and Tijssen (1998) and Jin et al. (2008)
  - For a given route, determining the delivered quantities in the subproblem is the linear relaxation of a bounded knapsack problem
  - Maximum of one split customer per route
  - All other customers receive a full delivery
  - Integrality requirements on the master problem dynamic columns (not valid for an exact method)
Main difficulty with branch-and-price for SDVRPTW

Delivered quantities are decision variables

- Branch-and-price heuristics of Sierksma and Tijssen (1998) and Jin et al. (2008)
  - For a given route, determining the delivered quantities in the subproblem is the linear relaxation of a bounded knapsack problem
  - Maximum of one split customer per route
  - All other customers receive a full delivery
  - Integrality requirements on the master problem dynamic columns (not valid for an exact method)
Main difficulty with branch-and-price for SDVRPTW

- Exact branch-and-price method of Gendreau et al. (2006)
  - Subproblem generates only routes
  - Quantities are decided in the master problem
  - *Exponential numbers* of quantity variables and constraints in the master problem (depend on number of generated routes)
Our contribution

New branch-and-price method

- Subproblem is an elementary shortest path problem with resource constraints, combined with the linear relaxation of a bounded knapsack problem
- Maximum of one split customer per route
- Other customers receive a zero or a full delivery
- Integrality requirements not on the master problem dynamic columns
- Convex combinations of these columns can yield routes with multiple split deliveries
- Specialized dynamic programming algorithm for the subproblem
Our contribution

New branch-and-price method

- Subproblem is an elementary shortest path problem with resource constraints, combined with the linear relaxation of a bounded knapsack problem
- Maximum of one split customer per route
- Other customers receive a zero or a full delivery
  - Integrality requirements not on the master problem dynamic columns
  - Convex combinations of these columns can yield routes with multiple split deliveries
  - Specialized dynamic programming algorithm for the subproblem
Our contribution

New branch-and-price method

- Subproblem is an elementary shortest path problem with resource constraints, combined with the linear relaxation of a bounded knapsack problem
- Maximum of one split customer per route
- Other customers receive a zero or a full delivery
- Integrality requirements not on the master problem dynamic columns
- **Convex combinations** of these columns can yield routes with multiple split deliveries
- Specialized dynamic programming algorithm for the subproblem
Our contribution

New branch-and-price method

- Subproblem is an elementary shortest path problem with resource constraints, combined with the linear relaxation of a bounded knapsack problem
- Maximum of one split customer per route
- Other customers receive a zero or a full delivery
- Integrality requirements not on the master problem dynamic columns
- Convex combinations of these columns can yield routes with multiple split deliveries
- Specialized dynamic programming algorithm for the subproblem
Outline

1. Introduction
   - Problem definition
   - Literature review
   - Main difficulty with branch-and-price for SDVRPTW

2. Formulation
   - Master problem
   - Subproblem

   - Column generation
   - Cutting and branching

4. Computational results

5. Conclusions
Integer master problem

Minimize \[ \sum_{r \in R} \sum_{w \in W_r} c_r \theta_{rw} \]

subject to:

\[ \sum_{r \in R} \sum_{w \in W_r} \delta_{iw} \theta_{rw} \geq d_i, \quad \forall i \in \mathcal{N} \]

\[ \sum_{r \in R} \sum_{w \in W_r} a_{ir} \theta_{rw} \geq k_i^C, \quad \forall i \in \mathcal{N} \]

\[ \sum_{r \in R \setminus \{0\}} \sum_{w \in W_r} \theta_{rw} = H, \]

\[ H \in \left[k^C(\mathcal{N}), |\mathcal{F}|\right], \text{ integer,} \]
Integer master problem (cont’d)

\[ \theta_{rw} \geq 0, \quad \forall \ r \in \mathcal{R}, \ w \in \mathcal{W}_r \]

\[ \sum_{r \in \mathcal{R}} \sum_{w \in \mathcal{W}_r} x_{ijr} \theta_{rw} = y_{ij}, \quad \forall \ (i, j) \in \mathcal{A} \]

\[ \sum_{r \in \mathcal{R}} \sum_{w \in \mathcal{W}_r} b_{ij\ell r} \theta_{rw} = z_{ij\ell}, \quad \forall \ (i, j, \ell) \in \mathcal{B} \]

\[ y_{ij} \text{ binary,} \quad \forall \ (i, j) \in \mathcal{A}(\mathcal{N}) \]

\[ y_{ij} \text{ integer,} \quad \forall \ (i, j) \in \mathcal{A} \setminus \mathcal{A}(\mathcal{N}) \]

\[ z_{ij\ell} \text{ binary,} \quad \forall \ (i, j, \ell) \in \mathcal{B}. \]
A small example with $n = 3$ customers and $Q = 4$

Optimal solution is (*delivered quantities in parenthesis*)

- $0-1(4)-4$ with a flow of 1, $0-3(4)-4$ with a flow of 1
- $0-1(2)-2(2)-3(0)-4$ with a flow of 0.5
- $0-1(0)-2(2)-3(2)-4$ with a flow of 0.5
Subproblem

Minimize \[ \sum_{(i,j) \in A} (c_{ij} - \alpha_i - \nu_{ij})x_{ij} - \sum_{i \in N} \pi_i \delta_i - \beta - \sum_{(i,j,\ell) \in B} \mu_{ij\ell} \xi_{ij\ell} \]

subject to:

\[ \sum_{i \in N} x_{0,i} = 1, \]

\[ \sum_{j \in V^+(i)} x_{ij} - \sum_{j \in V^-(i)} x_{ji} = 0, \quad \forall i \in N \]

\[ x_{ij}(s_i + t_{ij} - s_j) \leq 0, \quad \forall (i,j) \in A \]

\[ e_i \leq s_i \leq l_i, \quad \forall i \in V \]
Subproblem (cont’d)

\[
\sum_{i \in N} \delta_i \leq Q,
\]

\[0 \leq \delta_i \leq \bar{d}_i \sum_{j \in V^+(i)} x_{ij}, \quad \forall i \in N\]

\[x_{ij} \in \{0, 1\}, \quad \forall (i, j) \in A\]

\[x_{ij} + x_{j\ell} \leq \zeta_{ij\ell} + 1 \leq \frac{x_{ij} + x_{j\ell}}{2} + 1, \quad \forall (i, j, \ell) \in B\]

\[\zeta_{ij\ell} \in \{0, 1\}, \quad \forall (i, j, \ell) \in B.\]

Assuming no \(\zeta\) variables, the subproblem is an elementary shortest path problem with time windows and vehicle capacity, combined with the linear relaxation of a bounded knapsack problem.
Outline

1. Introduction
   - Problem definition
   - Literature review
   - Main difficulty with branch-and-price for SDVRPTW

2. Formulation
   - Master problem
   - Subproblem

   - Column generation
   - Cutting and branching

4. Computational results

5. Conclusions
Branch-and-price-and-cut

- Column generation used to compute lower bounds
- Cutting planes added to strengthen linear relaxations
- Branching used to derive integer solutions
Column generation

- Standard column generation
- Label-setting algorithm for solving the subproblem
- Accelerating strategies
Label-setting algorithm

For the VRPTW

- **Label** $E$
  - $C$: reduced cost
  - $S$: time
  - $L$: total quantity delivered
  - $V^i$: customer $i$ reachable or not

- **Extension functions**: e.g., $S_j = \max\{e_j, S_i + t_{ij}\}$

- **Dominance**: $E_1$ dominates $E_2$ if
  - $C_1 \leq C_2$
  - $S_1 \leq S_2$
  - $L_1 \leq L_2$
  - $V^i_1 \leq V^i_2$, $\forall i \in \mathcal{N}$
Label-setting algorithm

Not applicable because
- delivered quantities are decision variables
- reduced cost and load resource are functions of these quantities

New algorithm for the SDVRPTW
- When extending a label along an arc, up to three labels can be created: zero, split, and full deliveries
- New binary resource to limit the number of split deliveries
- Reduced cost is a linear function of the total quantity delivered
- Dominance procedure must compare such functions
Label-setting algorithm

Not applicable because

- delivered quantities are decision variables
- reduced cost and load resource are functions of these quantities

New algorithm for the SDVRPTW

- When extending a label along an arc, up to three labels can be created: zero, split, and full deliveries
- New binary resource to limit the number of split deliveries
- Reduced cost is a linear function of the total quantity delivered
- Dominance procedure must compare such functions
Label-setting algorithm (cont’d)

Label \( E \)

- \( C \): reduced cost excluding cost of split delivery (if any)
- \( S \): time
- \( L \): total quantity delivered in full deliveries
- \( V^i \): customer \( i \) reachable or not
- \( P \): split delivery done or not
- \( \Delta \): maximum quantity that can be delivered in the split delivery (if any)
- \( \Pi \): unit dual price for the split delivery (if any)

Adapted extension functions
Label-setting algorithm (cont’d)

<table>
<thead>
<tr>
<th>Label E</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C$ : reduced cost <strong>excluding cost of split delivery (if any)</strong></td>
</tr>
<tr>
<td>$S$ : time</td>
</tr>
<tr>
<td>$L$ : total quantity delivered in <strong>full deliveries</strong></td>
</tr>
<tr>
<td>$V^i$ : customer $i$ reachable or not</td>
</tr>
<tr>
<td>$P$ : split delivery done or not</td>
</tr>
<tr>
<td>$\Delta$ : maximum quantity that can be delivered in the split delivery (if any)</td>
</tr>
<tr>
<td>$\Pi$ : unit dual price for the split delivery (if any)</td>
</tr>
</tbody>
</table>

- Adapted extension functions
Label-setting algorithm (cont’d)

$E_1$ dominates $E_2$ if

- $E_1$ and $E_2$ are associated with the same node
- all feasible extensions of $E_2$ are also feasible for $E_1$
- cost of every feasible extension of $E_2$ is greater than or equal to the cost of a similar extension of $E_1$
Label-setting algorithm (cont’d)

Cost of a label $E = (C, S, L, V^i, P, \Delta, \Pi)$ is

$$Z(G) = C - \Pi(G - L) \quad \text{for } G \in [L, L + \Delta]$$

$E_1$ can dominate $E_2$, but not the other labels.
**Label-setting algorithm (cont’d)**

*E*₁ can dominate *E*₂ if

- \( S_1 \leq S_2 \)
- \( L_1 \leq L_2 \)
- \( V^i_1 \leq V^i_2, \forall \; i \in \mathcal{N} \)
- \( P_1 \leq P_2 \)
- \( C_1 - \Delta_1 \Pi_1 \leq C_2 - \Delta_2 \Pi_2 \)
- \( C_1 - (L_2 - L_1) \Pi_1 \leq C_2 \)
- \( C_1 - (L_2 + \Delta_2 - L_1) \Pi_1 \leq C_2 - \Delta_2 \Pi_2 \)
Accelerating strategies

1. Initial columns: dedicated trips 0 – i – n + 1
2. If \( \pi_j = 0 \), then only zero deliveries at \( j \)
4. Heuristic column generator (omit \( V_1^i \leq V_2^i \), \( \forall i \in \mathcal{N} \), in the dominance rule)
Accelerating strategies

1. Initial columns: dedicated trips $0 - i - n + 1$
2. If $\pi_j = 0$, then only zero deliveries at $j$
4. Heuristic column generator (omit $V_1^i \leq V_2^i, \forall i \in \mathcal{N}$, in the dominance rule)
Accelerating strategies

1. Initial columns: dedicated trips $0 - i - n + 1$
2. If $\pi_j = 0$, then only zero deliveries at $j$
4. Heuristic column generator (omit $V_1^i \leq V_2^i$, $\forall i \in \mathcal{N}$, in the dominance rule)
Accelerating strategies

1. Initial columns: dedicated trips $0 - i - n + 1$
2. If $\pi_j = 0$, then only zero deliveries at $j$
4. Heuristic column generator (omit $V_1^i \leq V_2^i, \forall i \in \mathcal{N}$, in the dominance rule)
Cutting

- **k-path inequalities** (Kohl et al., 1999)
  - \( k_U = \max\{k^C_U, k^T_U\} \), where
    - \( k^C_U = \lceil \sum_{i \in U} \frac{d_i}{q} \rceil \): minimum according to vehicle capacity
    - \( k^T_U \): minimum according to time windows (1 or 2)

- **Arc-flow inequalities** (Gendreau et al, 2006)
  - Maximum flow of 1 on arcs \((i, j)\) and \((j, i)\) for \(i, j \in \mathcal{N}\)
**Cutting**

- *k*-path inequalities (Kohl et al., 1999)
  - \( k_U = \max\{k^C_U, k^T_U\} \), where
    - \( k^C_U = \lceil \sum_{i \in U} \frac{d_i}{Q} \rceil \) : minimum according to vehicle capacity
    - \( k^T_U \) : minimum according to time windows (1 or 2)

- Arc-flow inequalities (Gendreau et al, 2006)
  - Maximum flow of 1 on arcs \((i, j)\) and \((j, i)\) for \(i, j \in \mathcal{N}\)
Branching

In order of priority, we branch on

- number of vehicles used ($H$)
- number of vehicles visiting a customer ($\sum_j y_{ij}$)
  - add $y_{ij}$ and corresponding constraint in the master problem
- number of vehicles on an arc ($y_{ij}$)
  - add $y_{ij}$ and corresponding constraint in the master problem
- number of vehicles on two consecutive arcs ($z_{ij\ell}$)
  - add $z_{ij\ell}$ and corresponding constraint in the master problem
  - modify the subproblem algorithm
Outline

1 Introduction
   - Problem definition
   - Literature review
   - Main difficulty with branch-and-price for SDVRPTW

2 Formulation
   - Master problem
   - Subproblem

3 Branch-and-price-and-cut method
   - Column generation
   - Cutting and branching

4 Computational results

5 Conclusions
Instances

Modified VRPTW instances of Solomon (1987)

- Allow split deliveries

- Solomon: 56 instances with 100 customers (6 classes)
- $2 \times 56$ other instances taking the first 25 and 50 customers
- $Q = 30, 50, 100$
- Total of 504 instances

- Same instances as in Gendreau et al. (2006)

Maximum CPU time = 1 hour, 2.8GHz PC
Linear relaxation results

Computational time

- increases with number of customers
- decreases with capacity $Q$

- Slower increase than with the method of Gendreau et al. (2006) who used a 1.6GHz PC

Example (C1 instances with $Q = 30$)

<table>
<thead>
<tr>
<th>Gendreau et al.</th>
<th>0.3, 8, 304 seconds for $n = 25, 50, 100$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Our method</td>
<td>0.4, 3, 17 seconds for $n = 25, 50, 100$</td>
</tr>
</tbody>
</table>
### Integer solution results

<table>
<thead>
<tr>
<th>n</th>
<th>class</th>
<th>Q</th>
<th>inst</th>
<th>numbers</th>
<th>gap (%)</th>
<th>numbers</th>
<th>times (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>solved veh splits</td>
<td>lp</td>
<td>lp+cuts cuts nodes</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>R1</td>
<td>30</td>
<td>12</td>
<td>12 12.0 3.8</td>
<td>2.1</td>
<td>0.3</td>
<td>42.1+1.2 709.5</td>
</tr>
<tr>
<td>50</td>
<td></td>
<td></td>
<td>12</td>
<td>12 7.3 1.0</td>
<td>1.6</td>
<td>0.1</td>
<td>25.3+0.0 5.6</td>
</tr>
<tr>
<td>100</td>
<td></td>
<td></td>
<td>12</td>
<td>12 5.1 0.1</td>
<td>0.5</td>
<td>0.2</td>
<td>2.5+0.0 3.9</td>
</tr>
<tr>
<td>25</td>
<td>C1</td>
<td>30</td>
<td>9</td>
<td>4 16.0 5.0</td>
<td>2.9</td>
<td>0.3</td>
<td>58.3+23.5 15918.0</td>
</tr>
<tr>
<td>50</td>
<td></td>
<td></td>
<td>9</td>
<td>9 10.0 1.8</td>
<td>1.8</td>
<td>&lt;0.1</td>
<td>25.8+0.1 3.2</td>
</tr>
<tr>
<td>100</td>
<td></td>
<td></td>
<td>9</td>
<td>8 5.0 0.0</td>
<td>1.9</td>
<td>1.0</td>
<td>25.0+1.6 123.8</td>
</tr>
<tr>
<td>25</td>
<td>RC1</td>
<td>30</td>
<td>8</td>
<td>8 18.0 7.0</td>
<td>2.5</td>
<td>&lt;0.1</td>
<td>86.1+3.5 1422.5</td>
</tr>
<tr>
<td>100</td>
<td></td>
<td></td>
<td>8</td>
<td>8 6.0 0.4</td>
<td>0.8</td>
<td>&lt;0.1</td>
<td>11.6+0.0 1.3</td>
</tr>
<tr>
<td>25</td>
<td>R2</td>
<td>30</td>
<td>11</td>
<td>11 12.0 3.5</td>
<td>2.4</td>
<td>0.5</td>
<td>49.1+1.8 1220.6</td>
</tr>
<tr>
<td>50</td>
<td></td>
<td></td>
<td>11</td>
<td>11 7.0 1.0</td>
<td>1.5</td>
<td>0.1</td>
<td>21.6+0.0 17.0</td>
</tr>
<tr>
<td>100</td>
<td></td>
<td></td>
<td>11</td>
<td>9 4.0 0.3</td>
<td>1.6</td>
<td>0.8</td>
<td>8.5+0.7 57.7</td>
</tr>
<tr>
<td>25</td>
<td>C2</td>
<td>30</td>
<td>8</td>
<td>4 16.0 6.0</td>
<td>1.4</td>
<td>0.2</td>
<td>53.0+ 9.5 2563.5</td>
</tr>
<tr>
<td>50</td>
<td></td>
<td></td>
<td>8</td>
<td>3 10.0 2.0</td>
<td>1.3</td>
<td>0.4</td>
<td>30.3+13.3 4342.0</td>
</tr>
<tr>
<td>100</td>
<td></td>
<td></td>
<td>8</td>
<td>8 5.0 0.6</td>
<td>0.8</td>
<td>0.1</td>
<td>12.6+0.0 5.3</td>
</tr>
<tr>
<td>25</td>
<td>RC2</td>
<td>30</td>
<td>8</td>
<td>7 18.0 7.0</td>
<td>1.8</td>
<td>&lt;0.1</td>
<td>82.6+2.0 1055.8</td>
</tr>
<tr>
<td>100</td>
<td></td>
<td></td>
<td>8</td>
<td>8 6.0 0.4</td>
<td>0.8</td>
<td>&lt;0.1</td>
<td>11.5+0.0 1.9</td>
</tr>
</tbody>
</table>

Maximum CPU time = 1 hour, 2.8GHz PC
### Integer solution results (cont’d)

<table>
<thead>
<tr>
<th>n</th>
<th>class</th>
<th>Q</th>
<th>inst</th>
<th>Nb</th>
<th>numbers</th>
<th>gap (%)</th>
<th>numbers</th>
<th>times (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>solved</td>
<td>veh</td>
<td>splits</td>
<td>lp</td>
</tr>
<tr>
<td>50</td>
<td>R1</td>
<td>50</td>
<td>12</td>
<td></td>
<td>1</td>
<td>15.0</td>
<td>3.0</td>
<td>2.3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>100</td>
<td>12</td>
<td></td>
<td>5</td>
<td>9.8</td>
<td>0.4</td>
<td>1.2</td>
</tr>
<tr>
<td>50</td>
<td>RC1</td>
<td>50</td>
<td>8</td>
<td></td>
<td>8</td>
<td>20.0</td>
<td>3.9</td>
<td>0.6</td>
</tr>
<tr>
<td></td>
<td></td>
<td>100</td>
<td>8</td>
<td></td>
<td>8</td>
<td>10.0</td>
<td>0.6</td>
<td>0.9</td>
</tr>
<tr>
<td>50</td>
<td>R2</td>
<td>100</td>
<td>11</td>
<td></td>
<td>1</td>
<td>8.0</td>
<td>1.0</td>
<td>0.8</td>
</tr>
<tr>
<td>50</td>
<td>C2</td>
<td>50</td>
<td>8</td>
<td></td>
<td>1</td>
<td>18.0</td>
<td>6.0</td>
<td>1.3</td>
</tr>
<tr>
<td>50</td>
<td>RC2</td>
<td>50</td>
<td>8</td>
<td></td>
<td>8</td>
<td>20.0</td>
<td>4.9</td>
<td>0.6</td>
</tr>
<tr>
<td></td>
<td></td>
<td>100</td>
<td>8</td>
<td></td>
<td>8</td>
<td>10.0</td>
<td>0.8</td>
<td>1.0</td>
</tr>
<tr>
<td>100</td>
<td>R1</td>
<td>100</td>
<td>12</td>
<td></td>
<td>1</td>
<td>20.0</td>
<td>0.0</td>
<td>0.1</td>
</tr>
</tbody>
</table>

**Maximum CPU time = 1 hour, 2.8GHz PC**
Remarks

- Gap decreases with capacity $Q$ (need split deliveries)
- Cycling increases with capacity $Q$
- Large gaps for C1 and C2 instances

- $k$-path inequalities are useful
- Arc-flow inequalities are not useful

- Gendreau et al. (2006) solved 27 instances (1.6GHz PC)
- We solved 175 instances (2.8GHz PC)
Outline

1 Introduction
   - Problem definition
   - Literature review
   - Main difficulty with branch-and-price for SDVRPTW

2 Formulation
   - Master problem
   - Subproblem

3 Branch-and-price-and-cut method
   - Column generation
   - Cutting and branching

4 Computational results

5 Conclusions
Conclusions

In this paper
- Novel decomposition
- New subproblem type
- New label-setting algorithm
- Relatively good results

We can do better
- Accelerating strategies for solving the subproblem
- Heuristics for solving the subproblem
- Other valid inequalities to reduce gaps