Stochastic Vehicle Routing: an Overview and an Exact Solution Approach for the Vehicle Routing Problem with Stochastic Demands under an Optimal Restocking Recourse Policy

Michel Gendreau
CIRRELT and MAGI
École Polytechnique de Montréal

DII - Dipartimento di Ingegneria dell'Informazione
Università degli Studi di Brescia
Brescia – October 30, 2018

Acknowledgements

- Ola Jabali (CIRRELT and Politecnico di Milano)
- Walter Rei (CIRRELT and UQÀM)
- Majid Salavati-Khoshghalb (CIRRELT and U. de Montréal)
- Gilbert Laporte (CIRRELT and HEC Montréal)
- Charles Gauvin (CIRRELT and École Polytechnique)
- Guy Desaulniers (GERAD and École Polytechnique)

Outline

1. Introduction
2. Basic concepts in stochastic optimization
3. Modeling paradigms
4. VRP with stochastic demands
5. Some other recourse policies for the VRPSD
6. An exact algorithm to solve the VRPSD under an optimal restocking policy
7. Conclusion and perspectives

Introduction

Vehicle Routing Problems

- Introduced by Dantzig and Ramser in 1959
- One of the most studied problem in the area of logistics
- The basic problem involves delivering given quantities of some product to a given set of customers using a fleet of vehicles with limited capacities.
- The objective is to determine a set of minimum-cost routes to satisfy customer demands.

Vehicle Routing Problems

Many variants involving different constraints or parameters:

- Introduction of travel and service times with route duration or time window constraints
- Multiple depots
- Multiple types of vehicles
- …

What is Stochastic Vehicle Routing?

Basically, any vehicle routing problem in which one or several of the parameters are not deterministic:

- Demands
- Travel or service times
- Presence of customers
- …

Main classes of stochastic VRPs

- VRP with stochastic demands (VRPSD)
  - A probability distribution is specified for the demand of each customer.
  - One usually assumes that demands are independent (this may not always be very realistic...).
- VRP with stochastic customers (VRPSC)
  - Each customer has a given probability of requiring a visit.
- VRP with stochastic travel times (VRPSTT)
  - The travel times required to move between vertices, as well as sometimes service times, are random variables.

Basic Concepts in Stochastic Optimization

Dealing with uncertainty in optimization

- Very early in the development of operations research, some top contributors realized that :
  - In many problems there is very significant uncertainty in key parameters;
  - This uncertainty must be dealt with explicitly.
- This led to the development of :
  - Stochastic programming with recourse (1955)
  - Dynamic programming (1958)
  - Chance-constrained programming (1959)
  - Robust optimization (more recently)

Information and decision-making

In any stochastic optimization problem, a key issue is:

- How do the revelation of information on the uncertain parameters and decision-making (optimization) interact?
  - When do the values taken by the uncertain parameters become known?
  - What changes can I (must I) make in my plans on the basis of new information that I obtain?

Stochastic programming with recourse

- Proposed separately by Dantzig and by Beale in 1955.
- The key idea is to divide problems in different stages, between which information is revealed.
- The simplest case is with only two stages. The second stage deals with **recourse actions**, which are undertaken to adapt plans to the realization of uncertainty.
- Basic reference:

  J.R. Birge and F. Louveaux, Introduction to Stochastic Programming, 2nd edition, Springer, 2011.

Dynamic programming

- Proposed by Bellman in 1958.
- A method developed to tackle effectively sequential decision problems.
- The solution method relies on a time decomposition of the problem according to stages. It exploits the so-called *Principle of Optimality*.
- Good for problems with limited number of possible **states** and **actions**.
- Basic reference:

  D.P. Bertsekas, Dynamic Programming and Optimal Control, 3rd edition, Athena Scientific, 2005.

Chance-constrained programming

- Proposed by Charnes and Cooper in 1959.
- The key idea is to allow some constraints to be satisfied only with some probability.

  E.g., in VRP with stochastic demands,

  Pr{total demand assigned to route *r* ≤ *capacity* } ≥ 1-α

Robust optimization

- Here, uncertainty is represented by the fact that the uncertain parameter vector must belong to a given polyhedral set (without any probability defined)
  - E.g., in VRP with stochastic demands,

    having set upper and lower bounds for each demand, together with an upper bound on total demand.
- Robust optimization looks in a minimax fashion for the solution that provides the best “worst case”.

Modelling Paradigms

Real-time optimization

Also called re-optimization

- Based on the implicit assumption that information is revealed over time as the vehicles perform their assigned routes.
- Relies on Dynamic programming and related approaches (Secomandi et al.)
- Routes are created piece by piece on the basis on the information currently available.
- Not always practical (e.g., recurrent situations)

A priori optimization

- A solution must be determined beforehand; this solution is “confronted” to the realization of the stochastic parameters in a second step.
- Approaches:
  - Chance-constrained programming
  - (Two-stage) stochastic programming with recourse
  - Robust optimization
  - [“Ad hoc” approaches]

Chance-constrained programming

- Probabilistic constraints can sometimes be transformed into deterministic ones (e.g., in in VRP with stochastic demands, when one imposes that

  Pr{total demand assigned to route *r* ≤ *cap.* } ≥ 1-α,

  if customer demands are independent and Poisson).

- This model completely ignores what happens when things do not "turn out correctly".

Robust optimization

- Not used very much in stochastic VRP up to now.
- Model may be overly pessimistic.

Stochastic programming with recourse

- **Recourse** is a key concept in a priori optimization
  - What must be done to "adjust" the a priori solution to the values observed for the stochastic parameters!
  - Another key issue is deciding when information on the uncertain parameters is provided to decision-makers.
- Solution methods:
  - Integer L-shaped (Laporte and Louveaux)
  - Column generation (Branch & Price)
  - Heuristics (including metaheuristics)
- Probably closer to actual industrial practices, if recourse actions are correctly defined!

VRP with Stochastic Demands

VRP with stochastic demands

- Probably, the most studied stochastic vehicle routing problem.
- A variant of the well-known CVRP in which the customer demands are not fixed quantities, but instead random variables with given distributions.
- Building solutions in which capacity constraints would be satisfied for all possible realizations of the customer demands is clearly too conservative.

VRP with stochastic demands (2)

- We must consider solutions in which the total demand of customers assigned to a given route can possibly exceed the capacity of a vehicle.
- What should be done if this happens ?!?
- We must engage in some corrective or RECOURSE action.

Classical recourse

- It is important to understand that recourse actions should reflect operating policies of the company operating the fleet of vehicles.
- The *classical recourse strategy* used in most of the VRPSD literature consists of returning to the depot to restore the vehicle capacity and then resuming the planned route from the point of *failure*.
- This strategy allows for an “easy” computation of the expected recourse associated with any given route.

Other possible recourses

- The classical recourse strategy is not very bright. It is even shocking to some people who are not used to it.
- Yang, Mathur, and Ballou (2000) have proposed a policy of *optimal restocking* of the vehicle based on dynamic programming, but it can only be applied to given routes.
- Ak and Erera (2007) have suggested "pairing" to enhance operations and reduce the number of back and forth trips back to the depot.

VRP with stochastic demands

- Approximate solutions can be obtained fairly easily using metaheuristics
(e.g., Tabu Search, as in Gendreau et al., 1996).

- Computing effectively the value of the recourse function still remains a challenge.

A branch-and-cut approach (direct formulation)

The following material is taken from

- O. Jabali, W. Rei, M. Gendreau, G. Laporte (2014). New Valid Inequalities for the Multi-Vehicle Routing Problem with Stochastic Demands. *Discrete Applied Mathematics,* 177, 121-136.

Model

Input

$G(V,E)$ = undirected graph, $V = \{v_1, \ldots, v_n\}$ and $E = \{(v_i, v_j)\colon v_i, v_j \in V, i < j\}$
ξ_i = demand of client i, where $i = 2, \ldots, n$
$C = (c_{ij})$ travel cost matrix
D = capacity of each vehicle

Decision variables

$$\begin{array}{lll} x_{ij} = \{0,1\} & \text{for } i,j > 1 \\ x_{1j} = \{0,1,2\} & \text{for } \quad j > 1 \end{array}$$

The considered case

Client demands are independent
$\xi_j \sim N(\mu_j, \sigma_j)$ and $\xi_j \in (0, D)$, $j = 2, \ldots, n$
Recourse rules $\Rightarrow$ return to depot only when failure occurs
For a given route $(v_{r_1} = v_1, v_{r_2}, \ldots, v_{r_{t+1}} = v_1)$

$$\begin{aligned} \mathcal{Q}^{1,r} &= 2\sum_{i=2}^{t}\sum_{l=1}^{i-1} P\left(\sum_{s=2}^{i-1}\xi_{r_s} \leq lD < \sum_{s=2}^{i}\xi_{r_s}\right) c_{1r_i} \\ \mathcal{Q}(x) &= \sum_{i=1}^{m} \min\{\mathcal{Q}^{k,1}, \mathcal{Q}^{k,2}\}, \end{aligned}$$

Model

$$\text{(VRPSD)} \qquad \text{Minimize} \sum_{i<j} c_{ij} x_{ij} + \mathcal{Q}(x)$$

subject to

$$\sum_{j=2}^{n} x_{1j} = 2m,$$

$$\sum_{i<k} x_{ik} + \sum_{j>k} x_{kj} = 2, \qquad (k = 2, \ldots, n),$$

$$\sum_{v_i, v_j \in S} x_{ij} \leq |S| - \left\lceil \sum_{v_i \in S} \mathbf{E}(\xi_i)/D \right\rceil, \qquad S \subset V \setminus \{v_1\}, 2 \leq |S| \leq n-2$$

$$0 \leq x_{ij} \leq 1 \qquad 1 \leq i < j < n),$$

$$0 \leq x_{0j} \leq 2 \qquad (j = 2, \ldots, n),$$

$$x = (x_{ij}) \qquad \text{integer.}$$

A branch-and-cut approach

- Computational results on problems with independent truncated Normal demands
  - 30 instances for each combination of *m* and *n*
  - 10 hours of CPU time for each run.

n	m	Solved	Runtime (min)	Gap
60	2	19	227	0.2%
70	2	14	372	0.4%
80	2	14	366	0.5%
50	3	9	458	0.8%
60	3	8	473	1.1%
70	3	11	419	1.6%
40	4	5	519	2.5%
50	4	4	529	3.9%
60	4	3	552	2.9%

A branch-and-price approach (set covering formulation)

The following material is taken from

- C. Gauvin, G. Desaulniers, M. Gendreau (2014). A Branch-Cut-and-Price Algorithm for the Vehicle Routing Problem with Stochastic Demands, *Computers & Operations Research*, 50, 141-153.

Litterature

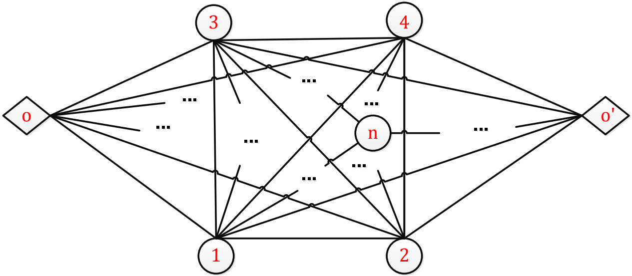
- **Heuristics**
  - Tillman (1965) - Savings-based heuristics for multi-depot and Poisson demands
  - Golden, Stewart (1983)
  - Gendreau, Séguin (1996) - Tabou search
- **Integer L-shaped method for problems with simple recourse**
  - Laporte, Louveaux (1993)
  - Gendreau, Laporte, Séguin (1995)
  - Hjorring, Holt (1999)
  - Laporte, Louveaux, Van Hamme (2001)
- **Column Generation**
  - Christiansen, Lysgaard (2007) Branch-and-price

Our Contribution

Develop a competitive branch-cut-and-price algorithm for the VRPSD based on the work of Christiansen et Lysgaard (2007).

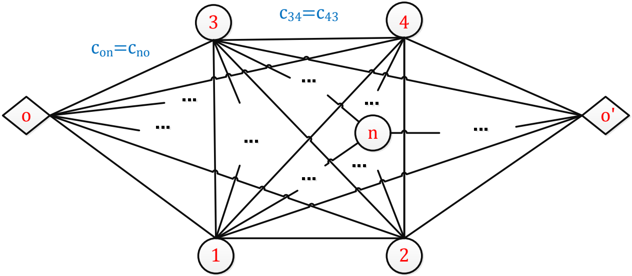
Notation

- $\mathcal{G} = (\mathcal{N}' = \mathcal{N} \cup \{o, o'\}, \mathcal{A})$: undirected graph
- $\mathcal{N} = \{1, 2, ..., n\}$: set of clients
- $\mathcal{A} = \{(i,j) | i,j \in \mathcal{N}; i \neq j\} \cup \{(o,j) | j \in V\} \cup \{(j,o') | j \in \mathcal{N}\}$: set of arcs



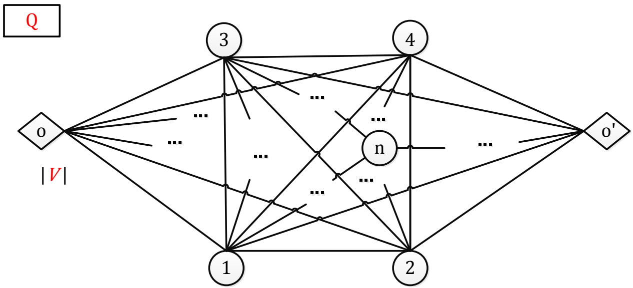
Notation

- c_{ij} : determinist travel cost from $i \in \mathcal{N}'$ à $j \in \mathcal{N}'$
- $p = (i_1, ..., i_h)$: route where $i_j \in \mathcal{N}'$ for $j \in \{1, ..., h\}$
- c_p : determinist travel cost of route p
- $\hat{c}_p$: total expected failure cost of route p
- α_{ip} : binary parameter indicating if route p visits client $i \in \mathcal{N}$ or not
- $\mathscr{P}$: set of all routes from o to o' which are feasible on average



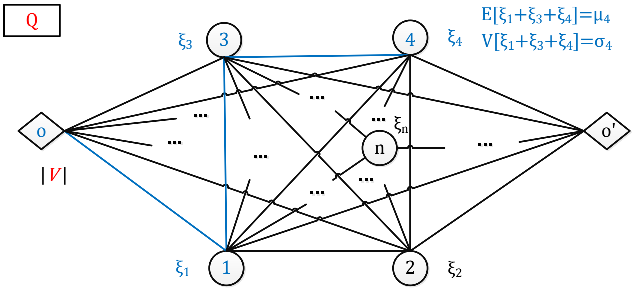
Notation

- $\mathcal{V}$: set of $|\mathcal{V}|$ identical vehicles
- Q : capacity of a vehicle



Notation

- $\xi_i \sim \Psi(\mathbb{E}_\xi[\xi_i], \mathbb{V}_\xi[\xi_i])$: random variable indicating demand of client $i \in \mathcal{N}$ following distribution Ψ ;
- Given path $p = (i_1, i_2, ...i_h)$
  - $\sum_{j=1}^{h} \mathbb{E}_\xi\left[\xi_{i_j}\right] = \mu_{i_h}$: expected cumulated demand at i_h
  - $\sum_{j=1}^{h} \mathbb{V}_\xi\left[\xi_{i_j}\right] = \sigma_{i_h}$: expected cumulated variance at i_h



Notation - decision variables

First stage

- x_{ij} : binary variable indicating if a vehicle follows the arc $(i, j) \in \mathcal{A}$.
- λ_p : binary variable indicating if we choose route p or not

Second stage

- $\mathcal{Q}(x)$: recourse function (expected failure cost of a given route)

Master Problem and Subproblem - VRPSD

$$
\begin{aligned}
\min_{\lambda} \quad & \sum_{p \in \mathscr{P}} \hat{c}_p \lambda_p \\
\text{s. t. :} \quad & \sum_{p \in \mathscr{P}} \alpha_{ip} \lambda_p = 1 \forall i \in \mathcal{N} \\
& \lambda_p \in \{0,1\} \quad \forall p \in \mathscr{P}
\end{aligned}
\qquad (MP)
$$

$$
\begin{aligned}
\min_{x} \quad & \sum_{i \in \mathcal{N}'} \sum_{j \in \mathcal{N}'} \bar{c}_{ij} x_{ij} + \mathcal{Q}(x) \\
\text{s. à :} \quad & \sum_{(i,j) \in \delta^-(\{j\})} x_{ij} - \sum_{(j,i) \in \delta^+(\{j\})} x_{ji} = \begin{cases} -1 & \text{if } j = o \\ 0 & \text{else } \forall j \in \mathcal{N}' \\ 1 & \text{if } j = o' \end{cases} \\
& \sum_{i \in \mathcal{N}'} \sum_{j \in \mathcal{N}'} \mathbb{E}_{\xi}\left[\xi_j\right] x_{ij} \leq Q \\
& x_{ij} \in \{0,1\} \quad \forall i,j \in \mathcal{N}'
\end{aligned}
\qquad (SP)
$$

Reduced cost

$$\bar{c}_{ij} = c_{ij} - \pi_j \text{ if } j \neq o'$$

Computation of $\mathcal{Q}(x)$

Given

- $p = (i_1, i_2, \ldots i_{h-1}, i_h) \subseteq \mathcal{N}'$ from $o = i_1$ to i_h

[1]Expected failure cost at client i_h for p

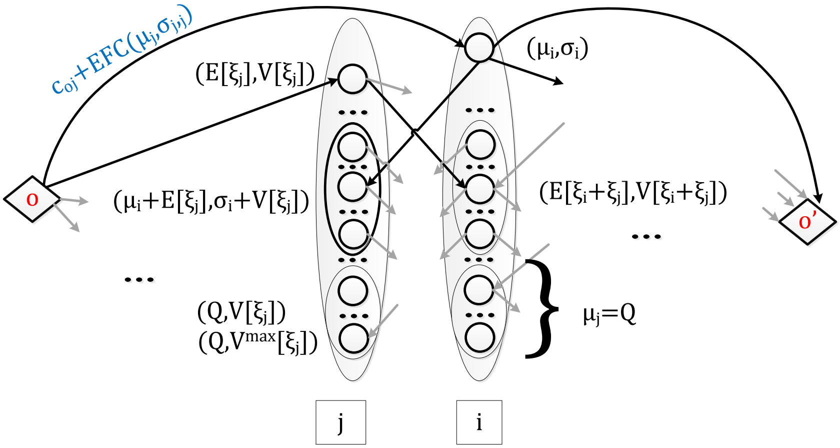
$$\mathrm{EFC}(\mu_{i_h}, \sigma_{i_h}, i_h) = 2c_{oi_h} \sum_{u=1}^{\infty} (\mathbb{P}_\xi \left[\sum_{l=1}^{h-1} \xi_{i_l} \leq uQ < \sum_{l=1}^{h} \xi_{i_l} \right])$$

1-Dror & Trudeau (1987), Laporte et al. (2002), Christiansen & Lysgaard (2007)

Total cost of route p= determinist cost p + expected failure cost p

$$\hat{c}_p = \sum_{j=1}^{h} \left(c_{i_h i_{h+1}} + \mathrm{EFC}(\mu_{i_{h+1}}, \sigma_{i_{h+1}}, i_{h+1}) \right)$$

Creation of the state-space graph $\mathcal{GS} = (\mathcal{NS}, \mathcal{AS})$



Numerical results

- Max alloted time : 20 minutes,
- 3,4 GHz and 16 GB of RAM vs 1,5GHz and 480 MB of RAM $\Rightarrow$ scaling factor : 1.98
- Poisson demands + A, E, P instances

Instance	Base (This work)						Christiansen & Lysgaard (2007)			Acceleration factor
	Time	$\frac{\text{LB root}}{\text{UB}}$	# B & B Nodes	# Rounds of cuts	# Cuts CC	# Cuts SRI	Time	$\frac{\text{LB root}}{\text{UB}}$	# B & B Nodes	
A-n32-k5	24.1	0.98	0	7	6	90	142.4	0.96	2467	5.9
A-n33-k5	5.3	1	0	6	29	15	4	0.99	117	0.8
A-n33-k6	4.9	1	0	4	3	20	24.7	0.98	909	5.1
A-n34-k5	9.0	0.98	0	6	27	28	#	0.97	16059	$\geq$ 67.3
A-n36-k5	46.9	0.98	0	10	30	90	#	0.92	8035	$\geq$12.9
A-n37-k5	23.4	0.98	0	7	12	72	#	0.97	9191	$\geq$25.9
A-n37-k6	22.8	0.99	2	11	56	62	#	0.98	16195	$\geq$26.6
A-n38-k5	42.8	0.97	6	11	22	74	#	0.95	14499	$\geq$14.2
A-n39-k5	5.0	1	0	0	0	0	1.5	1	9	0.3
A-n39-k6	19.3	0.99	0	8	28	45	140.9	0.97	2431	7.3
A-n44-k6	125.0	0.99	12	21	21	189	#	0.98	11077	$\geq$4.9
A-n45-k6	86.3	0.98	2	11	9	120	#	0.9	9313	$\geq$7.0
A-n45-k7	30.5	1	0	9	8	70	445.5	0.99	5365	14.6
A-n46-k7	18.7	0.99	0	4	58	30	#	0.98	8149	$\geq$32.4
A-n48-k7	27.5	0.99	0	6	42	30	#	0.93	7729	$\geq$22.0
A-n53-k7	512.4	0.99	12	23	37	240	#	0.93	5385	$\geq$1.2

Numerical results (cont'd)

Instance	Base (This work)						Christiansen & Lysgaard (2007)			Acceleration factor
	Time	$\frac{\text{LB root}}{\text{UB}}$	# B & B Nodes	# Rounds of cuts	# Cuts CC	# Cuts SRI	Time	$\frac{\text{LB root}}{\text{UB}}$	# B & B Nodes	
A-n54-k7	310.5	0.99	6	18	55	126	#	0.94	5925	≥2.0
A-n55-k9	33.2	0.99	0	7	50	45	#	0.91	9979	≥18.3
A-n60-k9	#	0.42	20	35	49	219	#	0.93	6889	#
E-n22-k4	0.1	1	0	0	0	0	0.5	1	9	5
E-n33-k4	32.6	1	0	0	0	0	43.4	0.99	73	1.3
E-n51-k5	#	0.95	4	29	23	212	#	0.97	2771	#
P-n16-k8	0.0	1	0	1	1	0	0	1	17	#
P-n19-k2	9.7	0.94	0	7	4	60	77.3	0.94	1815	8.0
P-n20-k2	128.8	0.95	6	15	5	130	177.8	0.95	3191	1.4
P-n21-k2	2.2	1	0	1	3	0	2.5	0.99	27	1.1
P-n22-k2	46.7	0.97	0	8	3	90	110.6	0.97	1335	2.4
P-n22-k8	0.0	1	0	2	1	5	0	1	65	#
P-n23-k8	0.0	1	0	0	0	0	0.5	1	0	#
P-n40-k5	6.9	1	0	1	4	0	4.5	1	35	0.7
P-n45-k5	1193.8	0.98	12	34	22	272	#	0.95	4561	#
P-n50-k10	54.6	0.99	16	28	29	119	#	0.95	18901	≥11.1
P-n50-k7	40.4	0.99	0	11	26	75	#	0.95	5311	≥15.0
P-n50-k8	35.7	0.99	2	11	9	98	#	0.91	10409	≥17.0
P-n51-k10	10.0	0.99	0	8	21	56	217.2	0.99	5431	21.7
P-n55-k10	26.4	0.99	0	10	23	60	#	0.92	11257	≥23.0
P-n55-k15	19.9	1	36	32	23	61	400	0.99	20027	20.1
P-n55-k7	103.8	0.99	0	14	22	120	#	0.94	2441	≥5.8
P-n60-k10	417.5	0.99	34	52	42	273	#	0.95	4969	≥1.5
P-n60-k15	7.0	1	0	7	37	26	#	1	19029	≥86.6
Average	91.7	0.97	4.25	11.9	21	80.55	99.6	0.96	6284.93	13.6

A robust optimization approach

- C.E. Gounaris, W. Wiesemann, C.A. Floudas (2013). The Robust Capacitated Vehicle Routing Problem Under Demand Uncertainty. *Operations Research*, 61 (3), 677-693.

- The authors consider the generic case where the customer demands are supported on a polyhedron.
- Robust solutions must be feasible for all demand vectors in the polyhedron.
- Several interesting references.

Some other Recourse Policies for the VRPSD

Other possible recourses

- One can easily imagine other recourse policies, which would be easily implementable in practice, and which would make more sense.
- This is what we did in the two first papers of Majid Salavati-Khosgalb's dissertation:
  - SALAVATI-KHOSHGALB, M., GENDREAU, M., JABALI, O., REI, W., "A Rule-Based Recourse for the Vehicle Routing Problem with Stochastic Demands", *Transportation Science.* Forthcoming.
  - SALAVATI-KHOSHGALB, M., GENDREAU, M., JABALI, O., REI, W., "A Hybrid Recourse Policy for the Vehicle Routing Problem with Stochastic Demands", *EURO Journal on Transportation and Logistics.* Forthcoming.

Fixed-policy recourses

- Each fixed policy can be derived from a given operational convention.
- It generates related policy-based recourse actions, which are defined as a set of thresholds associated with the customer visits that are scheduled along a given route.
- These thresholds determine when the vehicle performing the route should preventively return to the depot.

Four classes of pro-active policies

- *Demand-based policies,* in which thresholds relate directly to the demand levels of customers or the capacity of the vehicle.
- *Risk-based policies,* in which thresholds are computed on the basis of the probability of failure at the next or at following customers.
- *Distance-based policies* also account for the distance between customers and the depot.
- *Hybrid* (or *mixed*) *policies* combine features of two or three of the other classes.

Three demand-based policies

- The αQ *policy* is defined in terms of the vehicle capacity *Q:* PR trips occur whenever the residual capacity falls below a fraction α *of Q.*
  - Three values are examined for α: 0.05, 0.10, and 0.20. In this case, all thresholds are equal.
- In the second policy considered, the threshold θ_{ij} is a function of the expected demand of the following customer v_{ij+1}.
  - Three values are considered for β = 0.80, 1.00, 1.25.
- The third policy is based on the expected demand of the customers remaining on the route after v_{ij}.

A mixed policy

- We consider a *mixed policy* that combines the basic structure of a *risk-based policy* with a *distance-based policy.*
- After serving each customer, the risk of failure at the next customer is computed,
  - If the risk is ***high***: perform a preventive restocking trip;
  - If the risk is ***low***: proceed;
  - If the risk is ***medium***: apply a distance-based policy.
- The distance-policy criterion is to compare the cost of a preventive restocking trip at the current location vs. the expected cost along the remainder of the route.

Solution approach

- We consider an exact solution approach based on the Integer L-Shaped method of Laporte and Louveaux.
- The algorithm is coded in C++.
- The branching routine is implemented by using the OOBB package coded at the CIRRELT.
- Subtour elimination and stochastic capacity constraints are separated using the CVRPSEP package of Lysgaard et al. (2004).

Solution approach (2)

- To enhance running times, we use different families of lower bounding functionals (valid inequalities) pertaining to *partial routes*, as in Jabali et al. (2014).
- The lower bounding functionals derive from the idea originally proposed by Hjorring and Holt (1999)
- Generalizing these lower bounding functionals to deal with the new recourse policies is definitely **non-trivial**!!!

An Exact Algorithm to Solve the Vehicle Routing Problem with Stochastic Demands under an Optimal Restocking Policy

M. Gendreau[1], M. Salavati[2], O. Jabali[3], W. Rei[4]

(1) : CIRRELT and MAGI, École Polytechnique de Montréal

(2) : CIRRELT and DIRO, Université de Montréal

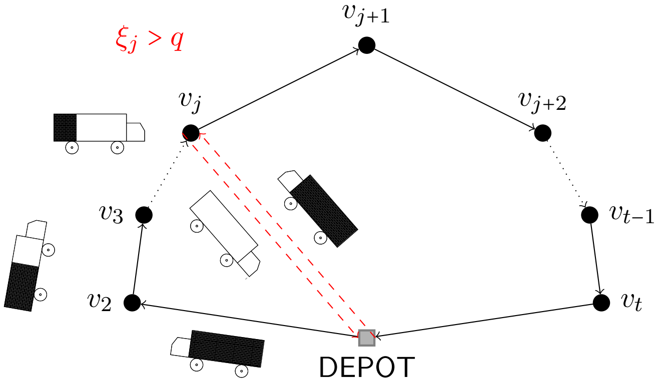
(3) : CIRRELT and DEIB, Politecnico di Milano

(4) : CIRRELT and ESG, Université du Québec à Montréal

Recourse Policy: Traditional Recourse

Traditional Recourse Policy

Follow the planned route.
Execute a BF trip whenever route failures observed.

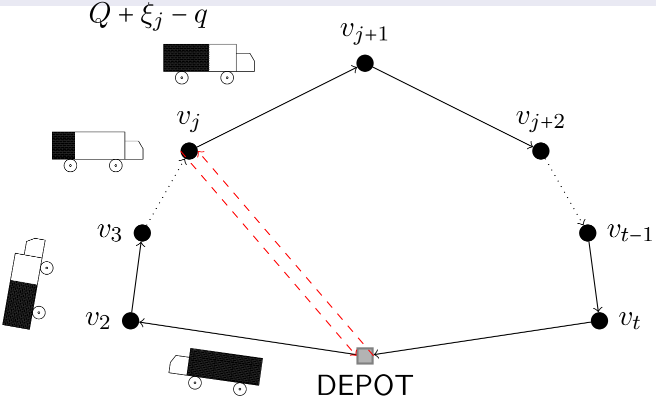


Recourse Policy: Traditional Recourse

Traditional Recourse Policy

Follow the planned route.

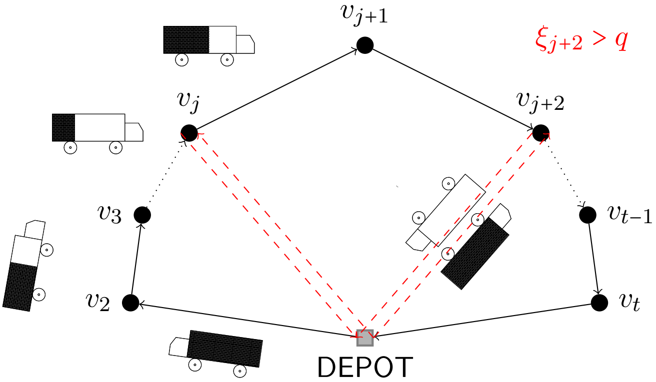
Execute a BF trip whenever route failures observed.



Recourse Policy: Traditional Recourse

Traditional Recourse Policy

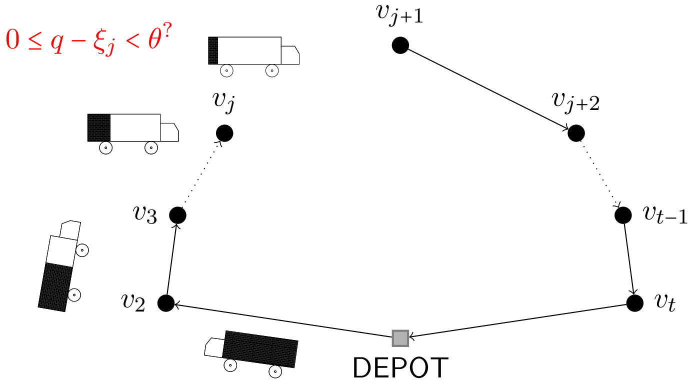
Follow the planned route.
Execute a BF trip whenever route failures observed.



A Rule-Based Recourse for VRPSD

Rule-Based Recourse Policy

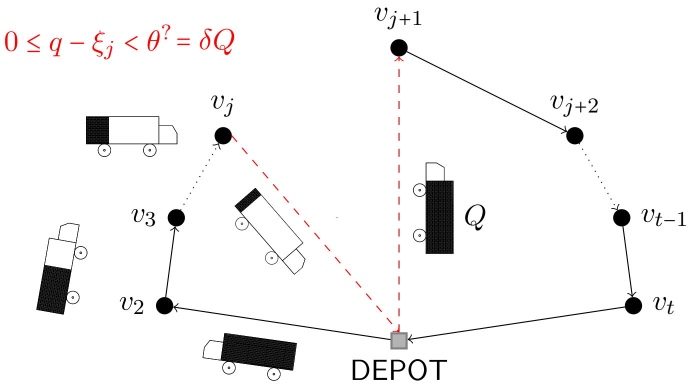
Follow the planned route. Execute BF trips when route failure occurred. Execute PR trips based on fixed thresholds.



A Rule-Based Recourse for VRPSD

Rule-Based Recourse Policy

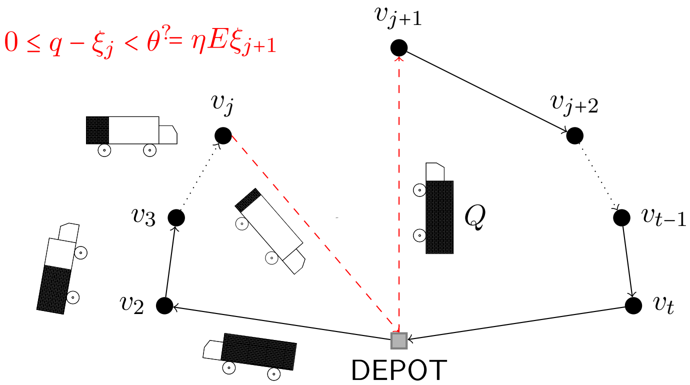
Follow the planned route. Execute BF trips when route failure occurred. Execute PR trips based on fixed thresholds.



A Rule-Based Recourse for VRPSD

Rule-Based Recourse Policy

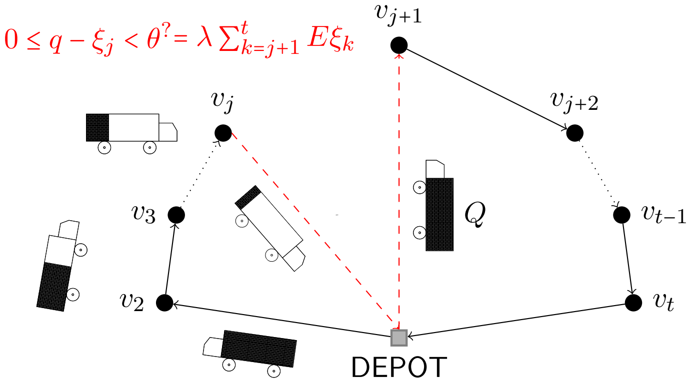
Follow the planned route. Execute BF trips when route failure occurred. Execute PR trips based on fixed thresholds.



A Rule-Based Recourse for VRPSD

Rule-Based Recourse Policy

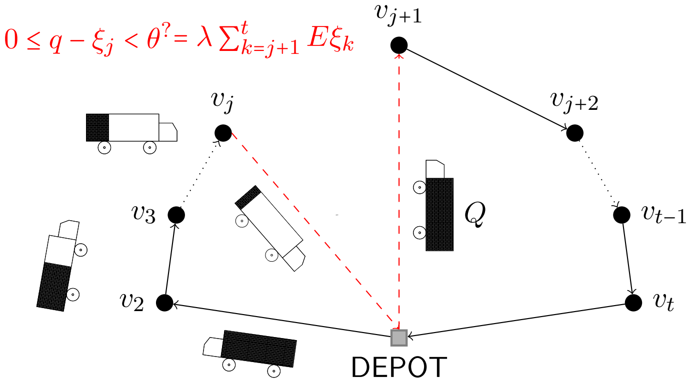
Follow the planned route. Execute BF trips when route failure occurred. Execute PR trips based on fixed thresholds.



A Rule-Based Recourse for VRPSD

Rule-Based Recourse Policy

Follow the planned route. Execute BF trips when route failure occurred. Execute PR trips based on fixed thresholds.



Recourse Policy:

Rule-Based Recourse Policy

Given planned route i as $\vec{v} = (v_1 = v_{i_1}, \ldots, v_{i_j}, v_{i_{j+1}}, \ldots, v_{i_t}, v_{i_{t+1}} = v_1)$.
Given threshold $\vec{\theta} = (\theta_{i_2}, \ldots, \theta_{i_t})$ for route $\vec{v}$.
We define Recourse Function $F_{i_j}(q)$ as the expected recourse cost starting from vertex v_{i_j} with q units of residual capacity:

Recourse Function for Computing Expected Recourse Cost

$$F_{i_j}(q) = \begin{cases} \sum_{s:\xi^s_{i_j} > q} \Big(b + 2c_{1i_j} + F_{i_j+1}(Q + q - \xi^s_{i_j}) \Big) p^s_{i_j} + \\ \sum_{s:q-\theta_{i_j} < \xi^s_{i_j} \le q} \Big(c_{1i_j} + c_{1i_j+1} - c_{i_j i_j+1} + F_{i_j+1}(Q) \Big) p^s_{i_j} + \\ \sum_{s:\xi^s_{i_j} \le q-\theta_{i_j}} F_{i_j+1}(q - \xi^s_{i_j}) p^s_{i_j} & \text{if } j = 2, \ldots, t \end{cases} \tag{1}$$

$$Q^{\vec{v},1} = F_{i_1}(Q), \;\; Q(\vec{v}) = \min\{Q^{\vec{v},1}, Q^{\vec{v},2}\} \tag{2}$$

Literature Review

Literature Review

Volume-based Recourse Policies:

1. Traditional Recourse proposed by Dror and Trudeau (1986).
2. Optimal Restocking Policy proposed by Yee and Golden (1980).
3. A Rule-Based Recourse proposed by Salavati-Khoshghalb, Gendreau, Jabali, and Rei (2017).

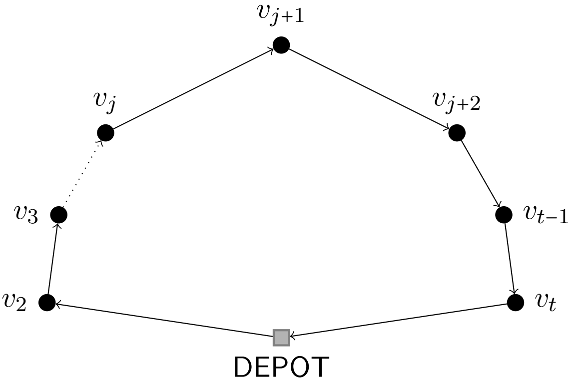
Risk-and-Distance-based Recourse Policy:

4. A Hybrid Recourse proposed by Salavati-Khoshghalb, Gendreau, Jabali, and Rei (2017).

Optimal Restocking Policy

How it works?

Compute proceeding cost-to-go and replenishment cost-to-go.
Choose optimal action by choosing the action which incurs the min. cost.

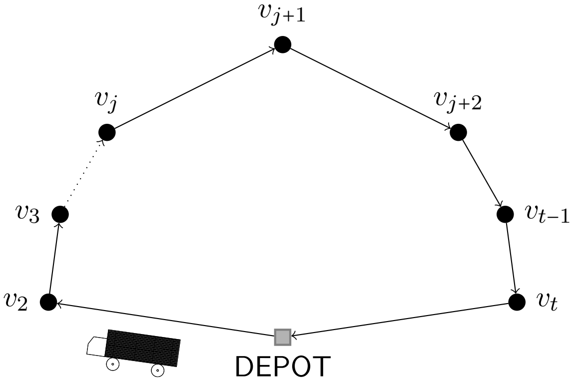


Optimal Restocking Policy

How it works?

Compute proceeding cost-to-go and replenishment cost-to-go.

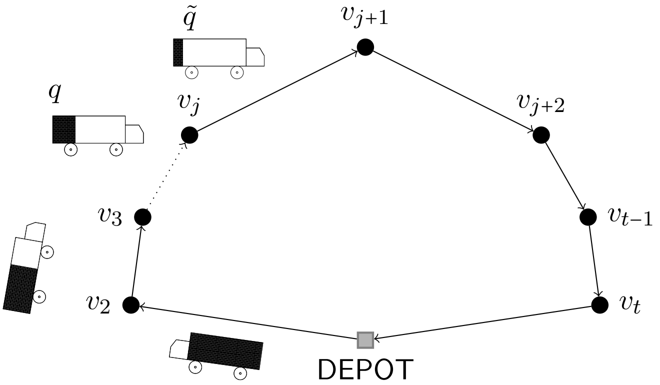
Choose optimal action by choosing the action which incurs the min. cost.



Optimal Restocking Policy

How it works?

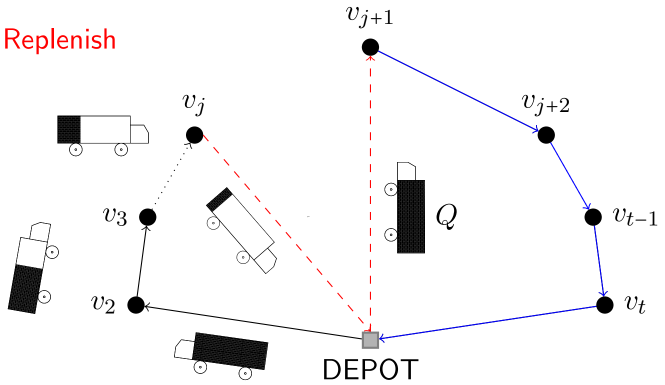
Compute proceeding cost-to-go and replenishment cost-to-go.
Choose optimal action by choosing the action which incurs the min. cost.



Optimal Restocking Policy

How it works?

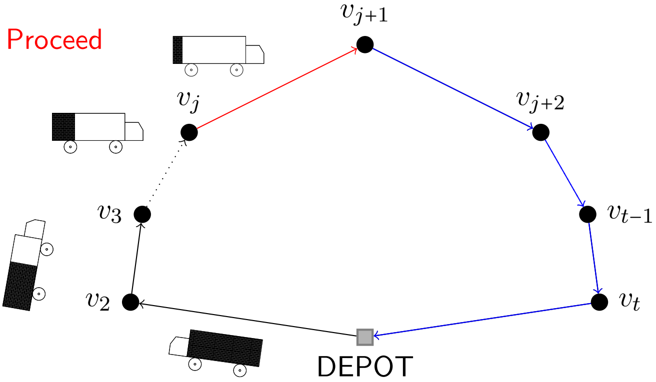
Compute proceeding cost-to-go and replenishment cost-to-go.
Choose optimal action by choosing the action which incurs the min. cost.



Optimal Restocking Policy

How it works?

Compute proceeding cost-to-go and replenishment cost-to-go.
Choose optimal action by choosing the action which incurs the min. cost.



Recourse Policy: Optimal Policy

Optimal threshold $\theta^*_{i_j}$?

By comparing proceed and replenish we can compute optimal thresholds as follows:

$$\theta^*_{i_j} = \min_{0,\ldots,Q}\{\tilde{q} | \text{proceed} \leq \text{replenish}\}, \text{ then } \begin{cases} \text{PR trip} & \text{if } \tilde{q} < \theta^*_{i_j} \\ \text{proceed} & \text{otherwise} \end{cases}$$

$$F_{i_j}(\tilde{q}) = \min \begin{cases} \text{proceed} : c_{i_j, i_{j+1}} + \sum_{k:\xi^k_{i_{j+1}} \leq \tilde{q}} F_{i_{j+1}}(\tilde{q} - \xi^k_{i_{j+1}}) p^k_{i_{j+1}} + \\ \qquad \sum_{k:\xi^k_{i_{j+1}} > \tilde{q}} [b + 2c_{1,i_{j+1}} + F_{i_{j+1}}(Q + \tilde{q} - \xi^k_{i_{j+1}})] p^k_{i_{j+1}}, \\ \text{replenish} : c_{1,i_j} + c_{1,i_{j+1}} + \sum_{k=1}^{s} F_{i_{j+1}}(Q - \xi^k_{i_{j+1}}) p^k_{i_{j+1}}. \end{cases}$$

Exact Algorithms to Tackle VRPSD

Integer L-shaped algorithm

Integer L-shaped algorithm as a general B&C framework is proposed by Laporte and Louveaux (1993).
Gendreau et al. (1995), Laporte et al. (2002), and Jabali et al. (2014)...

Column Generation

B&P: Christiansen and Lysgaard (2004).
B&C&P: Gauvin et al. (2014).

Two-stage Stochastic Integer Program with Recourse

$$\underset{x}{\text{minimize}} \quad \sum_{i<j} c_{ij} x_{ij} + Q(x) \tag{3}$$

$$\text{subject to} \quad \sum_{j=2}^{n} x_{1j} = 2m, \tag{4}$$

$$\sum_{i<k} x_{ik} + \sum_{k<j} x_{kj} = 2, \qquad k = 2, \dots, n \tag{5}$$

$$\sum_{v_i, v_j \in S} x_{ij} \le |S| - \left\lceil \frac{\sum_{v_i \in S} E(\xi_i)}{Q} \right\rceil, \qquad (S \subset V \setminus \{v_1\}; 2 \le |S| \le n-2) \tag{6}$$

$$0 \le x_{ij} \le 1, \qquad 2 \le i < j < n \tag{7}$$

$$0 \le x_{1j} \le 2, \qquad j = 2, \dots, n \tag{8}$$

$$x = (x_{ij}), \qquad \text{integer} \tag{9}$$

where, $Q(x) = \sum^{m} \min\{Q^{k,1}, Q^{k,2}\}$.

Solution Method: Integer L-shaped Alg.(General B&C)

First Relaxation: Current Problem (CP^0)

1. Replace $Q(x)$ by Θ in (1).
2. Bound Θ from below by L in (11).
3. Relax subtour elimination and capacity constraints (4).
4. Relax integrality requirements (7).

At Each Iteration

1. Solve relaxation.
2. Add subtour elimination and capacity constraints (4) when violated.
3. Add LBF cuts (10) to improve lower bounds on Θ.
4. Branching to achieve integrality requirements (7).
5. Add optimality cuts (11) when $Q(x) > \Theta$.

Solution Method: Integer L-shaped Alg.(General B&C)

$$CP : \min_{x,\Theta} \quad \sum_{i<j} c_{ij}x_{ij} + \Theta \tag{10}$$

subject to $(4),\ (5),\ (7),\ (8),$

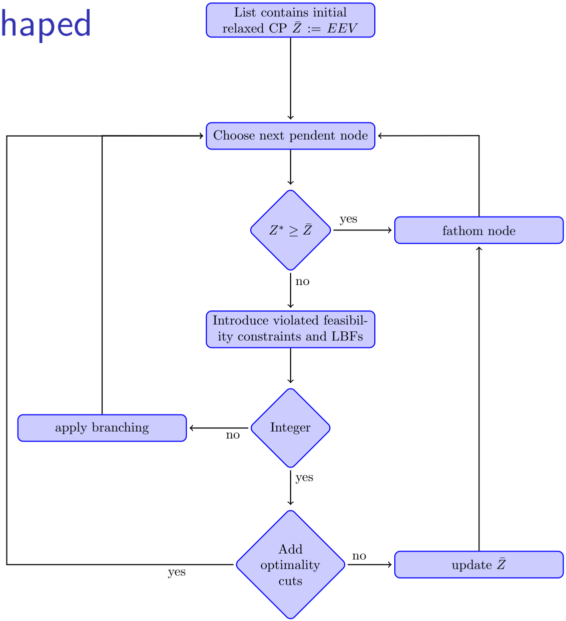
$$\sum_{v_i,v_j \in S^k} x_{ij} \leq |S^k| - \left\lceil \frac{\sum_{v_i \in S^k} E(\xi_i)}{Q} \right\rceil \qquad 2 \leq |S^k| \leq n-2, \tag{11}$$

$$L + (\Theta_p^q - L)\left(\sum_{h \in \mathbf{PR}^q} W_p^h(x) - |\mathbf{PR}^q| + 1 \right) \leq \Theta \quad p \in \{\alpha, \beta, \gamma\}, \tag{12}$$

$$L \leq \Theta \tag{13}$$

$$\sum_{\substack{1 \leq i \leq j \\ x_{ij}^f = 1}} x_{ij} \leq \sum_{1 \leq i \leq j} x_{ij}^f - 1 \qquad . \tag{14}$$

Integer L-Shaped Algorithm



An Exact Algorithm

Feasibility cuts

Remove infeasible routes by adding capacity and subtour elimination constraints (CVRP package Lysgaard et al 2014).

Bounding Scheme for Fractional Solutions

Add valid inequalities that improve lower bound at fractional solutions with certain structures
This requires to compute expected recourse cost of such fractional solutions.

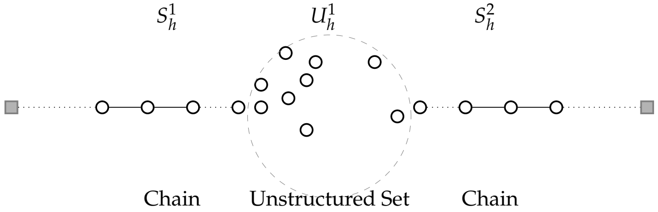
General Lower Bound

Compute a general lower bound L that improves optimality gap.

Literature Review

Bounding Scheme for Fractional Solutions

- Hjorring and Holt (1999) introduce the concept of partial route for single-VRPSD (traditional, discrete, restricted failures)
- Laporte et al. (2002) generalize for multi-VRPSD (traditional, continuous)
- Jabali et al. (2014) generalize the structure of partial routes (traditional, continuous)



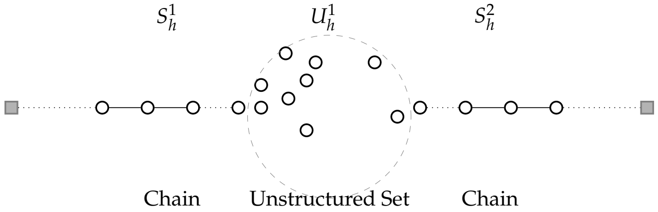
An Exact Algorithm

Bounding Scheme for Fractional Solutions

Add valid inequalities that improve lower bound at fractional solutions with certain structures.
This requires to compute expected recourse cost of such fractional solutions.

Fractional Solution

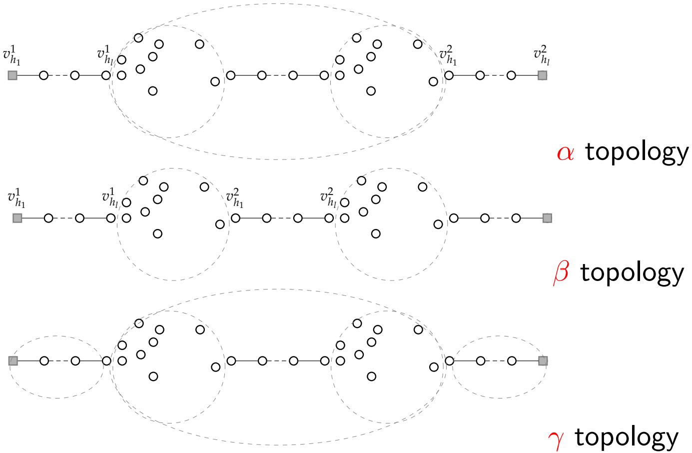
Fractional Solutions with certain structures are called partial routes.



An Exact Algorithm

Fractional Solution

Partial Routes proposed by Jabali et al. (2014).



An Exact Algorithm

Bounding Scheme for Fractional Solutions

Add valid inequalities that improve lower bound at fractional solutions with certain structures.
This requires to compute expected recourse cost of such fractional solutions.

Expected Recourse Cost

First: we model a partial route with α topology as an artificial route.

$$\tilde{h} = (\mathrm{DEPOT} = v_{i_1}, \ldots, v_{i_{j-l}}, \square_{i_{j-l+1}}, \ldots, \square_{i_k}, \square_{i_{k+1}}, \ldots, \square_{i_j}, v_{i_{j+1}}, \ldots, v_{i_{t+1}} = \mathrm{DEPOT})$$

An Exact Algorithm: An Optimal Policy for VRPSD

Bounding Scheme for Fractional Solutions

This requires to compute expected recourse cost of such fractional solutions.

Expected Recourse Cost

Conditional optimal cost-to-go from $\square_{i_a}$ when $\square_{i_{a+1}}$ is restricted to $v_{u_1} \in U_h^1$ can be computed as $\hat{F}_{i_a}(s = (\square_{i_a}, q), s' = (v_{u_1}, q'))$:

$$\hat{F}_{i_a}(s = (\square_{i_a}, q), s' = (v_{u_1}, q')) = \min \begin{cases} \sum_{k:\xi_{u_1}^k \le q} \widetilde{F}_{i_{a+1}}(s' = (v_{u_1}, q' := q - \xi_{u_1}^k)) p_{u_1}^k + \\ \sum_{k:\xi_{u_1}^k > q} [b + 2c_{1,u_1} + \widetilde{F}_{i_{a+1}}(s' = (v_{u_1}, q' := Q + q - \xi_{u_1}^k))] p_{u_1}^k, \\ \\ c_{1,i_a} + c_{1,u_1} - c_{i_k,u_1} + \sum_{k=1}^{s_{u_1}} \widetilde{F}_{i_{a+1}}(s' = (v_{u_1}, q' := Q - \xi_{u_1}^k)) p_{u_1}^k. \end{cases}$$

$$\widetilde{F}_{i_a}(s = (\square_{i_a}, q)) = \min_{v_{u_e} \in U_h^1} \hat{F}_{i_a}(s = (\square_{i_a}, q), s' = (v_{u_e}, q')). \qquad (15)$$

An Exact Algorithm

General Lower Bound

Compute a general lower bound L that improves optimality gap

Definition

The general lower bound L is defined by Laporte and Louveaux (1993) as a feasible integer solution (i.e., m vehicle routes) with the least expected recourse cost defined as follows

$$L = Q(x^L) \leq \min_x \{Q(x) | x \text{ is feasible } \}$$

.

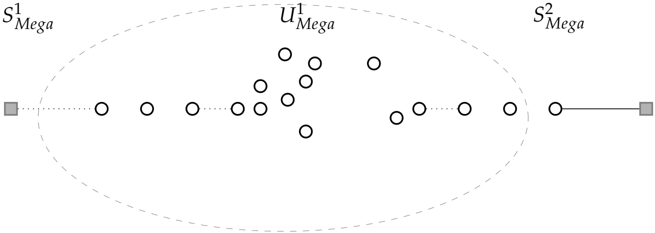
An Exact Algorithm

General Lower Bound

Compute a general lower bound L that improves optimality gap

Computation

We define a Mega partial route including all customers except one, then $L^* = \min_{S_h^2} Q(Mega) \leq L = Q(x^L)$ using bounding scheme.



An Exact Algorithm

General Lower Bound

Compute a general lower bound L that improves optimality gap

How we can use a Mega structure to compute L?

Define $Mega$ as

$$\tilde{l_z} = (v_1 = v_{i_1}, \square_{i_2}, \square_{i_3}, \ldots, \square_{i_{t-1}}, v_z, v_{i_{t+1}} = v_1).$$

for each vertex v_z.
Finally, a general lower bound L^* can be computed as

$$L^* = \min_{z:2,\ldots,n} \widetilde{F}_{v_1}^{\tilde{l_z}}(Q) - \sum_{k=1}^{m-1} c_{\mathrm{PR}}^k.$$

where, c_{PR}^k denotes the k^{th} least PR trip cost$(1 \leq k \leq m)$ and prove that $L^* \leq L$, so L^* is a general valid lower bound for Expected Recourse cost.

Three Groups of Instances

1. Symmetric Instances.
2. Asymmetric Instances.
3. Instances generated by Louveaux and Salazar (2017).

Instance Generation

Symmetric Instances

Three demand ranges $[1, 5]$, $[6, 10]$, and $[11, 15]$ are considered, and customers are assigned randomly to them.
Probabilities $\{0.1, 0.2, 0.4, 0.2, 0.1\}$ are associated accordingly.
Number of runs=11*10*4=440.

Asymmetric Instances

Five demand ranges $[1, 5]$, $[6, 10]$, $[11, 15]$, $[4, 7]$, and $[9, 12]$ are considered, and customers are assigned randomly to them.
Probabilities $\{0.1, 0.2, 0.4, 0.2, 0.1\}$ are associated accordingly for first three ranges and $\{0.4, 0.3, 0.2, 0.1\}$ for last two.
Number of runs=11*10*4=440.

Optimally solved inst., gap, and running time

Table: Result of running the optimal policy.

n	m	$\bar{f}$	Opt	$\leq 1\%$	Run(sec)	Gap	$\bar{f}$	Opt	$\leq 1\%$	Run(sec)	Gap
20	2	0.90	10	10	13.90	0.00%	0.92	10	10	10.00	0.00%
30	2	0.90	10	10	6.40	0.00%	0.92	8	10	1.12	0.05%
40	2	0.90	10	10	15.60	0.00%	0.92	10	10	80.90	0.00%
40	3	0.90	5	9	713.20	0.30%	0.92	8	8	4991.38	0.39%
40	4	0.90	0	2	– – –	2.45%	0.92	0	1	– – –	3.82%
50	2	0.90	10	10	20.60	0.00%	0.92	9	9	1412.78	0.19%
50	3	0.90	4	7	1609.50	1.03%	0.92	4	8	8656.25	0.67%
50	4	0.90	2	2	331.00	3.68%	0.92	1	1	7775.00	3.26%
60	2	0.90	10	10	1017.00	0.00%	0.92	9	10	82.78	0.02%
60	3	0.90	3	7	2978.33	0.77%	0.92	1	4	757.00	2.14%
60	4	0.90	0	2	– – –	2.82%	0.92	0	1	– – –	2.86%

Optimally solved inst., gap, and running time

Table: Result of running the optimal restocking policy.

n	m	$\bar{f}$	Opt	$\leq 1\%$	Run(sec)	Gap	$\bar{f}$	Opt	$\leq 1\%$	Run(sec)	Gap
20	2	0.94	10	10	1.70	0.00%	0.96	10	10	60.60	0.00%
30	2	0.94	10	10	2134.30	0.00%	0.96	9	9	1261.89	0.18%
40	2	0.94	10	10	8.20	0.00%	0.96	9	10	683.22	0.00%
40	3	0.94	4	8	11504.25	0.81%	0.96	2	3	19371.50	1.17%
40	4	0.94	0	1	– – –	3.01%	0.96	0	1	– – –	4.21%
50	2	0.94	10	10	44.20	0.00%	0.96	7	10	457.57	0.16%
50	3	0.94	3	8	1199.67	0.78%	0.96	1	4	1262.00	1.38%
50	4	0.94	0	2	– – –	2.50%	0.96	0	0	– – –	3.61%
60	2	0.94	8	10	574.25	0.07%	0.96	7	10	1580.86	0.11%
60	3	0.94	1	3	1006.00	1.91%	0.96	1	2	32411.00	2.02%
60	4	0.94	0	2	– – –	3.28%	0.96	1	1	9785.00	4.06%

Optimally solved inst., gap, and running time

Table: Result of running the optimal policy.

n	m	$\bar{f}$	Opt	$\leq 1\%$	Run(sec)	Gap	$\bar{f}$	Opt	$\leq 1\%$	Run(sec)	Gap
20	2	0.90	10	10	51.90	0.00%	0.92	10	10	74.80	0.00%
30	2	0.90	10	10	4.30	0.00%	0.92	10	10	103.30	0.00%
40	2	0.90	10	10	8.10	0.00%	0.92	10	10	40.30	0.00%
40	3	0.90	5	9	2285.40	0.23%	0.92	7	8	4587.86	0.46%
40	4	0.90	1	3	2207.00	1.93%	0.92	1	2	238.00	2.71%
50	2	0.90	10	10	68.10	0.00%	0.92	10	10	318.60	0.00%
50	3	0.90	7	8	2883.14	0.79%	0.92	4	5	4153.75	1.04%
50	4	0.90	2	2	9245.50	3.13%	0.92	1	2	1478.00	2.49%
60	2	0.90	8	10	511.00	0.10%	0.92	9	10	728.22	0.05%
60	3	0.90	5	8	6968.40	0.21%	0.92	0	6	– – –	1.19%
60	4	0.90	1	1	11020.00	4.77%	0.92	0	1	– – –	3.26%

Optimally solved inst., gap, and running time

Table: Result of running the optimal restocking policy.

n	m	$\bar{f}$	Opt	$\leq 1\%$	Run(sec)	Gap	$\bar{f}$	Opt	$\leq 1\%$	Run(sec)	Gap
20	2	0.94	10	10	0.30	0.00%	0.96	10	10	848.40	0.00%
30	2	0.94	10	10	365.40	0.00%	0.96	10	10	976.90	0.00%
40	2	0.94	9	10	184.89	0.02%	0.96	10	10	658.80	0.00%
40	3	0.94	4	9	3594.00	0.28%	0.96	2	6	21227.00	1.01%
40	4	0.94	0	2	− − −	3.11%	0.96	0	1	− − −	2.89%
50	2	0.94	9	10	18.22	0.00%	0.96	10	10	4675.00	0.00%
50	3	0.94	6	8	6655.67	0.27%	0.96	1	3	1720.00	1.27%
50	4	0.94	0	1	− − −	3.28%	0.96	0	0	− − −	3.79%
60	2	0.94	8	10	1693.88	0.05%	0.96	9	10	2866.00	0.09%
60	3	0.94	2	5	11888.00	0.96%	0.96	1	4	6095.00	1.43%
60	4	0.94	0	3	− − −	3.35%	0.96	0	0	− − −	3.59%

Overall Performance

Table: Performance of Optimal Restocking Policy.

Integer L-Shaped Algorithm		
opt. sol.	time	gap
51.59%	1407.42	0.90%

Savings: Sav1(savings on total cost) and Sav2(savings on recourse cost)

Table: Optimal restocking policy vs classical

n	m	$\bar{f}$	Sav1	Sav2	$\bar{f}$	Sav1	Sav2
20	2	0.90	0.42%	40.57%	0.92	0.86%	45.16%
30	2	0.90	0.33%	31.21%	0.92	0.26%	40.78%
40	2	0.90	0.08%	43.11%	0.92	0.31%	47.34%
40	3	0.90	0.16%	36.69%	0.92	0.30%	35.98%
40	4	0.90	– – –	– – –	0.92	– – –	– – –
50	2	0.90	0.03%	49.58%	0.92	0.27%	56.73%
50	3	0.90	0.09%	46.08%	0.92	0.10%	43.37%
50	4	0.90	0.12%	26.61%	0.92	0.12%	23.61%
60	2	0.90	0.14%	54.11%	0.92	0.19%	54.05%
60	3	0.90	0.12%	30.33%	0.92	0.26%	76.15%
60	4	0.90	– – –	– – –	0.92	– – –	– – –

Savings: Sav1(savings on total cost) and Sav2(savings on recourse cost)

Table: Optimal restocking policy vs classical

n	m	$\bar{f}$	Sav1	Sav2	$\bar{f}$	Sav1	Sav2
20	2	0.94	1.78%	53.47%	0.96	2.61%	55.38%
30	2	0.94	0.71%	44.39%	0.96	1.39%	53.85%
40	2	0.94	0.48%	55.08%	0.96	1.22%	62.90%
40	3	0.94	0.68%	38.94%	0.96	1.67%	57.43%
40	4	0.94	– – –	– – –	0.96	– – –	– – –
50	2	0.94	0.68%	59.56%	0.96	0.95%	64.37%
50	3	0.94	0.35%	53.32%	0.96	2.33%	64.37%
50	4	0.94	– – –	– – –	0.96	– – –	– – –
60	2	0.94	0.39%	55.01%	0.96	0.97%	67.57%
60	3	0.94	0.12%	36.01%	0.96	1.21%	56.21%
60	4	0.94	– – –	– – –	0.96	2.24%	39.80%

Savings: Sav1(savings on total cost) and Sav2(savings on recourse cost)

Table: Optimal restocking policy vs classical

n	m	$\bar{f}$	Sav1	Sav2	$\bar{f}$	Sav1	Sav2
20	2	0.90	0.27%	27.97%	0.92	0.86%	42.68%
30	2	0.90	0.33%	38.92%	0.92	0.20%	39.72%
40	2	0.90	0.12%	39.13%	0.92	0.07%	48.11%
40	3	0.90	0.13%	46.04%	0.92	0.40%	40.22%
40	4	0.90	0.53%	31.11%	0.92	0.37%	35.28%
50	2	0.90	0.16%	49.76%	0.92	0.24%	56.14%
50	3	0.90	0.09%	35.75%	0.92	0.87%	58.17%
50	4	0.90	0.16%	24.06%	0.92	0.43%	43.94%
60	2	0.90	0.07%	48.68%	0.92	0.24%	56.14%
60	3	0.90	0.17%	37.89%	0.92	−−−	−−−
60	4	0.90	0.13%	39.10%	0.92	−−−	−−−

Savings: Sav1(savings on total cost) and Sav2(savings on recourse cost)

Table: Optimal restocking policy vs classical

n	m	$\bar{f}$	Sav1	Sav2	$\bar{f}$	Sav1	Sav2
20	2	0.94	1.07%	45.48%	0.96	3.10%	57.52%
30	2	0.94	0.46%	43.24%	0.96	1.58%	52.68%
40	2	0.94	0.36%	59.29%	0.96	1.18%	64.97%
40	3	0.94	0.61%	43.66%	0.96	2.10%	48.42%
40	4	0.94	– – –	– – –	0.96	– – –	– – –
50	2	0.94	0.20%	56.19%	0.96	0.80%	58.47%
50	3	0.94	0.83%	57.10%	0.96	1.82%	56.66%
50	4	0.94	– – –	– – –	0.96	– – –	– – –
60	2	0.94	0.27%	50.97%	0.96	0.83%	66.41%
60	3	0.94	0.56%	41.98%	0.96	1.13%	62.16%
60	4	0.94	– – –	– – –	0.96	– – –	– – –

Louveaux and Salazar-Gonzalez (2017)

Instance Generation

Identical demands:

3 realizations: $\xi \in \{4, 5, 6\}$, Pr : $\{0.25, 0.5, 0.25\}$

9 realizations: $\xi \in \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$,

Pr : $\{0.04, 0.08, 0.12, 0.16, 0.2, 0.16, 0.12, 0.08, 0.04\}$

$$F_j(\tilde{q}) = \min \begin{cases} \text{Proceed} : \displaystyle\sum_{k:\xi_{j+1}^k \le \tilde{q}} F_{j+1}(\tilde{q} - \xi_{j+1}^k) p_{j+1}^k + \\ \qquad \displaystyle\sum_{k:\xi_{j+1}^k > \tilde{q}} [\Delta + 2c_{0,j+1} + F_{j+1}(Q + \tilde{q} - \xi_{j+1}^k)] p_{j+1}^k, \\ \text{Replenish} : \Delta + c_{0,j} + c_{0,j+1} - c_{j,j+1} + \displaystyle\sum_{k=1}^{s} F_{j+1}(Q - \xi_{j+1}^k) p_{j+1}^k. \end{cases}$$

Comparison with Louveaux and Salazar-Gonzalez (2017)

Table: Result for $\Delta = 0$

Instance				Our result						Louveaux and Salazar (2017)					
Instance	Veh.	$\bar{f}$	Scen.	Node	Run(min)	Gap	Routing	Recourse	L	Node	Routing	Recourse	Gap	Run(min)	L
E031-09h	2	0.90	3	12	0.00	0.00	332	0.752989	0.000000	325	332	0.7530	0	0.00	0.0000
E031-09h	2	0.95	3	115	0.00	0.00	334	1.295551	0.000000	2035	334	1.2956	0	0.02	0.0000
E031-09h	2	0.90	9	2447	0.17	0.00	334	3.673848	0.000000	3632	334	3.6738	0	0.04	0.0000
E031-09h	2	0.95	9	11923	6.28	0.00	334	10.525120	0.000000	36654	334	10.5251	0	1.30	0.0000
E031-09h	3	0.85	3	305	0.03	0.00	358	0.947237	0.000000	17950	358	0.9472	0	0.28	0.0000
E031-09h	3	0.90	3	3303	0.90	0.00	364	0.065544	0.000000	94518	364	0.0655	0	4.07	0.0000
E031-09h	3	0.85	9	22825	21.08	0.00	361	6.155214	0.000000	248044	361	6.1552	0	18.17	0.0000
E031-09h	3	0.90	9	92237	$5h.$	0.64	361	11.783918	0.000000	604022	363	10.1294	2.325	5 h.	0.0000
E051-05e	2	0.90	3	13	0.00	0.00	441	0.000234	0.000000	3260	441	0.0002	0	0.06	0.0000
E051-05e	2	0.95	3	139	0.01	0.00	441	0.311264	0.000000	3889	441	0.3113	0	0.10	0.0000
E051-05e	2	0.90	9	3967	1.29	0.00	441	2.006072	0.000000	10709	441	2.0061	0	0.30	0.0000
E051-05e	2	0.95	9	51292	$5h.$	0.59	441	7.082580	0.000000	314798	441	7.0826	0.130	5 h.	0.0000
E051-05e	3	0.85	3	17	0.01	0.00	459	0.002388	0.000000	6557	459	0.0000	0	0.16	0.0000
E051-05e	3	0.90	3	7	0.01	0.00	459	0.049098	0.000000	3449	459	0.0491	0	0.07	0.0000
E051-05e	3	0.85	9	1489	2.44	0.00	459	1.550320	0.000000	22525	459	1.5503	0	0.62	0.0000
E051-05e	3	0.90	9	80279	$5h.$	0.21	460	5.629006	0.000000	303297	460	5.6290	0	72.44	0.0000

Comparison with Louveaux and Salazar-Gonzalez (2017)

Table: Result for $\Delta = 0$

Instance				Our result						Louveaux and Salazar (2017)					
Instance	Veh.	$\bar{f}$	Scen.	Node	Run(min)	Gap	Routing	Recourse	L	Node	Routing	Recourse	Gap	Run(min)	L
E076-07s	2	0.90	3	1	0.00	0.00	549	0.004528	0.000000	757	549	0.0045	0	0.03	0.0000
E076-07s	2	0.95	3	187	0.06	0.00	550	0.163882	0.000000	7869	550	0.1639	0	0.34	0.0000
E076-07s	2	0.90	9	897	0.27	0.00	550	0.815011	0.000000	8522	550	0.8150	0	0.27	0.0000
E076-07s	2	0.95	9	47267	$5h.$	0.35	550	4.799920	0.000000	425613	550	4.7999	0	252.76	0.0000
E076-07s	3	0.85	3	251	0.65	0.00	567	0.125934	0.000000	43343	567	0.1259	0	2.75	0.0000
E076-07s	3	0.90	3	4033	27.59	0.00	568	1.266042	0.000000	213546	568	1.2660	0	36.14	0.0000
E076-07s	3	0.85	9	22870	$5h.$	0.14	568	1.952144	0.000000	440721	568	1.9521	0	107.42	0.0000
E076-07s	3	0.90	9	20039	$5h.$	0.50	570	3.249967	0.000000	579000	571	3.3077	1.029	5 h.	0.0000
E101-08e	2	0.90	3	1	0.00	0.00	640	0.001000	0.000000	2819	640	0.0010	0	0.19	0.0000
E101-08e	2	0.95	3	46525	$5h.$	0.07	640	1.731616	0.000000	83765	640	1.7316	0	29.32	0.0000
E101-08e	2	0.90	9	44929	169.55	0.00	640	1.304019	0.000000	34436	640	1.3040	0	13.05	0.0000
E101-08e	2	0.95	9	26237	$5h.$	0.83	640	6.119062	0.000000	172619	640	6.1191	0.851	5 h.	0.0000
E101-08e	3	0.85	3	2036	9.40	0.00	655	0.354229	0.000000	9025	655	0.3542	0	0.98	0.0000
E101-08e	3	0.90	3	26418	$5h.$	0.05	657	1.296878	0.000000	40442	657	1.2969	0	9.07	0.0000
E101-08e	3	0.85	9	8483	$5h.$	0.59	655	4.483187	0.000000	292928	657	1.9841	0	105.42	0.0000
E101-08e	3	0.90	9	8172	$5h.$	1.36	657	9.554738	0.000000	267310	658	9.5547	1.545	5 h.	0.0000

Comparison with Louveaux and Salazar-Gonzalez (2017)

Table: Result for $\Delta = 100$

Instance	Instance Veh.	$\bar{f}$	Scen.	Our result Node	Run(min)	Gap	Routing	Recourse	L	Louveaux and Salazar (2017) Node	Routing	Recourse	Gap	Run(min)	L
E031-09h	2	0.90	3	169	0.01	0.00	334	0.036972	0.003998	669	334	0.0370	0	0.01	0.0322
E031-09h	2	0.95	3	3265	0.37	0.00	334	11.169266	5.190493	1129	334	11.1693	0	0.01	9.7750
E031-09h	2	0.90	9	59511	5h.	3.04	334	25.855218	8.598236	4989	334	25.8552	0	0.06	21.8767
E031-09h	2	0.95	9	74436	5h.	9.75	334	67.120808	30.996017	46716	334	67.1208	0	2.18	55.7272
E031-09h	3	0.85	3	1471	0.23	0.00	358	3.573217	0.000002	52467	358	3.5732	0	1.63	0.0003
E031-09h	3	0.90	3	3821	1.13	0.00	364	0.452652	0.050483	101055	364	0.4527	0	6.67	0.3827
E031-09h	3	0.85	9	127177	5h.	3.74	364	23.823043	3.143872	541383	364	23.8230	0	158.13	19.8215
E031-09h	3	0.90	9	78789	5h.	9.58	366	54.263840	12.249430	575050	364	55.1384	2.570	5 h.	45.4655
E051-05e	2	0.90	3	17	0.00	0.00	441	0.002620	0.000026	2733	441	0.0026	0	0.06	0.0024
E051-05e	2	0.95	3	14766	25.34	0.00	441	3.595178	0.712359	2852	441	3.5952	0	0.07	3.2511
E051-05e	2	0.90	9	46241	5h.	2.29	441	16.760808	3.383876	14095	441	16.7608	0	0.43	14.5815
E051-05e	2	0.95	9	26849	5h.	7.07	442	51.976878	18.781663	368184	441	52.5861	0.530	5 h.	44.9723
E051-05e	3	0.85	3	20	0.01	0.00	459	0.062053	0.000000	8771	459	0.0000	0	0.32	0.0000
E051-05e	3	0.90	3	107	0.03	0.00	459	0.342615	0.000271	8710	459	0.3426	0	0.22	0.2906
E051-05e	3	0.85	9	49377	5h.	1.66	465	9.691321	0.232376	20158	459	10.1148	0	0.97	8.4706
E051-05e	3	0.90	9	27171	5h.	7.54	465	40.333816	3.920572	375607	460	40.6002	0	117.98	34.6055

Comparison with Louveaux and Salazar-Gonzalez (2017)

Table: Result for $\Delta = 100$

Instance				Our result						Louveaux and Salazar (2017)					
Instance	Veh.	$\bar{f}$	Scen.	Node	Run(min)	Gap	Routing	Recourse	L	Node	Routing	Recourse	Gap	Run(min)	L
E076-07s	2	0.90	3	1	0.00	0.00	549	0.049578	0.000000	1433	549	0.0496	0	0.06	0.0002
E076-07s	2	0.95	3	25005	197.20	0.00	550	2.762314	0.129071	4344	550	2.7623	0	0.17	2.5724
E076-07s	2	0.90	9	32914	$5h.$	0.96	550	8.695714	1.162683	17355	550	8.6957	0	0.91	7.7989
E076-07s	2	0.95	9	20332	$5h.$	5.28	551	43.766938	14.337217	2079	557	43.2132	0	0.09	37.5621
E076-07s	3	0.85	3	1417	3.89	0.00	567	1.238007	0.000000	90188	568	0.0098	0	9.19	0.0000
E076-07s	3	0.90	3	15759	$5h.$	0.69	574	0.434985	0.000000	472268	570	0.4141	0	189.78	0.0035
E076-07s	3	0.85	9	23601	$5h.$	1.37	573	5.347527	0.029557	583500	571	3.0801	0.496	5 h.	2.7316
E076-07s	3	0.90	9	15355	$5h.$	5.61	574	31.493586	2.006243	313769	579	25.1061	2.687	5 h.	21.8722
E101-08e	2	0.90	3	1	0.00	0.00	640	0.013355	0.000000	1296	640	0.0134	0	0.10	0.0000
E101-08e	2	0.95	3	16934	$5h.$	0.66	645	0.384135	0.018200	176070	645	0.3841	0.657	5 h.	0.3482
E101-08e	2	0.90	9	20995	$5h.$	1.32	643	6.560891	0.481734	184880	643	6.5609	0.654	5 h.	4.3275
E101-08e	2	0.95	9	13665	$5h.$	4.65	644	38.262388	10.511748	186210	645	35.4935	1.467	5 h.	30.2575
E101-08e	3	0.85	3	16055	$5h.$	0.01	655	4.586534	0.000000	101256	657	0.0016	0	45.68	0.0000
E101-08e	3	0.90	3	18444	$5h.$	0.55	663	0.340071	0.000000	334797	661	0.6416	0.333	5 h.	0.0022
E101-08e	3	0.85	9	13544	$5h.$	2.33	664	11.504967	0.002695	217234	659	16.5437	2.873	5 h.	1.3399
E101-08e	3	0.90	9	13786	$5h.$	5.85	660	41.696028	0.630127	145040	682	16.3244	4.211	5 h.	13.8368

Conclusions and Perspectives

Conclusion and perspectives

- Stochastic vehicle routing is a rich and promising research area.
- Much work remains to be done in the area of recourse definition.
- SVRP models and solution techniques may also be useful for tackling problems that are not really stochastic, but which exhibit similar structures
- Up to now, very little work on problems with stochastic travel and service times, while one may argue that travel or service times are uncertain in most routing problems!
- Correlation between uncertain parameters is possibly a major stumbling block in many application areas, but almost no one seems to work on ways to deal with it.