A Hybrid Monte Carlo Local Branching Algorithm for the Single Vehicle Routing Problem with Stochastic Demands

Michel GENDREAU
Interuniversity Research Centre on Enterprise Networks,
Logistics and Transportation (CIRRELT)
Département d'informatique et de recherche opérationnelle
Université de Montréal

Walter REI
CIRRELT and Université du Québec à Montréal

Patrick SORIANO
CIRRELT and HEC Montréal

VIP'08
Oslo – June 12-14, 2008

Presentation Outline

1. The single vehicle routing problem with stochastic demands
2. Monte Carlo sampling in stochastic programming
3. Local branching
4. Monte Carlo local branching hybrid algorithm
5. Computational results
6. Conclusion

The Single-Vehicle VRP with Stochastic Demands

- A stochastic VRP in which a single capacitated vehicle must deliver (unknown) demands to a set of customers.
- Customers demands are revealed only when the vehicle arrives at a given location.
- The vehicle follows an a priori (TSP) tour, until it returns to the depot or it cannot meet the demand of a customer (route failure).
- When a failure occurs, the vehicle returns to the depot to get replenished (recourse action).
- A special case of the classical VRPSD.
- More complex recourse strategies (e.g., various restocking schemes) could be handled in the same fashion.

Notation

- $G(V, E)$: an undirected graph
- $V = \{v_1, \ldots, v_N\}$: the set of vertices
- $E = \{(v_i, v_j) : v_i, v_j \in V, i < j\}$: the set of edges
- v_1: the a depot where the vehicle must start and finish its route
- $\xi_j, j \in V \setminus \{v_1\}$: (stochastic) demand of customer j
- D: capacity of the vehicle
- $C = [c_{ij}]$: travel costs between vertices
- $\overline{f} = \sum_{j=1}^{N} \mathbf{E}[\xi_j]/D$: expected filling rate of the vehicle

Formulation

$$
\begin{aligned}
\text{Min} \quad & \sum_{i<j} c_{ij} x_{ij} + \mathcal{Q}(x) && && (1)\\
\text{s.t.} \quad & \sum_{j=2}^{N} x_{1j} = 2, && && (2)\\
& \sum_{i<k} x_{ik} + \sum_{j>k} x_{kj} = 2, && k = 2, \ldots, N, && (3)\\
& \sum_{i\in S} \sum_{j\notin S,\ j>i} x_{ij} + \sum_{i\notin S} \sum_{j\in S,\ j>i} x_{ij} \geq 2, && S \subseteq V,\ |S| \geq 3, && (4)\\
& x_{ij} \in \{0,1\}, && 1 \leq i < j \leq N. && (5)
\end{aligned}
$$

- $\mathcal{Q}(x)$ is the recourse function, which gives the expected cost of recourse.
- Constraints (2) and (3) ensure that the route starts and ends at the depot and that each customer is visited once.
- Inequalities (4) are the subtour elimination constraints.

Previous Work – Exact methods

- Gendreau, Laporte, and Séguin (1995): application of 0–1 integer L-shaped algorithm.
- Hjorring and Holt (1999): introduction of a new type of cuts that use information taken from partial routes.
- Rei, Gendreau, and Soriano (2006): new inequalities based on local branching for the 0–1 integer L-shaped algorithm; excellent results on instances with Normal (independent) demands.

Instances where both the filling rate and the number of customers are large still present a tremendous challenge which justifies the development of efficient heuristics for this problem.

Previous Work – Heuristics (Related Problems)

- Gendreau, Laporte, and Séguin (1996): tabu search procedure for routing problems where customers and demands are stochastic.
- Yang, Mathur, and Ballou (2000): heuristics for routing problems with stochastic demands for which restocking (returning to the depot before visiting the next customer) is considered.
- Bianchi *et al.* (2005): several metaheuristics for stochastic routing problems that allow restocking.
- Secomandi (2000, 2001): neuro-dynamic programming algorithms for the case where re-optimization is applied.
- Chepuri and Hommem-De-Mello (2005): cross-entropy method to solve an alternate formulation (some customers may not be serviced, but at a penalty).

Monte Carlo Sampling in Stochastic Programming

Linderoth, Shapiro, and Wright (2006) distinguish two types of approaches:

- Interior approaches solve the problem at hand directly, but whenever the algorithm being used requires information concerning the recourse function, sampling is applied to approximate this information.
  - Dantzig and Glynn (1990): sampling in the L-shaped algorithm to estimate cuts
  - Higle and Sen (1996): stochastic decomposition
  - Ermoliev (1988): stochastic quasi-gradient methods (sampling used to produce a quasi-gradient from which a descent direction is obtained)
- In the exterior approach, one uses sampling beforehand as a way to approximate the recourse function (next slide).

Sampling the Recourse Function

- We want to solve $\min_{x \in X} f(x) = \mathbf{E}_\xi[c^\top x + Q(x, \xi(\omega))] = c^\top x + \mathbf{E}_\xi[Q(x, \xi(\omega))]$.
- Let $\{\omega^1, \ldots, \omega^n\}$ be a subset of randomly generated events of Ω, then $\widehat{f}_n(x) = c^\top x + \frac{1}{n} \sum_{i=1}^{n} Q(x, \xi(\omega^i))$ is a sample average approximation of $f(x)$.
- One may now define the approximating problem as $\min_{x \in X} \widehat{f}_n(x)$.
- Mak, Morton, and Wood (1999): the average value of the approximating problem over all possible samples is a lower bound on the optimal value of the problem.
- Similarly, if $\tilde{x}$ is a feasible first-stage solution, then $\mathbf{E}\left[\widehat{f}_n(\tilde{x})\right] \geq f(\tilde{x})$.

Sampling the Recourse Function (con'd)

- By using unbiased estimators for $\mathbf{E}\left[\min_{x\in X} \widehat{f}_n(x)\right]$ and for $\mathbf{E}\left[\widehat{f}_n(\tilde{x})\right]$, one can construct confidence intervals on the optimal gap associated with $\tilde{x}$.
- Unbiased estimators can be obtained by using batches of subsets $\{\omega^1, \ldots, \omega^n\}$. Let $\widehat{f}_n^j$ be the jth sample average approximation function using a randomly generated subset of size n and let $\widehat{v}_n^j = \min_{x\in X} \widehat{f}_n^j(x)$, for $j = 1, \ldots, m$. Then $L_m^n = \frac{1}{m}\sum_{j=1}^m \widehat{v}_n^j$ and $U_m^n = \frac{1}{m}\sum_{j=1}^m \widehat{f}_n^j(\tilde{x})$ can be used to estimate the gap associated with $\tilde{x}$.
- Under certain conditions, if $\hat{x}_n$ is an optimal solution to problem $\min_{x\in X} \widehat{f}_n(x)$, then it can be shown that $\hat{x}_n$ converges with probability 1 to the set of optimal solutions to the original problem as $n \to \infty$.
- Shapiro and Homem-De-Mello (2000): when the probability distribution of ξ is discrete, given some assumptions, $\hat{x}_n$ is an exact optimal solution for n large enough.

The Sampling Average Approximation Method

- Kleywegt, Shapiro, and Homem-De-Mello (2001): definition of the sample average approximation (or SAA) method
  - Randomly generate batches of samples of random events.
  - Solve the approximating problems (each solution obtained is an approximation of the optimal solution to the original stochastic problem).
  - Estimates on the optimal gap using bounds L_m^n and U_m^n are then generated to obtain a stopping criterion.
  - n may be increased if either the gap or the variance of the gap estimation is to large.
- The SAA method was adapted for the case of stochastic programs with integer recourse by Ahmed and Shapiro (2002).
- Linderoth, Shapiro, and Wright (2006): numerical experiments using the SAA method that show the usefulness of the approach.

Local Branching

- A method introduced by Fischetti and Lodi (2003) to take advantage of the fact that certain generic solvers (e.g., CPLEX) are quite efficient to solve small integer 0-1 problems.
- Therefore, one can divide the feasible region of a problem into a series of smaller subregions and then use a generic solver in order to better explore each of the subregions created.
- In the case of a 0-1 integer problem, the function used in order to divide the feasible region is the Hamming distance defined from a given integer point.
- Let us suppose that we are solving $\min_{x \in X} f(x) = c^\top x + \mathcal{Q}(x)$, where
  - $X = \{x \mid Ax = b,\ x \in X \cap \{0,1\}^{n_1}\}$,
  - x^0 is vector of 0-1 values such that $x^0 \in X$,
  - $N_1 = \{1, \ldots, n_1\}$ and $S_0 = \{j \in N_1 \mid x_j^0 = 1\}$,

  the Hamming distance relative to x^0 is $\Delta(x, x^0) = \sum_{j \in S_0} (1 - x_j) + \sum_{j \in N_1 \setminus S_0} x_j$.

Local Branching (con'd)

- Using function $\Delta(x, x^0)$, one can divide the feasible region of the problem, by creating two subproblems, one for which the constraint $\Delta(x, x^0) \leq \kappa$ is added, and the other for which $\Delta(x, x^0) \geq \kappa + 1$ is added (where κ is a certain fixed integer value).
- Constraint $\Delta(x, x^0) \leq \kappa$ can considerably reduce the size of the feasible region of problem when value κ is fixed to an appropriate value.
- Therefore, one can use an adapted generic solver in order to solve this subproblem.
- Using the new solution found, the procedure may continue by dividing the subregion defined by $\Delta(x, x^0) \geq \kappa + 1$ into two more subproblems where the smaller subregion is explored in the same way as before.
- If the left the problem is infeasible or unattractive, a diversification procedure is applied (enlarge feasible set).

Monte Carlo Sampling and Local Branching

- When using Monte Carlo sampling to approximate the recourse function, one alleviates the stochastic complexity of the problem.
- Local branching allows one to control the combinatorial explosion associated with the first-stage of problem.
- We now show how principles from Monte Carlo sampling and local branching can be combined to form the basis for developing an effective multi-descent heuristic for the SVRPSD.

Local Branching with Sampling

A straighforward approach:

- Use a fixed-size sample of scenarios to represent demand uncertainty; this defines a simpler SVRPSD, which is just a fairly large MIP.
- Solve this MIP with Fischetti and Lodi's procedure.

This is not what we will do.

An Important Insight

- In “reasonable” instances of the VRPSD, the expected number of failures, while significant, is still low.
- This implies that the cost of the a priori routes will in general amount for a large fraction of the overall objective.
- When there is only one vehicle, the a priori route is just a traveling salesman tour on the depot and the customers.
- Thus, our optimal first-stage solution has to be a pretty good solution to the TSP.
- In fact, we can interpret this solution as an optimal TSP solution “adjusted” to account for possible failures.

Multi-descent Scheme

- Could be used without sampling if computing the recourse function exactly was tractable.
- The search is structured around fixed-depth descents according to the local branching scheme.
- The very first base descent starts from the solution of TSP defined on the depot and the customers.
- To induce diversification, descents after the first one are initiated by solving the TSP to which we add the local branching constraints for the right-hand side branch of all descents performed (all cuts previously found are also included).

Descent Structure

- As indicated descents are of fixed depth; new samples of realizations are drawn for each local branching problem.
- The local branching problem is solved using the branch-and-cut algorithm of Rei, Gendreau, and Soriano (2006) with
  1. subtour elimination constraint,
  2. partial route cuts
  3. local branching cuts

Test Problems

- The problem generator used follows the same principles as the one proposed in Hjorring and Holt (1999).
- Graph vertices were generated in a $[0, 100]^2$ square following uniform distributions and the cost matrix was then set to be the Euclidean distances between vertices.
- Each customer was assigned an average demand following a $[1, 10]$ uniform distribution and the standard deviation was set to be 30% of the mean.
- Problems of sizes $n = 60,\ 70,\ 80,\ 90$ were created.
- For each size, five instances were generated for which $\overline{f} = 1.025, 1.05, 1.075, 1.10$.
- 60 instances (difficult ones were selected).
- Additional tests were made on 20 instances of size 150.
- All experiments were performed on a 2.4 GHz AMD Opteron 64-bit processor.

Computational Experiments

- The experiments are organized in three phases.
- First phase: determination of the best value for the size of the neighbourhoods (κ) and the number of scenarios (n) that should be used to solve the local branching subproblems.
- Second phase, analysis of how results vary when the number of descents is increased.
- Also, comparison of the multi-descent scheme with the L-shaped algorithm of Rei *et al.* and with the Or-opt algorithm described in Yang *et al.* (2000).
- The initial solution for the Or-opt heuristic is obtained by applying a greedy insertion procedure.
- From this initial route, the Or-opt exchange algorithm is then called to improve the solution.
- Or-opt moves are evaluated exactly (i.e., using the original recourse function).
- Third phase: all algorithms are tested and compared on the larger problems generated (i.e., N=150).
- All results for the multi-descent heuristic algorithm are average values over five runs.
- All experiments were performed on a 2.4 GHz AMD Opteron 64-bit processor.

N	$\overline{f}$	nb. i.	$\kappa = 4$ $n = 100$	$n = 200$	$n = 300$	$\kappa = 6$ $n = 100$	$n = 200$	$n = 300$	$\kappa = 8$ $n = 100$	$n = 200$	$n = 300$
60	1.025	1	1314.54	1313.2	1312.43∗†	1314.72	1313.72	1312.43∗†	1314.32	1312.43∗†	1312.43∗†
	1.050	3	1347.5	1344.68∗	1346.85	1344.34	1343.24∗	1345.85	1341.08∗†	1343.21	1343.47
	1.075	5	1333.97	1334.05	1333.74∗	1332.26	1332.28	1331.99∗†	1332.99	1333.06	1332.88∗
	1.100	5	1343.14	1343.13	1341.96∗	1338.03	1337.96∗†	1339.03	1341.01	1340.2∗	1340.41
70	1.025	3	1434.6∗	1434.72	1434.82	1430.89∗†	1433.67	1433.53	1433.25	1433.13	1432.38∗
	1.050	3	1401.91	1401.51	1401.33∗	1401.8	1401.11∗†	1401.8	1401.74	1401.3	1401.19∗
	1.075	5	1455.5	1454.82∗	1454.82∗	1444.4	1442.78∗	1445.72	1443.68	1442.68∗†	1443.89
	1.100	4	1498.74	1497.45∗	1499.62	1495.57	1498.28	1495.25∗	1496.08	1494.53∗†	1495.1
80	1.025	2	1483.34	1481.99∗†	1485.77	1482.98∗	1483.06	1483.17	1483.09	1482.94∗	1483.92
	1.050	2	1494.04∗	1494.43	1494.6	1494.18	1494.03	1493.93∗†	1494.72	1493.96	1493.93∗†
	1.075	5	1491.76∗	1491.87	1491.95	1487.55∗	1488.32	1488.13	1487.23	1486.55∗†	1487
	1.100	5	1503.97	1501.01	1500.12∗	1495.02	1494.89	1494.09∗†	1494.19∗	1494.36	1496.33
90	1.025	2	1576.52	1577.63	1575.63∗	1575.27∗	1575.31	1575.63	1574.74	1573.96	1573.08∗†
	1.050	5	1606.11	1606.29	1605.87∗	1605.56∗	1606.56	1606.45	1604.73∗†	1604.96	1605.58
	1.075	5	1605.54∗	1606.28	1605.92	1604.48	1604.62	1603.92∗	1602.51	1602.01∗†	1604.73
	1.100	5	1598.81∗	1599.23	1599.41	1599.02	1598.76	1598.59∗	1596.54∗†	1596.55	1596.71
Local best (∗)			5	4	8	5	4	7	4	7	6
Absolute best (†)			0	1	1	1	2	4	3	5	3

Table 1: The effect of κ and n on the quality of the solutions obtained for one descent

N	Category	nb. i.	L-Shaped		M-LB-2		M-LB-4		M-LB-6		M-LB-8		Or-Opt	
			Val.	Gap	Val.	Gap	Val.	Gap	Val.	Gap	Val.	Gap	Val.	Gap
60	sol.	9	1336.88	0.98%	1338.77	1.12%	1337,87	1.06%	1337,25	1.01%	1336,67	0.97%	1399,19	5.39%
	not sol.	5	1324.93	2.06%	1330.96	2.51%	1328.92	2.36%	1328.67	2.34%	1328.45	2.32%	1355.86	4.30%
70	sol.	6	1409.93	0.95%	1411.54	1.06%	1410.25	0.97%	1410.32	0.97%	1408.59	0.85%	1471.73	5.11%
	not sol.	9	1469.13	1.64%	1469.41	1.66%	1467.35	1.52%	1466.90	1.49%	1466.59	1.47%	1525.39	5.27%
80	sol.	7	1463.69	0.97%	1466.58	1.16%	1466.30	1.15%	1465.58	1.10%	1464.71	1.04%	1533.07	5.45%
	not sol.	7	1517.92	3.54%	1511.70	3.15%	1510.00	3.04%	1508.60	2.95%	1507.83	2.90%	1588.56	7.83%
90	sol.	3	1606.30	0.96%	1606.84	0.99%	1605.79	0.92%	1605.65	0.92%	1605.59	0.91%	1662.59	4.31%
	not sol.	13	1592.46	2.46%	1594.13	2.56%	1592.53	2.46%	1591.51	2.40%	1590.75	2.35%	1644.35	5.53%
	und.	1	-	-	1618.76	1.80%	1618.56	1.79%	1618.56	1.79%	1618.56	1.79%	1699.13	6.44%
Total	sol.	25	1422.25	0.97%	1424.19	1.10%	1423.35	1.04%	1422.93	1.01%	1422.05	0.95%	1485.69	5.20%
	not sol.	34	1505.13	2.42%	1505.44	2.44%	1503.64	2.32%	1502.80	2.27%	1502.24	2.23%	1558.95	5.79%

Table 2: Average Solution Quality (value and optimal gap)

N	Category	nb. i.	L-Shaped	M-LB-2	M-LB-4	M-LB-6	M-LB-8	Or-Opt
60	sol.	9	25.39	5.77	12.04	18.31	24.59	0.48
	not sol.	5	100.62	6.60	13.23	20.16	26.68	0.55
70	sol.	6	11.39	7.56	14.82	23.02	30.36	0.94
	not sol.	9	100.26	12.30	24.72	37.03	50.29	0.88
80	sol.	7	23.81	15.24	29.94	44.92	60.62	2.18
	not sol.	7	100.30	18.64	38.28	57.67	78.13	2.13
90	sol.	3	28.69	16.51	33.79	53.48	72.27	3.35
	not sol.	13	100.30	24.78	50.52	76.24	102.20	3.78
	und.	1	104.91	24.26	53.96	95.05	121.74	3.57
Total	sol.	25	21.98	10.14	20.33	31.11	41.78	1.41
	not sol.	34	100.33	17.54	35.69	53.79	72.40	2.20

Table 3: Average Solution Times (min.)

N	$\overline{f}$	Category	nb. i.	L-Shaped		M-LB-2		M-LB-4		M-LB-6		Or-Opt	
				Val.	Gap	Val.	Gap	Val.	Gap	Val.	Gap	Val.	Gap
150	1.025	sol.	1	1955.09	1.00%	1947.50	0.61%	1947.50	0.61%	1947.50	0.61%	2041.48	5.19%
		not sol.	4	1911.30	1.67%	1898.93	1.03%	1898.35	1.00%	1898.24	0.99%	2020.59	6.99%
150	1.050	sol.	1	1904.16	0.99%	1919.45	1.77%	1914.71	1.53%	1908.98	1.24%	2029.01	7.08%
		not sol.	4	1956.94	2.67%	1957.04	2.67%	1956.18	2.63%	1954.54	2.55%	2044.62	6.85%
150	1.075	not sol.	5	1992.79	3.41%	1984.81	3.02%	1984.32	3.00%	1983.01	2.93%	2100.94	8.37%
150	1.100	not sol.	5	1983.74	3.22%	1988.45	3.43%	1982.86	3.15%	1981.85	3.10%	2082.83	7.79%
Total		sol.	2	1929.63	0.99%	1933.47	1.19%	1931.11	1.07%	1928.24	0.92%	2035.25	6.13%
		not sol.	18	1964.20	2.80%	1960.57	2.61%	1958.55	2.51%	1957.52	2.46%	2065.54	7.56%

Table 4: Results on larger problems: average solution quality (value and optimal gap)

N	$\overline{f}$	Category	nb. i.	L-Shaped	M-LB-2	M-LB-4	M-LB-6	Or-Opt
150	1.025	sol.	1	33.81	24.00	45.60	76.73	33.82
		not sol.	4	302.44	55.65	104.83	160.31	43.24
150	1.050	sol.	1	225.97	75.41	154.29	227.22	41.86
		not sol.	4	303.97	78.64	154.12	236.77	44.27
150	1.075	not sol.	5	302.77	79.33	163.76	246.26	41.11
150	1.100	not sol.	5	301.36	79.50	158.71	238.69	36.11
Total		sol.	2	129.89	49.70	99.95	151.97	37.84
		not sol.	18	302.57	73.96	147.12	222.95	40.89

Table 5: Results on larger problems: average solution times (min.)

Conclusion

- We have proposed a new hybrid algorithm that combines both local branching principles and Monte Carlo sampling in a multi-descent search strategy for integer 0-1 stochastic programming problems.
- By controlling simultaneously the inherent complexities associated with both the first-stage problem and the recourse function, one is able to better limit the effort needed to solve the approximating subproblems.
- Furthermore, by using the local branching constraints in order to obtain diversification for the search strategy, one is able to better explore the feasible region of the original problem.
- This method was specialized to the case of the SVRPSD and was proven to be quite effective to solve hard instances of the problem.
- However, the algorithmic principles that were used are all quite general.
- In future work, it would be interesting to see how one could adapt these ideas to other stochastic programming problems.