Node-based and flow-based formulations for the Generalized Vehicle Routing Problem

06-815 Course Project

Jay Shah - jrshah@andrew.cmu.edu

1 Introduction

Several real world situations can be modeled as a GVRP and its special cases. The post-box collection problem described in (Laporte et al., 1996) becomes an assymetric GVRP if one or more vehicles are used. The distribution of goods by sea in an archipelago as in Phillippines, New Zealand, Italy, Greece etc. can also be modeled as a GVRP by selecting a number of potential harbors for each island and a fleet of ships is required to visit exactly one harbor for each island.

All calculations were done on a Windows PC with an Intel i7 processor (2.90 GHz).

2 Definition of the GVRP

Let $G = (V, A)$ be a directed graph with $V = \{0, 1, 2, \ldots, n\}$ as the set of vertices and $A = \{(i, j)|i, j \in V, i \neq j\}$ as the set of arcs. A non-negative cost c_{ij} is associated with wach arc $(i, j) \in A$.

The set of vertices V is partitioned into $k+1$ mutually exclusive subsets (clusters) $V_0, V_1, \ldots, V_k$. The cluster V_0 has only one vertex 0 which represents the depot and the remaining n nodes belong to the remaining k clusters. These can represent geographically distrbuted customers or islands. Each customer has a demand and the demand of each cluster (denoted by q) is the total demand of customers in that cluster. The demand of each cluster can be satsfied via any of its vertices. There exist m vehicles, each with capacity Q.

The generalized vehicle routing problem (GVRP) consists in finding the minimum total cost tours starting and ending at the depot, such that each cluster should be visited exactly once, the entering and leaving nodes of each cluster is the same and the sum of all the demands of any tour (route) does not exceed the capacity of the vehicle Q. An example of a GVRP with a feasible tour is shown in Fig. 1.

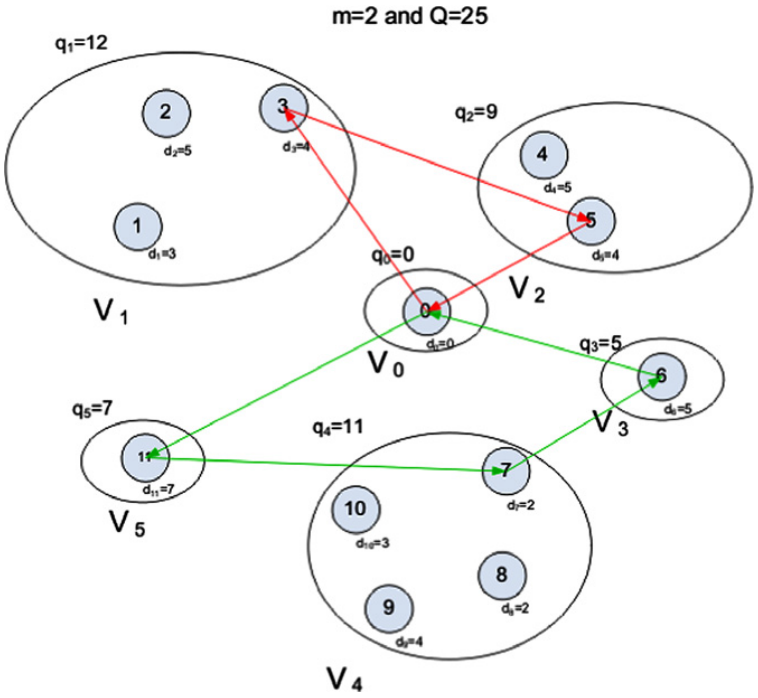


Figure 1: An example of a GVRP and a feasible solution (Pop et al., 2012)

3 MILP formulations

The first mixed integer linear formulation of the GVRP was presented by Kara and Bektas (2003) .

3.1 General formulation

The related sets, decision variables and parameters for the GVRP are defined as follows:

Sets:

$V = \{0, 1, 2, \ldots, n\}$ the set of nodes corresponding to the customers, where 0 represents the depot. V is partitioned into mutually exclusive and exhaustive non-empty sub-sets V_0, V_1, $\ldots V_k$, each of which represents a cluster of customers, where $V_0 = \{0\}$ is the depot.

$K = \{0, 1, 2, \ldots, k\}$ is the set of clusters.

$A = \{(i, j)|i, j \in V, i \neq j\}$ is the set of arcs.

Decision variables:

We define the following binary variables:

$$x_{ij} = \begin{cases} 1 & \text{if arc } (i,j) \text{ is included in the tour of a vehicle, } i \in V_p, j \in V_r, p, r \in K, \\ 0, & \text{otherwise} \end{cases}$$

$$W_{pr} = \begin{cases} 1 & \text{if there is a path from cluster } V_p \text{ to cluster } V_r, \ p, r \in K, \\ 0, & \text{otherwise} \end{cases}$$

Parameters:

c_{ij} is the cost of traveling from node i to node j, $i \neq j$, $i \in V_p$, $j \in V_r$, $p \neq r$, $p, r \in K$;

d_i is the demand of customer i, $i = 1, 2, \ldots, n$;

q_r is the demand of the cluster V_r, $q_r = \sum_{i \in V_r} d_i$, $r \in K$;

$M = \{1, \ldots, m\}$ is the set of identical vehicles;

Q is the capacity of each vehicle.

We assume that $k \geq m$ ie. the number of vehicles is no greater than the number of clusters.

3.1.1 Cluster degree constraints

For each cluster except V_0, only a single outgoing arc to any other node belonging to other clusters can exist. This condition is expressed by:

$$\sum_{i \in V_p} \sum_{j \in V \setminus V_p} x_{ij} = 1, p \neq 0, p \in K \tag{1}$$

There can be exactly one incoming arc to a cluster from any other node belonging to other clusters, excluding V_0. This condition is implied by:

$$\sum_{i \in V \setminus V_p} \sum_{j \in V_p} x_{ij} = 1, p \neq 0, p \in K \tag{2}$$

There should be at most m leaving arcs from and at most m entering arcs to the depot:

$$\sum_{i=1}^{n} x_{0i} \leq m \tag{3}$$

$$\sum_{i=1}^{n} x_{i0} \leq m \tag{4}$$

For the purposes of this study, the constraints (3) & (4) have been taken to be equality constraints.

3.1.2 Cluster connectivity constraints

The entering and leaving nodes should be the same for each cluster, which is satisfied by:

$$\sum_{i \in V \setminus V_p} x_{ij} = \sum_{i \in V \setminus V_p} x_{ji}, j \in V_p, p \in K \tag{5}$$

Flows from cluster V_p to the cluster V_r are defined by w_{pr}. Thus, w_{pr} should be equal to the sum of x_{ij}'s from V_p to V_r:

$$w_{pr} = \sum_{i \in V_p} \sum_{j \in V_r} x_{ij} \tag{6}$$

With the above definitions and constraints, a general integer programming formulation for the GVRP is given as:

$$\begin{aligned}
&\min \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij} \\
&\text{s.t. (1) - (6)} \\
&\qquad \text{Capacity bounding constraints} && (7) \\
&\qquad \text{Subtour elimination constraints} && (8) \\
&\qquad x_{i,j} \in \{0,1\}, \forall (i,j) \in A
\end{aligned}$$

We do not need to impose integrality constraints on w_{pr} because Eqs. (1) - (4) and (6) automatically imply that w_{pr} is either 0 or 1.

Model formulations of the GVRP are thus constrained by (1) - (6) and "capacity bounding" and "subtour elimination" constraints given in (7) and (8) above.

Pop et al. (2012) has presented two different formulations based on additional auxiliary decision variables defined: *node-based* and *flow-based* formulation.

3.2 Node-based formulation

In this case, the additional auxiliary variables is defined with respect to the vertices of the graph:

u_p: the load of a vehicle just after leaving the cluster V_p, or delivered amount of goods just after leaving the cluster V_p, $p \neq 0$, $p \in K$.

4 Implementation

The described ODE solver was implemented in MATLAB as `ode_Mic` and used to solve various stiff ODEs descried in literature. The code also include options to include an analytical Jacobian function if one is available. If no function is provided, the Jacobian is calculated numerically using finite differences (code `jacobianest`). The tolerance vector was modified each time according to the specific problem being solved.

5 Test examples

5.1 Example 1

We use the method described above to solve a set of rate equations described by Seinfeld et al. **?**:

$$\begin{aligned} \frac{dy_1}{dt} &= -0.04y_1 + 10^4 y_2 y_3 \\ \frac{dy_2}{dt} &= 0.04y_1 - 10^4 y_2 y_3 - 3 \cdot 10^7 y_2^2 \\ \frac{dy_3}{dt} &= 3 \cdot 10^7 y_2^2 \\ y_1(0) = 1, y_2(0) &= 0, y_3(0) = 0 \end{aligned} \tag{9}$$

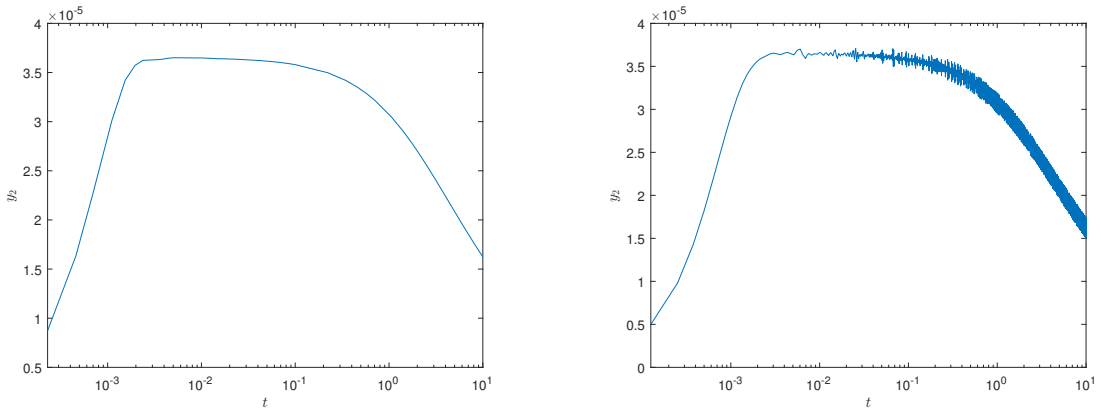
The analytical jacobian matrix for this system is given by:

$$J = \begin{bmatrix} -0.04 & 10^4 y_3 & 10^4 y_2 \\ 0.04 & -10^4 y_3 - 6 \cdot 10^7 y_2 & -10^4 y_2 \\ 0 & 6 \cdot 10^7 y_2 & 0 \end{bmatrix}$$

The tolerance vector was taken as $\epsilon = [10^{-3}, 10^{-7}, 10^{-3}]$. The integration was carried out in the range [0,10] with an initial step size of 10^{-4}. The same problem was also solved using `ode15s`, `ode23s` and `ode45`.

Table 1 shows the statistics involved in computing the solution for the algorithm under consideration and inbuilt MATLAB solvers. All the three stiff solvers show markedly better

performance across all parameters as compared to `ode45`. The performance of `ode_Mic` with an analytical Jacobian is comparable to MATLAB's inbuilt stiff solvers. Figure 2 shows the plot of y_2 vs time for stiff and non-stiff solutions. We see that the solution using `ode45` undergoes oscillations at higher time steps.



(a) Stiff solver (`ode_Mic`, `ode15s` or `ode23s`) (b) Non-stiff solver (`ode45`)

Figure 2: Comparision of solutions for non-stiff vs. stiff solvers (Example 1)

Table 1: Solver statistics for Michelsen's algorithm compared with inbuilt MATLAB solvers (Example 1)

	Running time (s)	Function evaluations	Steps
		`ode_Mic`	
Analytical Jacobian	0.00963	168	29
Numerical Jacobian	0.14337	8960	29
		`ode15s`	
Analytical Jacobian	0.03540	81	41
Numerical Jacobian	0.05238	89	41
		`ode23s`	
Analytical Jacobian	0.00851	60	19
Numerical Jacobian	0.01168	115	19
		`ode45`	
	1.20454	52201	29109

5.2 Example 2

The second test example is a model for a fluid bed reactor (Aiken and Lapidus, 1974):

$$\begin{aligned}
\frac{dy_1}{dt} &= 1.30(y_3 - y_1) + 1.04 \cdot 10^4 k y_2 \\
\frac{dy_2}{dt} &= 1.88 \cdot 10^3 (y_4 - y_2(1+k)) \\
\frac{dy_3}{dt} &= 1752 + 266.7 y_1 - 269.3 y_3 \\
\frac{dy_4}{dt} &= 0.1 + 320 y_2 - 321 y_4 \\
y_1(0) = 759.167, y_2(0) &= 0, y_3(0) = 600, y_4(0) = 0.1
\end{aligned} \tag{10}$$

where $k = 0.0006 \exp(20.7 - 15000/y_1)$.

The tolerance vector was taken as $\epsilon = [1, 1, 0.1, 0.1]$. The integration was carried out in the range [0,500] with an initial step size of 10^{-4}. Similar to Example 1, the running time of `ode_Mic` with a Numerical Jacobian was comparable to that of `ode15s` and `ode23s` (Table 2). The performance of `ode45` is several orders of magnitude worse than the stiff solvers.

Table 2: Solver statistics for Michelsen's algorithm compared with inbuilt MATLAB solvers (Example 2)

	Running time (s)	Function evaluations	Steps
		`ode_Mic`	
Analytical Jacobian	0.014103	252	43
Numerical Jacobian	0.341434	16112	39
		`ode15s`	
Analytical Jacobian	0.190613	2354	764
Numerical Jacobian	0.02084	126	85
		`ode23s`	
Analytical Jacobian	0.018893	341	114
Numerical Jacobian	0.014007	331	48
		`ode45`	
	55.881972	2123530	1327201

5.3 Example 3 - Oregonator reaction

The oscillatory reaction between $HBrO_2$, Br^- and Ce(IV) is known as the Oregonator reaction (Field and Noyes, 1974). The rate expressions are given by:

$$\begin{aligned}
\frac{dy_1}{dt} &= 77.27(y_2 + y_1(1 - 8.375 \times 10^{-6} y_1 - y_2)) \\
\frac{dy_2}{dt} &= \frac{1}{77.27}(y_3 - (1 + y_1)y_2) \\
\frac{dy_3}{dt} &= 0.161(y_1 - y_3) \\
y_1(0) = 1, y_2(0) &= 2, y_3(0) = 3
\end{aligned} \tag{11}$$

In this system, we observe that at smaller integration limits (e.g. [0,10]) the problem is

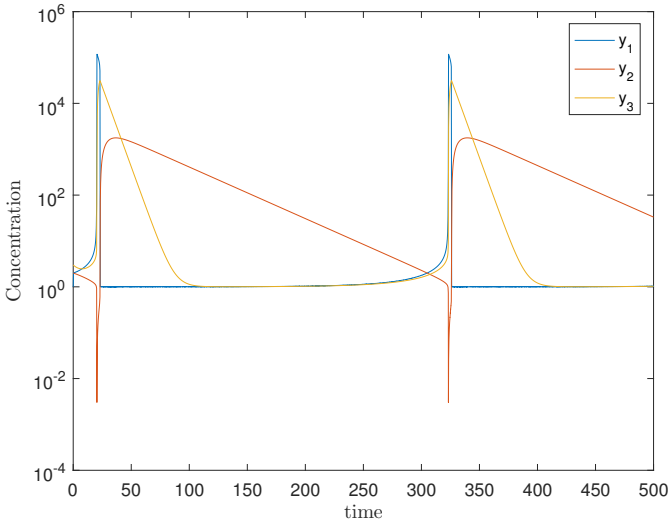


Figure 3: Oregonator reaction plot

non-stiff and both stiff and non-stiff solvers integrate the system effectively. However, at larger intervals (once oscillations start taking place), the stiff solvers show markedly better performance than `ode45` (Table 3).

Table 3: Solver statistics for Michelsen's algorithm compared with inbuilt MATLAB solvers (Example 3 - Oregonator reaction)

	Running time	Function evaluations	Steps
		`ode_Mic`	
Analytical Jacobian	0.15919	5538	923
Numerical Jacobian	4.46163	295203	923
		`ode15s`	
Analytical Jacobian	0.11093	1566	583
Numerical Jacobian	0.11796	1845	583
		`ode23s`	
Analytical Jacobian	0.06939	2142	709
Numerical Jacobian	0.14239	4266	709
		`ode45`	
	584.78864	28648200	17905633

5.4 Example 4 - Photochemical smog model

A kinetic model Photochemical smog formation presented by Gelinas (Gelinas, 1972) is solved. The original model given in ref. (Gelinas, 1972) involves 29 species and over 60 equations with rate constants varying by 20 orders of magnitude. A simplified form of the model as described in ref. (Michelsen, 1977) with 12 species and 22 reactions is used and shown in Tables 4 & 5.

The system was solved using `ode_Mic`, `ode15s`, `ode23s` and `ode45` for increasing integration limits. As expected (especially for this large system), `ode45` is orders of magnitude worse than any of the stiff solvers for all integration limits. Interestingly, `ode45` varies as $\mathcal{O}(n)$ whereas `ode_Mic` varies as $\mathcal{O}(n^{0.5})$ (where n is max integration limit).

Table 4: Components in the smog model

1. NO_2	2. NO	3. O
4. O_3	5. C_4H_8	6. $C_3H_7O_2$
7. HO_2	8. CH_3CO_3	9. CH_3O_2
10. HO	11. $C_4H_8OHO_2$	12. CH_2OHO_2

Table 5: Photochemical smog model (Example 3) - Reactions and rate constants.

1			$\longrightarrow$	2	+	3	6.7×10^{-3}
3			$\longrightarrow$	4			6.73×10^{4}
2	+	4	$\longrightarrow$	1			9.1×10^{-2}
3	+	5	$\longrightarrow$	6	+	7	3.8×10^{2}
6	+	2	$\longrightarrow$	1	+	7	3.0
4	+	5	$\longrightarrow$	7	+	8	4.9×10^{-5}
4	+	5	$\longrightarrow$	4			2.1×10^{-5}
2	+	8	$\longrightarrow$	1	+	9	3.0
1	+	8	$\longrightarrow$	inactive			0.1
5	+	8	$\longrightarrow$	9			1.0×10^{-4}
2	+	9	$\longrightarrow$	1	+	7	3.0
5	+	10	$\longrightarrow$	11			1.0×10^{2}
2	+	11	$\longrightarrow$	1	+	12	3.0
2	+	12	$\longrightarrow$	1	+	10	3.0
2	+	7	$\longrightarrow$	1	+	10	50
7	+	7	$\longrightarrow$	inactive			50
7	+	10	$\longrightarrow$	inactive			50
7	+	6	$\longrightarrow$	inactive			50
7	+	9	$\longrightarrow$	inactive			50
7	+	8	$\longrightarrow$	inactive			50
7	+	11	$\longrightarrow$	inactive			50
7	+	12	$\longrightarrow$	inactive			50

Table 6: Solver statistics for Michelsen's algorithm compared with inbuilt MATLAB solvers (Example 4 - smog model)

Time step (s)	Running time (s)	Function evaluations	Steps
	`ode_Mic`		
1	1.010029	15072	13
10	3.025674	51496	42
100	8.835829	158262	126
1000	25.085625	456571	362
	`ode15s`		
1	0.021004	61	26
10	0.060441	146	48
100	0.085993	233	75
1000	0.034693	284	94
	`ode23s`		
1	0.083637	252	17
10	0.057533	569	38
100	0.081612	914	61
1000	0.104977	1259	84
	`ode45`		
1	6.104102	137725	86245
10	67.269395	1379530	862373
100	653.744277	13792100	8620221

6 Conclusion

The integration algorithm proposed by Michelsen was implemented and found to be well suited for a variety of stiff problems arising in chemical engineering. Using the analytical Jacobian improves performance in each of the stiff solvers used. In all these cases, an arbitrary initial step size was chosen. A more systematic method may improve performance by reducing the number of step length adjustments required for the first step.

References

Aiken, R. C. and Lapidus, L. An effective numerical integration method for typical stiff systems. *AIChE Journal*, 20(2):368–375 (1974).

Dormand, J. and Prince, P. A family of embedded Runge-Kutta formulae. *Journal of Computational and Applied Mathematics*, 6(1):19 – 26 (1980).

Field, R. J. and Noyes, R. M. Oscillations in chemical systems. IV. Limit cycle behavior in a model of a real chemical reaction. *The Journal of Chemical Physics*, 60(5):1877–1884 (1974).

Gelinas, R. J. Stiff systems of kinetic equationsA practitioner's view. *Journal of Computational Physics*, 9(2):222 – 236 (1972).

Ghiani, G. and Improta, G. An efficient transformation of the generalized vehicle routing problem. *European Journal of Operational Research*, 122(1):11 – 17 (2000).

Kara, I. and Bektas, T. New mathematical models of the generalized vehicle routing problem and extensions. *Proc. of the 5th EURO/INFORMS Joint International Meeting* (2003).

Laporte, G., Asef-Vaziri, A., and Sriskandarajah, C. Some Applications of the Generalized Travelling Salesman Problem. *Journal of the Operational Research Society*, 47(12):1461–1467 (1996).

Michelsen, M. L. Application of semi-implicit RungeKutta methods for integration of ordinary and partial differential equations. *The Chemical Engineering Journal*, 14(2):107 – 112 (1977).

Pop, P. C., Kara, I., and Marc, A. H. New mathematical models of the generalized vehicle routing problem and extensions. *Applied Mathematical Modelling*, 36(1):97 – 107 (2012).

Shampine, L. F. and Reichelt, M. W. The MATLAB ODE Suite. *SIAM Journal on Scientific Computing*, 18(1):1–22 (1997).