Transportation Logistics

Part VII: VRP - advanced topics

- Dealing with TW and duration constraints
- A metaheuristic framework
- Solving VRP to optimality

The VPP with Time Windows (VRPTW)

Decision variables

$$x_{ij}^k = \begin{cases} 1, & \text{if arc } (ij) \text{ is traversed by vehicle } k, \\ 0, & \text{otherwise.} \end{cases}$$

$$B_i = \text{beginning of service at } i \text{ by vehicle } k$$

Parameters

$$c_{ij} = \text{the costs to traverse arc } (i,j)$$
$$d_i = \text{demand of customer } i$$
$$C = \text{vehicle capacity}$$

$$t_{ij} = \text{time needed to traverse arc } (i,j)$$
$$s_i = \text{the service time at } i$$
$$a_i = \text{beginning of the time window } i$$
$$b_i = \text{end of the time window } i$$

$K\ldots$ set of vehicles, $V\ldots$ set of all vertices, $A\ldots$ set of arcs, $N\ldots$ set of customers

$n\ldots$ number of customers, $0\ldots$ start depot, $n+1\ldots$ end depot

$$\min \sum_{k\in K}\sum_{i\in V}\sum_{j\in V} c_{ij}x_{ij}^k \tag{1}$$

subject to:

$$\sum_{k\in K}\sum_{j\in V\setminus\{n+1\}} x_{ij}^k = 1 \qquad \forall i \in N, \tag{2}$$

$$\sum_{j\in V} x_{0j}^k = 1 \qquad \forall k \in K, \tag{3}$$

$$\sum_{j\in V\setminus\{n+1\}} x_{ji}^k - \sum_{j\in V\setminus\{0\}} x_{ji}^k = 0 \qquad \forall k \in K, i \in N, \tag{4}$$

$$\sum_{i\in V} x_{i,n+1}^k = 1 \qquad \forall k \in K, \tag{5}$$

$$(B_i^k + s_i + t_{ij})x_{ij}^k \le B_j^k \qquad \forall k \in K, i \in V\setminus\{n+1\}, j \in V\setminus\{0\}, \tag{6}$$

$$a_i \le B_i^k \le b_i \qquad \forall k \in K, i \in V, \tag{7}$$

$$\sum_{i\in N} d_i \sum_{j\in V\setminus\{n+1\}} x_{ji}^k \le C \qquad \forall k \in K, \tag{8}$$

$$x_{ij}^k \in \{0,1\} \qquad \forall k \in K, i,j \in V. \tag{9}$$

VRPTW with duration constraints

Notation

T ... maximum route duration

Constraints

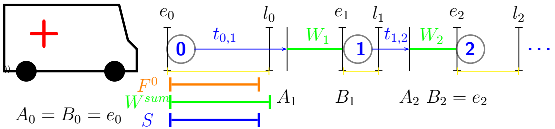
$$B_{n+1}^k - B_0^k \leq T \qquad k \in K \qquad (10)$$

VRPTW with duration constraints

Scheduling:

Forward Time Slack

Savelsbergh, M. (1995) The Vehicle Routing Problem with Time Windows: Minimizing Route Duration, ORSA Journal on Computing 4:146–154



$$F_2^0 = \min\{W_2^{sum}, S_2\},\ W_2^{sum} = W_1^{sum} + W_2,\ S_2 = \min\{S_1, W_2^{sum} + l_2 - B_2\}$$

A metaheuristic framework

Because metaheuristics for the CVRP outperform classical methods in terms of solution quality (sometimes now in terms of computing time), we believe there is little room left for significant improvement in the area of classical heuristics. The time has come to turn the page.

[Concluding words of Laporte and Semet's chapter on Classical Heuristics for the CVRP (2002) in Toth and Vigo (eds): 'The VRP'].

However,

classical heuristics/operators are important ingredients/building blocks for advanced methods, such as metaheuristics!

The metaheuristic idea

Definition

metaheuristic A top-level general strategy which guides other heuristics to search for feasible solutions in domains where the task is hard.

from: `http://encyclopedia2.thefreedictionary.com/metaheuristic`

Whenever there is no additional improving solution in the neighborhood defined by a local search operator (more, swap, ...), classical local search algorithms stop. The obtained solution is called a **local optimum**.

Metaheuristics provide a means to **escape from local optima** by, e.g., allowing intermediate infeasible or deteriorating solutions, solution perturbations, searching larger neighborhoods etc.

Several different types

(more or less in chronological order, not exhaustive)

- **Simulated/Deterministic Annealing** (allows intermediate deteriorations)
- **Tabu Search** (allows intermediate deteriorations (tabu list) and sometimes infeasible solutions)
- **Genetic/Memetic Algorithms** (populations of solutions)
- **Ant Colony Algorithms** (randomized pheromone updates)
- **Variable Neighborhood Search** (perturbations/shaking, may allow intermediate deteriorations and sometimes infeasible solutions)
- **(Adaptive) Large Neighborhood Search** (may allow intermediate deteriorations)

(Adaptive) Large Neighborhood Search

Frist introduced by Shaw (1998).

The idea

destroy parts of the current solution and then **repair** it again.

The name 'Large Neighborhood Search' indicates that a larger neighborhood is searched than typically employed in other neighborhood search based metaheuristics (e.g., tabu search often uses single vertex moves).

The combination of a destroy and a repair operator consitutes such a larger neighborhood.

Adaptive Large Neighborhood Search

1. generate a starting solution s; $s_{best} \leftarrow s$
2. repeat the following for 25.000 iterations
   1. choose a destroy operator d and a repair r operator
   2. apply d to s yielding s'
   3. apply r to s' yielding s''
   4. decide if s'' is accepted as new incumbent solution; if yes $s \leftarrow s''$
   5. check if s'' is better than s_{best}; if yes, $s_{best} \leftarrow s''$
   6. update the scores and weights of the operators
3. return s_{best}

Ropke, S. and Pisinger D. (2006) An Adaptive Large Neighborhood Search Heuristic for the Pickup and Delivery Problem with Time Windows. Transportation Science 40:455–472.

Destroy and Repair operators

used by Ropke and Pisinger (2006):

- random removal
- worst removal
- related removal

- greedy heuristic
- 2-regret
- 3-regret
- 4-regret
- m-regret

Destroy operators

q...number of nodes/requests to be removed

Random removal

randomly remove q requests from the solution s

Destroy operators

q...number of nodes/requests to be removed

Worst removal

- **repeat while** $q > 0$
  - $L =$ array of all planned requests sorted by descending costs $cost(i, s)$
  - choose a random number y from the interval $[0, 1)$
  - $r = L[y^p|L|]$
  - remove r from solution s
  - $q = q - 1$

$cost(i, s) =$ difference in costs if i removed from s

Destroy operators

q...number of nodes/requests to be removed

Related removal

- r = a randomly selected request from s;
- set of requests: $D = \{r\}$;
- **repeat** while $|D| < q$
  - r = a randomly selected request from D
  - L = an array containing all request from s not in D
  - sort L such that $i < j \rightarrow R(r, L[i]) < R(r, L[j])$
  - choose a random number y from the interval $[0, 1)$
    $D = D \cup \{L[y^p|L|]\}$;
- remove the requests in D from s

$R(i, j)$ = relatedness of i and j; weighted combination of, e.g. time and distance

Repair operators

Greedy insertion

In each iteration insert the node/request that can be inserted the cheapest.

Regret insertion

Insert the request with the largest regret value i^* at its best position. Repeat until no further requests can be inserted. $(l \in \{2, 3, 4, m\})$

$$i^* := \arg\max_{i \in V^o} \left\{ \sum_{k=2}^{\min(l,m)} \Big(f_\Delta(i,k) - f_\Delta(i,1) \Big) \right\},$$

The adaptive mechanism

define a weight w_i for each heuristic i
roulette wheel selection:
heuristic j is chosen with probability

$$\frac{w_j}{\sum_i w_i}$$

The adaptive mechanism

adaptive weight adjustment

in the beginning of each time segment (100 it), the score π_i of each heuristic is set to 0. the counter how often i is applied in a given segment is θ_i

scores are increased by σ_1, σ_2, σ_3:

σ_1 destroy repair operation yielded a new global best solution.
σ_2 destroy repair operation yielded a new current solution (never accepted before)
σ_3 destroy repair operation yielded an accepted a worse solution (never accepted)

w_{ij} weight of heuristic i in segment j

$$w_{i,j+1} = w_{ij}(1 - r) + r\frac{\pi_i}{\theta_i}$$

Acceptance scheme

the acceptance scheme is based on a **simulated annealing** criterion:

a solution is accepted with a probability of

$$e^{-(f(s')-f(s))/T}$$

T is called the temperature
in each iteration it is decreased by a cooling rate c: $T = Tc$ $(0 < c < T)$

s is the current solution
s' is the new solution

Adaptive Large Neighborhood Search

1. generate a starting solution s; $s_{best} \leftarrow s$
2. repeat the following for 25.000 iterations
    1. choose a destroy operator d and a repair r operator
    2. apply d to s yielding s'
    3. apply r to s' yielding s''
    4. decide if s'' is accepted as new incumbent solution; if yes $s \leftarrow s''$
    5. check if s'' is better than s_{best}; if yes, $s_{best} \leftarrow s''$
    6. update the scores and weights of the operators
3. return s_{best}

Ropke, S. and Pisinger D. (2006) An Adaptive Large Neighborhood Search Heuristic for the Pickup and Delivery Problem with Time Windows. Transportation Science 40:455–472.

(A)LNS variants have been applied successfully to

- The pickup and delivery problem with TW
  (Ropke and Pisinger, Transportation Science, 2006)
- Different variants of the VRPB
  (Ropke and Pisinger, EJOR, 2006)
- VRPTW, CVRP, MDVRP, site-dependent VRP, OVRP
  (Pisinger and Ropke, Computers & OR, 2007)
- PDP with multiple loading stacks
  (Coté, Gendreau, Potvin, 2009)
- Service technician routing and scheduling
  (Kovacs, Parragh, Doerner, Hartl, J Scheduling, 2011)
- Two-echelon VRP
  (Hemmelmayr, Cordeau, Crainic, 2011)
- ...

Formulating the VRP in terms of a set partitioning problem (SP)

$$\min \sum_{r \in \Omega} c_r x_r \qquad (11)$$

subject to

$$\sum_{r \in \Omega} a_{ir} x_r = 1 \qquad \forall i \in N \qquad (12)$$

$$x_r \in \{0, 1\} \qquad \forall r \in \Omega \qquad (13)$$

N ... set of customers
Ω ... set of all routes
a_{ir} 1 if i on route r, 0, otherwise.

The set Ω is hard to identify; it is potentially very, very large!

So, how can the problem be solved?

By means of column generation embedded into a branch and bound framework.

Column generation ...

... is a technique to solve large scale **linear programs** involving a huge number of variables.

... is based on the idea that only very few variables will be part of the basis ($x_{ij} > 0$) in the solution to the LP. So, it suffices to only consider those that are likely to be part of the basis.

The linear relaxation of SP (LSP)

$$\min \sum_{r \in \Omega} c_r x_r \tag{14}$$

subject to

$$\sum_{r \in \Omega} a_{ir} x_r = 1 \qquad \forall i \in N \qquad \pi_i \tag{15}$$

$$x_r \geq 0 \qquad \forall r \in \Omega \tag{16}$$

π_i is the dual variable associated with constraint (15).

The dual of LSP

$$\max \sum_{i \in N} \pi_i \tag{17}$$

subject to

$$\sum_{i \in N} a_{ir} \pi_i \leq c_r \qquad \forall r \in \Omega \tag{18}$$

$$\pi_i \text{ unrestricted} \qquad \forall i \in N \tag{19}$$

reduced cost (shadow price) of route r:

$$\bar{c}_r = c_r - \sum_{i \in N} a_{ir} \pi_i \geq 0$$

(for routes part of the basis, the reduced cost is 0)

The restricted LSP (RLSP)

$$\min \sum_{r \in \Omega'} c_r x_r \tag{20}$$

subject to

$$\sum_{r \in \Omega'} a_{ir} x_r = 1 \quad \forall i \in N \quad \pi_i \tag{21}$$

$$x_r \geq 0 \quad \forall r \in \Omega' \tag{22}$$

Ω' ... set of variables (columns) generated so far.

What's a promising new variable (column)?
A variable (column) for which the reduced cost
$\bar{c}_r = c_r - \sum_{i \in N} a_{ir} \pi_i \leq 0$

Column generation

- **Initialization** populate Ω' with a set of columns such that a feasible solution is possible (e.g. a heuristic solution to the VRP or all single customer routes)
- **Step 1** solve RLSP on Ω' (called **master problem**)
- **Step 2** retrieve dual information (π_i values)
- **Step 3** solve the **subproblem**: try to find columns (routes) of negative reduced cost $\bar{c}_r = c_r - \sum_{i \in N} a_{ir}\pi_i \leq 0$ (usually this can be done by solving a shortest path problem with additional constraints - based on Dijkstra/Bellman!)
- **if** no additional routes with negative reduced cost exist
    - STOP. The optimal solution to LSP has been found. (if this solution is integer it is also the optimal solution to SP)
- **else**
    - add the new column(s) to Ω' and go to step 1

Observations

★ If we solve the standard CVRP, the suproblem corresponds to solving a shortest path problem with a capacity constraint.

★ If we solve the standard VRPTW, the subproblem corresponds to solving a shortest path problem with time windows and a capacity constraint.

★ Also the subproblems are usually still NP-hard.

★ In general, the more restrictive the constraints (e.g. the tighter the time windows) the smaller the number of feasible routes and the faster the solution of the subproblem.

Many state-of-the-art exact methods ...

... combine column generation with branch and cut (aka branch and cut and price methods)
(for the CVRP: Fukasawa, Longo, Lysgaard, Poggi de Aragao, Reis, Uchoa, Werneck, 2006)

... a very recent successful exact algorithmic framework combines several bounding procedures using ideas from column generation as well as cutting plane generation. Then, based on the obtained lower and upper bound, they solve a restricted version of SP, containing only routes whose reduced cost is smaller than the gap between the upper and the lower bound.
(for the CVRP: Baldacci, Christofides, Mingozzi, 2008)

Largest CVRP instance solved to optimality: around 120 customers
($<$1h computation time - 2.6 GHz PC with 3 GB of RAM)

... the latest trends

Hybrid methods

Algorithms that combine ideas from MIP (branch and cut, column generation, etc.) with metaheuristics.

More complex problems

Integration of several planning levels/decisions.

More complex data

The integration of time-dependent or real time travel times/information.

References

Paolo Toth, and Daniele Vigo (2002) The Vehicle Routing Problem, SIAM. (Chapters 1 and 5)

W. Domschke (1997) 'Logistik: Rundreisen und Touren' Oldenbourg.