Restricted dynamic programming for the VRP

A flexible framework for solving realistic VRPS

Leendert Kok, Marco Schutten (UT, OMPL)
Jelke van Hoorn, Joaquim Gromiccho (ORTEC)
Overview

- Introduction
- DP for TSP and VRP
- Restricting the state space
- Computational results
- Conclusions
Introduction

- Transport costs constitute 4-10% of a product selling price
- Vehicle routing is a complex daily task for companies like logistical service providers and distribution firms
- The vehicle routing problem (VRP) is a generalization of the traveling salesman problem (TSP), which is NP-hard

Therefore, the VRP has been intensively studied for the last ≈ 30 years
Introduction (VRP)

• The VRP is the problem of optimally routing a homogeneous vehicle fleet along a set of customers

• General objectives are: minimizing number of vehicles and minimizing travel distances

• Constraints typically are:
  – Limited vehicle capacity (CVRP)
  – Time windows at customers (VRPTW)
  – Sequencing restrictions (pickup and delivery) (PDP)
Introduction (VRP)

VRPs are often solved using local search, because:

- Has proven to be very successful in obtaining high quality solutions
- Can adapt to many (real-life) restrictions
Example: CROSS exchange
Introduction (VRP)

However,

• Some constraints are hard to incorporate, like:
  – Time-dependent travel times (to account for traffic congestions)
  – Driving hours regulations
  – Truck drivers exchanging trucks

• Each VRP variant requires development of new algorithm
Solution

Solve the VRP using a framework based on restricted dynamic programming:

- Flexible in solving many VRP variants
- Routes are sequentially built → also difficult restrictions (e.g., timing restrictions) are easy to incorporate
- Can solve small problem instances to optimality
Dynamic programming for the TSP

The traveling salesman problem:

• Consider a set $V=\{0, 1, 2, \ldots, n\}$ of cities with distances $c_{ij}$

Objective:

• Visit each city exactly once, starting and ending in city $0$, such that the total distance is minimized
Dynamic programming for the TSP

A state \((S,j)\), \(S\) subset of \(V\setminus 0\) and \(j\) in \(S\), represents a minimum length path, s.t.:

- the path starts in node 0,
- it visits all cities in \(S\), ending in city \(j\)
- \(C(S,j)\) represents the length of such a minimum length path
Dynamic programming for the TSP

The costs of the states in stage 1 are given by:

- \( C(\{j\}, j) = c_{0j} \), for all \( j \) in \( V \)

In each successive stage the costs are calculated by the recurrence relation:

- \( C(S, j) = \min_{i \in S \setminus j} \{ C(S \setminus j, i) + c_{ij} \} \)
Dynamic programming for the TSP

Recurrence relation $C(S,j) = \min_{i \in S \setminus j} \{C(S \setminus j, i) + c_{ij}\}$:

- $(S,j)$ = the minimum length path visiting each city in $S$ and ending in $j$
- In this path, $j$ is preceded by one of the other cities in $S$, say $i^* \in S \setminus j$
- By definition, we have $C(S,j) = C(S \setminus j, i^*) + c_{i^*j}$
- Finally, since $C(S,j)$ is minimal, it must hold that $C(S \setminus j, i^*) + c_{i^*j} = \min_{i \in S \setminus j} \{C(S \setminus j, i) + c_{ij}\}$
Dynamic programming for the TSP

Running time of the DP:

- $V$ contains $2^n$ subsets $S$, each having $|S| \leq n$ possible end cities, resulting in $O(n^*2^n)$ states.
- Determining the value of each state requires the comparison of $|S-1| \leq n$ additions, resulting in:

A total running time of $O(n^2 \cdot 2^n)$, i.e., much smaller than complete enumeration ($O(n!)$).
Dynamic programming for the VRP

Apply DP to the VRP through the giant-tour representation (GTR) of vehicle routing solutions:

- Order the vehicles
- Introduce for each vehicle $k$ a unique origin $o_k$ and destination $d_k$ node
- Create the giant tour by connecting destination and origin nodes of successive vehicles
Example: 2 depots, 3 vehicles, 9 customers

Traditional representation:

Giant-tour representation:
Flexibility of DP for VRP through GTR

• Adding state dimensions
  – Capacity restrictions
  – Time windows
  – Maximum daily driving time

• Control via the input
  – Pickup and delivery
  – Heterogeneous vehicle fleet
  – Multiple depots
Flexibility of DP for VRP through GTR

Even complex timing restrictions can be handled easily:

• Time-dependent travel times:
  – Controlled via the input. Since routes are constructed sequentially, travel times are only requested once

• Driving hours regulations
  – Add state dimensions (remaining daily/weekly driving time, remaining daily/weekly working time, etc.)
Restricted DP

Although the running time of DP is the best found so far, DP is still too slow to solve problem instances of practical sizes:
Restricted DP

Although the running time of DP is the best found so far, DP is still too slow to solve problem instances of practical sizes:

A TSP with 250 customers requires more states than there are atoms in the universe…

(for complete enumeration, the number of solutions exceeds the number of atoms in the universe already with 60 customers)
Restricted DP

Therefore, restrict the state space by:

• Extending each state \((S,j)\) only with the \(E\) nearest neighbors of the endnode \(j\)
• Taking only the \(H\) states with smallest cost to the next stage
Restricted DP

However, requires some tailoring for each problem at hand, for example:

• Take a capacitated VRP: limited capacities, fixed number of vehicles, minimize total travel distance

• Suppose the distance from the depot to its nearest node $i$ is smaller than half the distance from the depot to its second nearest node $j$

Then in stage two the state with lowest cost is the vehicle route $depot-i-depot$
Restricted DP

However, returning low-filled vehicles generally results in a shortage of vehicles. This can be prevented by, e.g.:

• Adding a large cost each time a vehicle returns
• Adding a large cost each time a low-filled vehicle returns
• Allowing a vehicle only to return if the percentage of vehicles used does not exceed the percentage of demand served

Each variant of the VRP may require such tailoring, but the algorithmic framework can be applied to all variants
Computational results

Tested on a set of benchmarks instances for
the CVRP → least restricted VRP variant, so
largest solution space:

• Average optimality gap of 3%
• Computation times of 28 minutes on average
• Increasing $H$ significantly reduces optimality gap:
multiplying $H$ by 10 reduces gap by 30%
Questions?

or contact me, a.l.kok@utwente.nl