Practical relevance of the state-of-the-art exact VRP solvers

Ruslan Sadykov

Inria Bordeaux, France
Université Bordeaux, France

Joint work with Eduardo Uchoa, Artur Pessoa, Eduardo Queiroga, Najib Errami

DIMACS Workshop on Computational Approaches to Vehicle Routing

May 22, 2023
Plan of the talk

Exact SOTA algorithms for vehicle routing and their performance

POPMUSIC matheuristic for VRPs

VRPSolverEasy Python package
Contents

Exact SOTA algorithms for vehicle routing and their performance

POPMUSIC matheuristic for VRPs

VRPSolverEasy Python package
Set-partitioning formulation

- Set of feasible routes = set $P$ of resource constrained paths in graph $G = (V \cup \{0\}, A)$
- $h^p_a = 1$ if and only if path $p \in P$ contains arc $a$, otherwise 0
- Variable $x_a$ — arc $a \in A$ is used in the solution or not
- Variable $\lambda_p$ — path $p \in P$ is used in the solution or not

Min $\sum_{a \in A} c_a x_a$
S.t. $\sum_{a \in \delta(v)} x_a = 2, \quad v \in V,$

$Bx \leq b, \quad$ (add. constraints and robust cuts)
$D\lambda \leq d, \quad$ (non-robust cuts)

$x_a = \sum_{p \in P} h^p_a \lambda_p, \quad a \in A,$

$\sum_{p \in P} \lambda_p \leq K,$

$x_a, \lambda_p \in \{0, 1\}, \quad a \in A, p \in P.$
One continuous variable per feasible route.

Pricing problem is the Resource Constrained Shortest Path problem.
One continuous variable per feasible route.

Pricing problem is the Resource Constrained Shortest Path problem.

Additional constraints (cuts) are added to reduce the integrality gap.

Nodes (customers) are generalized to packing sets.

# of vehicles entering this set of clients $\geq 2$ (robust)

# of vehicles serving at least two of these three clients $\leq 1$ (non-robust)
Resource constrained shortest path problem (RCSP)

Labeling algorithm

- Enumeration of partial paths
- Relies heavily on domination
- Bi-directional search
- Buckets
- Completion bounds
Resource constrained shortest path problem (RCSP)

Labeling algorithm

- Enumeration of partial paths
- Relies heavily on domination
- Bi-directional search
- Buckets
- Completion bounds

Problem restriction using reduced cost arguments

- Remove arcs contained only in paths with reduced cost $> \text{primal-dual gap}$
- Enumerate all elementary paths with reduced cost $\leq \text{primal-dual gap}$

Figure from [Irnich et al., 2010]
Pricing problem relaxation: limited-memory concept

Specific RCSP instances

Few global resources but **hundreds of local resources**.
Pricing problem relaxation: limited-memory concept

Specific RCSP instances

Few global resources but hundreds of local resources.

- Track resource consumption inside limited memory
- Forget resource consumption once outside memory
Pricing problem relaxation: limited-memory concept

Specific RCSP instances

Few global resources but hundreds of local resources.

Dynamic relaxation

Restrict relaxation after column generation convergence: increase limited memories.
Heuristics and branching

Generic heuristics: diving and restricted master IP

- Work fine only for instances with short routes
- Cannot efficiently use diving heuristic due to resource relaxation and a significant primal-dual gap.
- Preliminary path enumeration with false gap is required before running restricted master heuristic.

Branching

- Strong branching is important
- Aggregated branching may be important
Some history

- [Balinski and Quandt, 1964] set-partitioning formulation for CVRP
- [Laporte and Nobert, 1983] branch-and-cut, rounded capacity cuts
- [Desrosiers et al., 1984] first branch-and-price
- [Lysgaard et al., 2004] best branch-and-cut algorithm
- [Fukasawa et al., 2006] robust branch-cut-and-price
- [Baldacci et al., 2008] path enumeration technique
- [Jepsen et al., 2008] (non-robust) subset-row cuts
- [Baldacci et al., 2011] $ng$-route relaxation
- [Pecin et al., 2017] limited-memory technique, best branch-cut-and-price
- Latest survey: [Costa et al., 2019]
- [Pessoa et al., 2020] VRP generic model and exact solver
World record for the CVRP exact solving

Figure: Optimal solution for X-n865-k95 (solved in \(\approx 20\) days)
Computational comparison between BCP and BC
10’000 instances with 100 customers [Queiroga et al., 2022].

All 10’000 instances are solved to optimality thanks to cluster (aggregated) branching [Uchoa and Silva, 2022].
Solutions times (for CVRP)

![Graph showing solution times for CVRP instances.](image-url)

- **X-axis**: Instance size
- **Y-axis**: Solution time (log scale)
- **Legend**: Default BCP with optimal cutoff and 30min time limit
Contents

Exact SOTA algorithms for vehicle routing and their performance

POPMUSIC matheuristic for VRPs

VRPSolverEasy Python package
POP MUSIC for CVRP: illustration
Partial OPtimization Metaheuristic Under Special Intensification Conditions [Taillard and Voss, 2002]

(a) Initial solution and a constructed subproblem. Seed client is marked in black.
(b) Improved solution after finding a better subsolution
POPMUSIC matheuristic for VRPs

Design choices

- Progressive increase of the subproblem size.
- Use of heuristic BCP
  - Time limit
  - False gap mechanism
  - Restricted master heuristic inside BCP

We used [Vidal, 2022] heuristic instead of BCP for the DIMACS VRP challenge!

However, less generic implementation.

How to set the termination criteria for the subproblem?
Design choices

▶ Progressive increase of the subproblem size.
▶ Use of heuristic BCP
  ▶ Time limit
  ▶ False gap mechanism
  ▶ Restricted master heuristic inside BCP

Using heuristic instead of BCP

▶ We used [Vidal, 2022] heuristic instead of BCP for the DIMACS VRP challenge!
▶ Extensively studied in [Santini et al., 2023]
▶ However, less generic implementation.
▶ How to set the termination criteria for the subproblem?
Computational comparison for the CVRP

HGS20: [Vidal, 2022].

$\text{POP}_x^2$: POPMUSIC-BCP starting with $x$-hour solution of [Vidal, 2022].

$X$ instances with $300 < n \leq 1000$ customers
Computational results for the CVRP with backhauls

**ILS-SP**: [Subramanian and Queiroga, 2020].

**POP1800**: POPMUSIC-BCP starting with 30-minutes solution of [Subramanian and Queiroga, 2020]

Instances with 300–1000 clients.
Computational results for the HFVRP

ILS19: [Penna et al., 2019].

POP1800: POPMUSIC-BCP starting with 30-minutes solution of [Penna et al., 2019].

Instances of type XH with 300–1000 clients, both limited and unlimited fleet.
Contents

Exact SOTA algorithms for vehicle routing and their performance

POPMUSIC matheuristic for VRPs

VRPSolverEasy Python package
Four layers of VRPSolver implementation

- **VRPSolverEasy** (simple interface)
  - Python
- **VRPSolver** (involved interface)
  - Julia
  - C++
- **CLP or CPLEX**
- **BaPCod** (generic BCP solver)
  - C++
- **RCSP solver** (designed for BCP)
  - C++
Four layers of VRPSolver implementation

1. **VRPSolverEasy** (simple interface)
   - Python: Open source, free

2. **VRPSolver** (involved interface)
   - Julia: Open source, academic-only

3. **BaPCod** (generic BCP solver)
   - C++: Open source, academic-only

4. **RCSP solver** (designed for BCP)
   - C++: Closed source

---

**Tools**

- CLP or CPLEX
Four layers of VRPSolver implementation

VRPSolverEasy (simple interface)
- Python: Open source free
- depots, vehicles, customers

VRPSolver (involved interface)
- C++: Open source academic-only
- variables, constraints, graphs, resources, packing sets

BaPCod (generic BCP solver)
- Julia: academic-only
- a 15-years old mess

RCSP solver (designed for BCP)
- C++: Closed source
- labels, buckets, VRP cut separation heuristics

CLP or CPLEX
VRPSolverEasy: compromise between generality and simplicity

https://github.com/inria-UFF/VRPSolverEasy

“OR-free” interface
Depots, customers, vehicles, links instead of variables, constraints, graphs, resources.

VRP variants covered
CVRP, VRPTW, HFVRP, MDVRP, OVRP, TOP, parallel links, site-dependent VRP, roaming delivery locations, and combinations.

VRP variants potentially covered
Arc routing, clustered VRP, generalized VRP, VRP with backhauls, Multi-trip VRP, Location-routing.
from VRPSolverEasy.src import solver
model = solver.Model()
model.add_depot(id=0)
for i in range(data.nb_customers):
    model.add_customer(id=i+1, demand=data.cust_demands[i])
for i,cust_i in enumerate(data.cust_coordinates):
    for j in range(i + 1, len(data.cust_coordinates)):
        dist = euclidean_distance(cust_i[0], cust_i[1],
                                    data.cust_coordinates[j][0],
                                    data.cust_coordinates[j][1])
        model.add_link(start_point_id=i+1,
                        end_point_id=j+1,
                        distance=dist)
model.add_vehicle_type(id=1, start_point_id=0,
                        end_point_id=0,
                        max_number=data.nb_customers,
                        capacity=data.vehicle_capacity,
                        var_cost_dist=1)
model.solve()
if model.solution.is_defined():
    print(model.solution)
VRPSolverEasy: computational results

Initial solution by OR-Tools run for \( n/2 \) sec. (included in solution time)

VRPSolverEasy time limit is 30 minutes

CVRP-Small, VRPTW-Solomon, HFVRP-Classic: \( \leq 100 \) customers

gap_I, gap_F — OR-Tools gap, VRPSolverEasy gap

<table>
<thead>
<tr>
<th>Problem</th>
<th>Dataset</th>
<th>Without built-in heuristic</th>
<th></th>
<th>With built-in heuristic</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>#opt</td>
<td>Time</td>
<td>gap_I</td>
<td>gap_F</td>
</tr>
<tr>
<td>CVRP</td>
<td>Small</td>
<td>87/87</td>
<td>33.8</td>
<td>0.99</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>X ≤ 200</td>
<td>14/26</td>
<td>844.9</td>
<td>4.36</td>
<td>2.35</td>
</tr>
<tr>
<td>VRPTW</td>
<td>Solomon</td>
<td>53/56</td>
<td>109.0</td>
<td>14.20</td>
<td>0.14</td>
</tr>
<tr>
<td></td>
<td>HG-200</td>
<td>27/60</td>
<td>881.0</td>
<td>9.02</td>
<td>4.63</td>
</tr>
<tr>
<td>HFVRP</td>
<td>Classic</td>
<td>31/40</td>
<td>234.5</td>
<td>10.35</td>
<td>0.95</td>
</tr>
<tr>
<td></td>
<td>XH ≤ 200</td>
<td>2/22</td>
<td>1730.4</td>
<td>18.07</td>
<td>13.44</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td></td>
<td>215/291</td>
<td></td>
<td>229/291</td>
<td></td>
</tr>
</tbody>
</table>
Practical relevance of exact VRP solvers

Possible now

- Finally easy to use!
- Small and moderate size (up to \( \approx 100 \) customers)
- Not so long routes (up to \( \approx 15 \) customers per route)
- For these instances
  - More generic than SOTA heuristic VRP solvers
  - More efficient than generic VRP solvers (OR-Tools, LocalSolver)

Perspectives

- Industrial-quality codes are needed!
- Parallel matheuristics for larger instances
- Hybrid heuristics which use dual information from column and cut generation


References II


References IV


