Structural decompositions and large neighborhoods for node, edge and arc routing problems

Thibaut Vidal

Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro
Rua Marquês de São Vicente, 225 - Gávea, Rio de Janeiro - RJ, 22451-900, Brazil
vidalt@inf.puc-rio.br

ODYSSEUS
Ajaccio, June 1–5th, 2015
1 Node and edge routing problems

2 Combined neighborhoods for arc routing problems
   - Work rationale and shortest path formulation
   - Cutting off complexity: memories + bidirectional search
   - Cutting off complexity: moves filtering via LBs

3 Problem generalizations

4 Towards “very very” large neighborhoods

5 Computational experiments
   - Integration into two state-of-the-art metaheuristics
   - Comparison with previous literature
   - CARP – To reduce or not to reduce
   - Problems with turn penalties and delays at intersections

6 Conclusions/Perspectives
1 Node and edge routing problems

2 Combined neighborhoods for arc routing problems
   • Work rationale and shortest path formulation
   • Cutting off complexity: memories + bidirectional search
   • Cutting off complexity: moves filtering via LBs

3 Problem generalizations

4 Towards “very very” large neighborhoods

5 Computational experiments
   • Integration into two state-of-the-art metaheuristics
   • Comparison with previous literature
   • CARP – To reduce or not to reduce
   • Problems with turn penalties and delays at intersections

6 Conclusions/Perspectives
Challenges

- Arc routing for home delivery, snow plowing, refuse collection, postal services, among others.
- Bring forth additional challenges beyond “academic” vehicle routing

  \[\Rightarrow\text{ Deciding} \quad \text{on travel directions for services on edges}\]

  \[\Rightarrow\text{ Shortest path between services are conditioned} \quad \text{by service orientations}\]

  (may also need to include some additional aspects such as turn penalties or delays at intersections).
State-of-the-art algorithms

- Until 2010 $\rightarrow$ Separate streams of research on heuristics for arc and node routing problems. Some of the current state-of-the-art algorithms include:

  - **Capacitated Vehicle Routing Problem (CVRP):**
    UTS of Cordeau et al. (1997, 2001), AMP of Tarantilis (2005), ILS/ELS of Prins (2009), ES and HGAs of Mester and Bräysy (2007); Nagata and Bräysy (2009); Vidal et al. (2012)...

  - **Capacitated Arc Routing Problem (CARP):**
    GLS of Beullens et al. (2003), HGA of Lacomme et al. (2001, 2004); Mei et al. (2009), VNS of Polacek et al. (2008), TS of Brandão and Eglese (2008)...

- Arc-routing specific decisions are addressed via a larger number of enumerative neighborhood classes: to optimize service orientations.
State-of-the-art algorithms

- Two alternative solution representations for the CARP:

**R1.** Explicit representation of assignment, sequencing decisions, service orientations, and intermediate paths.

**R2.** Explicit representation of assignment, sequencing decisions, and service orientations. Intermediate paths have been preprocessed.
Recent research on combined node, edge and arc routing problems (NEARP – also called mixed capacitated general routing problem MCGRP):

- Early constructive heuristics: (Pandi and Muralidharan, 1995; Gutiérrez et al., 2002)
- HGA of Prins and Bouchenoua (2005)
- SA of Kokubugata et al. (2007)
- LNS+MIP of Bosco et al. (2014)
- Remarkable unified metaheuristic: Dell’Amico et al. (2014). Covers a large set of CVRP, CARP, and NEARP benchmark instances. However, “AILS uses a total of 26 move subtypes: 13 types of 3-opt, 8 types of 2-opt, 2 types of Or-opt, 2 Swap types, and Flip.”
Large neighborhoods

- Interesting large neighborhood from Muyldermans et al. (2005), scarcely used until now: dynamic programming to generate optimal traversal directions for the services of a fixed route

  ⇒ Used as a stand-alone procedures, or combined with a RELOCATE move. Both searches in $O(n)$

  ⇒ Combined in Irnich (2008) with the neighborhood of Balas and Simonetti (2001), leading to promising results on mail delivery applications.
1. Node and edge routing problems

2. Combined neighborhoods for arc routing problems
   - Work rationale and shortest path formulation
   - Cutting off complexity: memories + bidirectional search
   - Cutting off complexity: moves filtering via LBs

3. Problem generalizations

4. Towards “very very” large neighborhoods

5. Computational experiments
   - Integration into two state-of-the-art metaheuristics
   - Comparison with previous literature
   - CARP – To reduce or not to reduce
   - Problems with turn penalties and delays at intersections

6. Conclusions/Perspectives
Rationale of this work

- Structural problem decomposition (used naturally in branch-and-price, less explicitly used in heuristics):

  \[ \text{Decision set } x_1 \quad \text{Decision set } x_2 \]

  \begin{align*}
  \text{Efficient exact methods, such as bi-directional dynamic programming} \\
  \text{or integer programming on restricted formulations} \\
  \rightarrow \text{used to derive other decisions}
  \end{align*}

  \begin{align*}
  \text{Heuristic search, e.g., local search on a decision set} \\
  \text{Difficult combinatorial optimization problem} \\
  \text{with several families of decisions}
  \end{align*}
Rationale of this work

- Structural problem decomposition:

  Efficient exact methods, such as bi-directional dynamic programming or integer programming on restricted formulations used to derive other decisions

  Difficult combinatorial optimization problem with several families of decisions

  Heuristic search, e.g., local search on a decision set

  SOLUTION AS PERMUTATIONS OF SERVICES

  DECODING in O(1)!

  OPTIMAL EVALUATION OF SERVICE ORIENTATIONS AND INTERMEDIATE PATHS
Solution representation and decoding

- How to decode/evaluate a solution = deriving optimal orientations for the services?

Solution Representation:

```
0 -> σ(1) -> σ(2) -> σ(3) -> σ(4) -> σ(5) -> 0
```

Shortest Path Problem:

```
Depot -> σ(1)
       ∨       ∨
       C_{11}  C_{12}
       ▽       ▽
S_{σ(1)}  S_{σ(2)}
       ∨       ∨
σ(2)
       ∨       ∨
σ(3)
       ∨       ∨
σ(4)
       ∨       ∨
σ(5)
       ∨       ∨
Depot
```

- Each service is represented by two nodes, one for each possible orientation. Travel costs $c_{ij}^{kl}$ between $(i,j)$ are conditioned by the orientations $(k,l)$ for departure and arrival.
Solution representation and decoding

- Same shortest path subproblem as Muyldermans et al. (2005), but used far beyond it’s original scope.
  - Operating a complete problem decomposition: searching in the space of service permutations (+ depot visits)
    - Systematically, for all solution and move evaluations
  - In very large neighborhoods: Ejections chains and Split algorithm
  - Also used to conceal decisions on service modes within the shortest path subproblem, for many variants of arc routing problems
- Evaluated in $\mathcal{O}(1)$ instead of $\mathcal{O}(n)$
- And even, using LBs on move evaluations, same average number of elementary operations as a CVRP move...
• Modern neighborhood-centered heuristics evaluate millions/billions of neighbor solutions during one run.

• Key property of classical routing neighborhoods:
  ▶ Any local-search move involving a bounded number of node relocations or arc exchanges can be assimilated to a concatenation of a bounded number of sub-sequences.
  ▶ Same subsequences appear many times during different moves.

> To decrease the computational complexity, compute auxiliary data on subsequences by induction on concatenation (⊕).
Auxiliary data structures = partial shortest paths

Partial shortest path $C(\sigma)[k, l]$ between the first and last service in the sequence $\sigma$, for any (entry, exit) direction pair $(k, l)$

Initialization

For $\sigma_0$ with a single visit $v_i$, $S(\sigma_0)[k, l] = \begin{cases} 0 & \text{if } k = l \\ +\infty & \text{if } k \neq l \end{cases}$

Evaluation

By induction on the concatenation operator:

$$C(\sigma_1 \oplus \sigma_2)[k, l] = \min_{x, y} \left\{ C(\sigma_1)[k, x] + c_{\sigma_1(|\sigma_1|)\sigma_2(1)}^{xy} + C(\sigma_2)[y, l] \right\}$$
• **Pre-processing partial shortest paths in the incumbent solution** – in $O(n^2)$ before the neighborhood exploration – dramatically simplifies the shortest paths:

![Graph](image)

**Shortest path problem:**

- Only a constant number of edges!
• Each move evaluation was still taking a bit more operations (constant of 4×) than in the classic CVRP.

• Even this can be avoided...
  ⇒ by developing lower bounds on the cost of neighbors...
Let $\bar{Z}(\sigma)$ be a lower bound on the cost of a route $\sigma$.

A move that modifies two routes: $\{\sigma_1, \sigma_2\} \Rightarrow \{\sigma'_1, \sigma'_2\}$ has a chance to be improving if and only if:

$$\Delta_{\Pi} = \bar{Z}(\sigma'_1) + \bar{Z}(\sigma'_2) - Z(\sigma_1) - Z(\sigma_2) < 0.$$
Lower bounds on moves

- Let $C^{\text{MIN}}(\sigma) = \min_{k,l} \{ C(\sigma)[k,l] \}$ the shortest path for the sequence $\sigma$ between any pair of origin/end orientations.

- Let $c_{ij}^{\text{MIN}} = \min_{k,l} \{ c_{ij}^{kl} \}$ be the minimum cost of a shortest path between services $i$ and $j$, for any orientation.

- Lower bound on the cost of a route $\sigma = \sigma_1 \oplus \cdots \oplus \sigma_X$ composed of a concatenation of $X$ sequences:

  $$\bar{Z}(\sigma_1 \oplus \cdots \oplus \sigma_X) = \sum_{j=1}^{X} C^{\text{MIN}}(\sigma_j) + \sum_{j=1}^{X-1} c_{\sigma_j,\sigma_{j+1}}^{\text{MIN}}.$$ 

- The bound helps to filter a lot of moves ($\geq 90\%$ even when used with granular search)
  - In practice: possible to evaluate a move in the space of service permutations for the CARP with roughly the same number of elementary operations as the same move for a CVRP!
1 Node and edge routing problems
2 Combined neighborhoods for arc routing problems
   • Work rationale and shortest path formulation
   • Cutting off complexity: memories + bidirectional search
   • Cutting off complexity: moves filtering via LBs
3 Problem generalizations
4 Towards “very very” large neighborhoods
5 Computational experiments
   • Integration into two state-of-the-art metaheuristics
   • Comparison with previous literature
   • CARP – To reduce or not to reduce
   • Problems with turn penalties and delays at intersections
6 Conclusions/Perspectives
Some preliminary definitions

- **Service**: A visit to a client, which cannot be split, but may be operated in different alternative ways.

- **Service Mode**: Alternative way to perform a service, may impact travel or service cost.
  \[\Rightarrow\] The set of possible *modes* for a service will be notated \(M_i\).
Generalizations via enriched mode definitions

- **CARP** – each service has two modes, one for each possible orientation (curb direction during service).

- Many other mode choices in problem variants:
  - choice of sidewalk and impact on intersection time (postal delivery, refuse collection)
  - lane (snow plowing)
  - parking spot
  - choice of visit location (GVRP and arc routing equivalents)
  - orders of visit clusters, e.g., in a city district (CluVRP and arc routing equivalents)
  - entry-exit of a facility...
Generalizations via enriched mode definitions

• Now, **node, edge and arc routing problems are greatly simplified:**

  - **Node** \(|M_i| = 1\) One mode for service;
  - **Arc** \(|M_i| = 1\) One mode for the only feasible service orientation;
  - **Edge** \(|M_i| = 2\) Two modes, one for each service orientation.

• Route-evaluation subproblem even more efficient since many services are now represented as a single node in the auxiliary graph
Generalizations via enriched mode definitions

- Problems with **turn penalties and delays at intersections** are greatly simplified:
- In previous literature – feasibility issues:
  - Solution of NEARP with turn penalties represented as sequences of services + SPs with turn restrictions between services did not necessarily lead to viable solutions:
  - Because of a **lack of characterization of the arrival edge** when servicing a node.
Generalizations via enriched mode definitions

• The needed information can be included in the definition of the mode:

| NODE | $|M_i| = p_i$ | $p_i$ modes to specify the arrival direction, where $p_i$ is the in-degree of $v_i$; |
| ARC | $|M_i| = 1$ | One mode for the only feasible service orientation; |
| EDGE | $|M_i| = 2$ | Two modes, one for each service orientation. |

• Then, turn penalties can easily be included in arc costs, in the auxiliary graph

• Done $\Rightarrow$ turn penalties are now optimally addressed (for any fixed sequence of services) without any further change
Problems with service clusters are greatly simplified:

- Problems with choices of service location (Generalized routing problems – GVRP) are greatly simplified...

- But also, inserting a break, going to an intermediate facility, recharging electric vehicles... are many ways of choosing a mode when servicing a customer.
  
  ▶ Keep in mind that in these cases, other resources than cost may be involved ⇒ RCSPs...
1 Node and edge routing problems

2 Combined neighborhoods for arc routing problems
   - Work rationale and shortest path formulation
   - Cutting off complexity: memories + bidirectional search
   - Cutting off complexity: moves filtering via LBs

3 Problem generalizations

4 Towards “very very” large neighborhoods

5 Computational experiments
   - Integration into two state-of-the-art metaheuristics
   - Comparison with previous literature
   - CARP – To reduce or not to reduce
   - Problems with turn penalties and delays at intersections

6 Conclusions/Perspectives
The concept can even be integrated into ejection chains-type neighborhoods to search an \textbf{exponential set of solutions} (visit permutations + depots) \textbf{in polynomial time} via a shortest-path formulation:
• The cost $c_{ij}$ of an arc $(i, j)$ corresponds to the difference of cost of $R(j)$ when removing service $j$ and inserting service $i$ with minimum cost in the route.
“Very very” large neighborhoods

- Using this problem decomposition and route evaluation procedure in the “Split” algorithm leads to another very large neighborhood.

- Still in $O(n^2)$
- Already known as Split “with flips” from Prins et al. (2009).
Contents

1 Node and edge routing problems
2 Combined neighborhoods for arc routing problems
   - Work rationale and shortest path formulation
   - Cutting off complexity: memories + bidirectional search
   - Cutting off complexity: moves filtering via LBs
3 Problem generalizations
4 Towards “very very” large neighborhoods
5 Computational experiments
   - Integration into two state-of-the-art metaheuristics
   - Comparison with previous literature
   - CARP – To reduce or not to reduce
   - Problems with turn penalties and delays at intersections
6 Conclusions/Perspectives
Integration into two state-of-the-art metaheuristics:

The iterated local search variant (ILS) of Prins (2009)
- Produces $n_C$ offspring from the incumbent solution and selects the best
- Search is restarted $n_P$ times, each run terminates after $n_I$ consecutive iterations
- I added the possibility to use penalized infeasible solutions (not in the original version of the algorithm).

The unified hybrid genetic search (UHGS) of Vidal et al. (2012, 2014)
UHGS

Classic genetic algorithm components: population, selection, crossover, and

1. Efficient **local-improvement** procedure. Replaces random mutation

2. Management of **penalized infeasible solutions**

3. Individual evaluation: **solution quality** and **contribution to population diversity**
Local improvement procedure used in both methods:

Very standard neighborhoods:

- **RELOCATE**, **SWAP**, **CROSS**, **2-opt** and **2-opt***.
  - Exploration in random order
  - First improvement policy
  - Restrictions of moves to $K^{th}$ closest customers
    ⇒ Number of neighbors in $\mathcal{O}(n)$
  - + one attempt of ejection chain on any local minimum.

Penalized infeasible solutions:

- Simple linear combination of the excess of load, distance or other resource constraints on routes.
  - Penalty coefficients are adapted during the search.
Metaheuristics

UHGS – Biased fitness: combining ranks in terms of solution cost $C(I)$ and contribution to the population diversity $D(I)$, measured as a distance to other individuals:

$$BF(I) = C(I) + \left(1 - \frac{nbElite}{popSize - 1}\right) D(I)$$

- Used for parents selection
  - Balancing quality with innovation to promote a more thorough exploration of the search space.
- Used during selection of survivors
  - Removing individuals with worst $BF(I)$ still guarantees elitism in terms of solution quality.

![Graph showing the tradeoff between fitness and diversity](image)
Experimental setting

- Literature on CARP and NEARP built around several sets of well-known benchmark instances:

|     | #    | Reference                  | $|N_R|$ | $|E_R|$ | $|A_R|$ | $n$               | Specificities                                           |
|-----|------|----------------------------|-------|-------|-------|------------------|--------------------------------------------------------|
| CARP:                     |      |                            |       |       |       |                  |                                                        |
| VAL | (34) | Benavent et al. (1992)     | 0     | [39,97]| 0     | [39,97]         | Random graphs; Only required edges                      |
| BMCV| (100)| Beullens et al. (2003)     | 0     | [28,121]| 0     | [28,121]       | Intercity road network in Flanders                     |
| EGL | (24) | Li and Eglese (1996)       | 0     | [51,190]| 0     | [51,190]       | Winter-gritting application in Lancashire              |
| NEARP:                    |      |                            |       |       |       |                  |                                                        |
| MGGDB | (138)| Bosco et al. (2012)     | [3,16]| [1,9]| [4,31]| [8,48]         | From CARP instances GBD                                |
| MGVAL | (210)| Bosco et al. (2012)     | [7,46]| [6,33]| [12,79]| [36,129]       | From CARP instances VAL                                |
| CBMix | (23)| Prins and B. (2005)     | [0,93]| [0,94]| [0,149]| [20,212]       | Randomly generated planar networks                     |
| BHW  | (20) | Bach et al. (2013)       | [4,50]| [0,51]| [7,380]| [20,410]       | From CARP instances GDB, VAL, & EGL                    |
| DI-NEARP | (24)| Bach et al. (2013) | [120,347]| [120,486]| 0     | [240,833]      | Newspaper and media product distribution               |
Experimental setting

- To prevent any possible over-tuning
  ⇒ using the original parameters of the metaheuristics
- Single core: Xeon 3.07 GHz CPU with 16 GB of RAM
- Single termination criterion on all instances
  ⇒ scaled to reach a similar CPU time as previous competitive algorithms.
Experimental setting

• For each benchmark set, we collected the best three solution methods in the literature (some are heavily tailored for specific benchmark sets).

| BLMV14 | Bosco et al. (2014) | LPR01 | Lacomme et al. (2001) | PDHM08 | Polacek et al. (2008) |
| BMCV03 | Beullens et al. (2003) | MLY14 | Mei et al. (2014) | TMY09 | Tang et al. (2009) |
| DHDII4 | Dell’Amico et al. (2014) | MPS13 | Martinelli et al. (2013) | UFF13 | Usberti et al. (2013) |

• Comparison with the proposed metaheuristics, which are searching the space of service permutations (our methods are not fine-tuned for any of these instance sets).
Experimental setting

- Reporting the average and best solution on 10 runs.
- All Gap(%) values measured from the current best known solutions (BKS)
- Warning – time measures for some previous algorithms: using known optimal solutions to trigger termination, or reporting the time to reach the best solution
  - Dependent on exogenous information
  - Not the complete search time
- Hence, two columns for time measures:
  ⇒ “T” for total CPU time when available,
  ⇒ “T*” for time to reach final solution.
## Comparison with previous literature

<table>
<thead>
<tr>
<th>Variant</th>
<th>Bench.</th>
<th>n</th>
<th>Author</th>
<th>Runs</th>
<th>Avg.</th>
<th>Best</th>
<th>T</th>
<th>T*</th>
<th>CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>CARP</td>
<td>GDB</td>
<td>30</td>
<td>TMY09</td>
<td>30</td>
<td>0.009%</td>
<td>0.000%</td>
<td>0.11</td>
<td>—</td>
<td>Xe 2.0G</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>BMCV03</td>
<td>1</td>
<td>0.000%</td>
<td>—</td>
<td>0.03</td>
<td>P- II 500M</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>MTY09</td>
<td>1</td>
<td>0.000%</td>
<td>—</td>
<td>0.01</td>
<td>Xe 2.0G</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>10</td>
<td>ILS</td>
<td>10</td>
<td>0.002%</td>
<td>0.000%</td>
<td>0.16</td>
<td>0.03</td>
<td>Xe 3.07G</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>UHGS</td>
<td>10</td>
<td>0.000%</td>
<td>0.000%</td>
<td>0.22</td>
<td>0.01</td>
<td>Xe 3.07G</td>
</tr>
<tr>
<td></td>
<td>VAL</td>
<td>1</td>
<td>MTY09</td>
<td>1</td>
<td>0.142%</td>
<td>—</td>
<td>0.11</td>
<td>Xe 2.0G</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>LPR01</td>
<td>1</td>
<td>0.126%</td>
<td>—</td>
<td>2.00</td>
<td>—</td>
<td>P-III 500M</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>BMCV03</td>
<td>1</td>
<td>0.060%</td>
<td>—</td>
<td>1.36</td>
<td>P- II 500M</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>10</td>
<td>ILS</td>
<td>10</td>
<td>0.054%</td>
<td>0.024%</td>
<td>0.68</td>
<td>0.16</td>
<td>Xe 3.07G</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>UHGS</td>
<td>10</td>
<td>0.048%</td>
<td>0.021%</td>
<td>0.82</td>
<td>0.08</td>
<td>Xe 3.07G</td>
</tr>
<tr>
<td></td>
<td>BMCV</td>
<td>1</td>
<td>BE08</td>
<td>1</td>
<td>0.156%</td>
<td>—</td>
<td>1.08</td>
<td>P- M 1.4G</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>MTY09</td>
<td>1</td>
<td>0.073%</td>
<td>—</td>
<td>0.35</td>
<td>Xe 2.0G</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>BMCV03</td>
<td>1</td>
<td>0.036%</td>
<td>—</td>
<td>2.57</td>
<td>—</td>
<td>P- II 450M</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10</td>
<td>ILS</td>
<td>10</td>
<td>0.027%</td>
<td>0.000%</td>
<td>0.82</td>
<td>0.22</td>
<td>Xe 3.07G</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>UHGS</td>
<td>10</td>
<td>0.007%</td>
<td>0.000%</td>
<td>0.87</td>
<td>0.11</td>
<td>Xe 3.07G</td>
</tr>
<tr>
<td></td>
<td>EGL</td>
<td>10</td>
<td>PDHM08</td>
<td>10</td>
<td>0.624%</td>
<td>—</td>
<td>30.0</td>
<td>8.39</td>
<td>P- IV 3.6G</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>UFF13</td>
<td>15</td>
<td>0.560%</td>
<td>0.206%</td>
<td>13.3</td>
<td>—</td>
<td>I4 3.0G</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>MTY09</td>
<td>1</td>
<td>0.553%</td>
<td>—</td>
<td>2.10</td>
<td>Xe 2.0G</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>10</td>
<td>ILS</td>
<td>10</td>
<td>0.236%</td>
<td>0.106%</td>
<td>2.35</td>
<td>1.33</td>
<td>Xe 3.07G</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>UHGS</td>
<td>10</td>
<td>0.153%</td>
<td>0.058%</td>
<td>4.76</td>
<td>3.14</td>
<td>Xe 3.07G</td>
</tr>
<tr>
<td></td>
<td>EGL-L</td>
<td>1</td>
<td>BE08</td>
<td>1</td>
<td>4.679%</td>
<td>—</td>
<td>17.0</td>
<td>—</td>
<td>P- M 1.4G</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>MPS13</td>
<td>10</td>
<td>2.950%</td>
<td>2.523%</td>
<td>20.7</td>
<td>—</td>
<td>I5 3.2G</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>MLY14</td>
<td>30</td>
<td>1.603%</td>
<td>0.895%</td>
<td>33.4</td>
<td>—</td>
<td>I7 3.4G</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10</td>
<td>ILS</td>
<td>10</td>
<td>0.880%</td>
<td>0.598%</td>
<td>23.6</td>
<td>15.4</td>
<td>Xe 3.07G</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>UHGS</td>
<td>10</td>
<td>0.645%</td>
<td>0.237%</td>
<td>36.5</td>
<td>27.5</td>
<td>Xe 3.07G</td>
</tr>
</tbody>
</table>
### Comparison with previous literature

<table>
<thead>
<tr>
<th>Variant</th>
<th>Bench.</th>
<th>n</th>
<th>Author</th>
<th>Runs</th>
<th>Avg.</th>
<th>Best</th>
<th>T</th>
<th>T*</th>
<th>CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>NEARP</td>
<td>MGGDB</td>
<td>[8,48]</td>
<td>BLMV14</td>
<td>1</td>
<td>1.342%</td>
<td>—</td>
<td>0.31</td>
<td>—</td>
<td>Xe 3.0G</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DHDII14</td>
<td>1</td>
<td>0.018%</td>
<td>—</td>
<td>60.0</td>
<td>0.86</td>
<td>CPU 3G</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ILS</td>
<td>10</td>
<td>0.010%</td>
<td>0.000%</td>
<td>0.13</td>
<td>0.03</td>
<td>Xe 3.07G</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>UHGS</td>
<td>10</td>
<td>0.015%</td>
<td>0.000%</td>
<td>0.16</td>
<td>0.01</td>
<td>Xe 3.07G</td>
</tr>
<tr>
<td></td>
<td>MGVAL</td>
<td>[36,129]</td>
<td>BLMV14</td>
<td>1</td>
<td>2.620%</td>
<td>—</td>
<td>16.7</td>
<td>—</td>
<td>Xe 3.0G</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DHDII14</td>
<td>1</td>
<td>0.071%</td>
<td>—</td>
<td>60.0</td>
<td>3.69</td>
<td>CPU 3G</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ILS</td>
<td>10</td>
<td>0.067%</td>
<td>0.019%</td>
<td>1.18</td>
<td>0.32</td>
<td>Xe 3.07G</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>UHGS</td>
<td>10</td>
<td>0.045%</td>
<td>0.011%</td>
<td>1.20</td>
<td>0.17</td>
<td>Xe 3.07G</td>
</tr>
<tr>
<td>NEARP</td>
<td>CBMix</td>
<td>[20,212]</td>
<td>HKSG12</td>
<td>2</td>
<td>—</td>
<td>3.076%</td>
<td>120</td>
<td>56.9</td>
<td>CPU 3G</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>BLMV14</td>
<td>1</td>
<td>2.697%</td>
<td>—</td>
<td>44.7</td>
<td>—</td>
<td>Xe 3.0G</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DHDII14</td>
<td>1</td>
<td>0.884%</td>
<td>—</td>
<td>60.0</td>
<td>19.6</td>
<td>CPU 3G</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ILS</td>
<td>10</td>
<td>0.733%</td>
<td>0.363%</td>
<td>2.46</td>
<td>1.48</td>
<td>Xe 3.07G</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>UHGS</td>
<td>10</td>
<td>0.381%</td>
<td>0.109%</td>
<td>4.56</td>
<td>3.08</td>
<td>Xe 3.07G</td>
</tr>
<tr>
<td>NEARP</td>
<td>BHW</td>
<td>[20,410]</td>
<td>HKSG12</td>
<td>2</td>
<td>—</td>
<td>1.949%</td>
<td>120</td>
<td>60.1</td>
<td>CPU 3G</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DHDII14</td>
<td>1</td>
<td>0.555%</td>
<td>—</td>
<td>60.0</td>
<td>21.4</td>
<td>CPU 3G</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ILS</td>
<td>10</td>
<td>0.440%</td>
<td>0.196%</td>
<td>5.22</td>
<td>2.90</td>
<td>Xe 3.07G</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>UHGS</td>
<td>10</td>
<td>0.208%</td>
<td>0.077%</td>
<td>7.95</td>
<td>5.87</td>
<td>Xe 3.07G</td>
</tr>
<tr>
<td>NEARP</td>
<td>DI-NEARP</td>
<td>[240,833]</td>
<td>HKSG12</td>
<td>2</td>
<td>—</td>
<td>1.639%</td>
<td>120</td>
<td>93.0</td>
<td>CPU 3G</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DHDII14</td>
<td>1</td>
<td>0.536%</td>
<td>—</td>
<td>60.0</td>
<td>36.3</td>
<td>CPU 3G</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ILS</td>
<td>10</td>
<td>0.199%</td>
<td>0.084%</td>
<td>30.0</td>
<td>21.3</td>
<td>Xe 3.07G</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>UHGS</td>
<td>10</td>
<td>0.139%</td>
<td>0.055%</td>
<td>29.6</td>
<td>16.7</td>
<td>Xe 3.07G</td>
</tr>
</tbody>
</table>
Comparison with previous literature

- New neighborhoods lead to much better solutions → even ILS already produces better solutions than previous literature
- UHGS goes further in performance → up to 0.503% and 0.958% improvement on the large instance sets
- Some BKSs for large CARP instances have been improved by up to 2.275%
- Average standard deviation in [0.000%, 0.292%]
- On the CARP benchmark sets, 187/191 BKS have been matched or improved. 153/155 known optimal solutions were found
- For the NEARP, 408/409 BKS have been matched or improved. All 217 known optimal solutions found.
Comparison with previous literature

- Boxplot visualizations of Gap(%) of various methods on large-scale instances:
- Gray colors indicate a significant difference of performance, as highlighted by pairwise Wilcoxon tests with adequate correction for multiplicity

![Boxplot visualizations](image_url)

Set EGL

- PDHM08 x UHGS, P.value = 9e–05
- MTY09 x UHGS, P.value = 0.00053
- UPP13 x UHGS, P.value = 6e–05
- ILS x UHGS, P.value = 0.00044

Set EGL-L

- BE08 x UHGS, P.value = 0.00195
- MPS13 x UHGS, P.value = 0.00195
- ML Y14 x UHGS, P.value = 0.00195
- ILS x UHGS, P.value = 0.00195
Comparison with previous literature

Set CBMix

HKSG12 x UHGS, P.value = 6e−05
BLMV14 x UHGS, P.value = 9e−05
DHD114 x UHGS, P.value = 2e−04
ILS x UHGS, P.value = 0.00013

 HKSG12 BLMV14 DHD114 ILS UHGS
0 1 2 3 4 5 6 7 HKSG12 x UHGS, P.value = 6e−05
BLMV14 x UHGS, P.value = 9e−05
DHD114 x UHGS, P.value = 2e−04
ILS x UHGS, P.value = 0.00013

Set BHW

HKSG12 x UHGS, P.value = 0.00065
DHD114 x UHGS, P.value = 0.00298
ILS x UHGS, P.value = 0.00233

 HKSG12 DHD114 ILS UHGS
0 1 2 3 4 5 6 HKSG12 x UHGS, P.value = 0.00065
DHD114 x UHGS, P.value = 0.00298
ILS x UHGS, P.value = 0.00233

Set DI-NEARP

HKSG12 x UHGS, P.value = 0
DHD114 x UHGS, P.value = 7e−05
ILS x UHGS, P.value = 0.00842

 HKSG12 DHD114 ILS UHGS
0 1 2 3 4 HKSG12 x UHGS, P.value = 0
DHD114 x UHGS, P.value = 7e−05
ILS x UHGS, P.value = 0.00842
Scalability

- Growth of the CPU time of UHGS as a function of the number of services, for the CARP instances (left figure) and NEARP instances (right figure). Log-log scale.

- A linear fit, with a least square regression, has been performed on the sample after logarithmic transformation:
  \[ f(n) = 0.00027n^{1.95970} \]

\[ T(n) \]

\[ f(n) = 0.00035n^{1.89167} \]

\[ T(n) \]

\[ \Rightarrow \text{CPU time appears to grow in } \Theta(n^2) \]
To reduce or not to reduce

- Previous slides: investigated whether methods using combined neighborhoods – with optimal choices of service orientations – can outperform methods based on more traditional neighborhoods.

- Now analyzing whether relying on a problem reduction from CARP to CVRP (Martinelli et al., 2013) with a classical routing metaheuristic can be profitable.

- The reduction increases the number of services by $\times 2$.
  - Half of the edges of a CVRP solution, with a large fixed negative cost, directly determine the service orientations in the associated CARP solution.
To reduce or not to reduce

- Applied the same ILS and UHGS on the transformed instances, now using a classical move evaluation for the CVRP.

<table>
<thead>
<tr>
<th></th>
<th>Gap(%) ILS</th>
<th>ILS&lt;sub&gt;CVRP&lt;/sub&gt;</th>
<th>T(min) ILS</th>
<th>ILS&lt;sub&gt;CVRP&lt;/sub&gt;</th>
<th>GDB</th>
<th>0.002%</th>
<th>0.000%</th>
<th>0.16</th>
<th>0.59</th>
<th>0.22</th>
<th>0.72</th>
</tr>
</thead>
<tbody>
<tr>
<td>VAL</td>
<td>0.054%</td>
<td>0.061%</td>
<td>0.68</td>
<td>2.39</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BMCV</td>
<td>0.027%</td>
<td>0.044%</td>
<td>0.82</td>
<td>2.79</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EGL</td>
<td>0.236%</td>
<td>0.345%</td>
<td>2.35</td>
<td>8.50</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EGL-L</td>
<td>0.880%</td>
<td>1.411%</td>
<td>23.6</td>
<td>60.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Gap(%) UHGS</td>
<td>UHGS&lt;sub&gt;CVRP&lt;/sub&gt;</td>
<td>T(min) UHGS</td>
<td>UHGS&lt;sub&gt;CVRP&lt;/sub&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GDB</td>
<td>0.000%</td>
<td>0.000%</td>
<td>0.22</td>
<td>0.72</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VAL</td>
<td>0.048%</td>
<td>0.048%</td>
<td>0.82</td>
<td>2.98</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BMCV</td>
<td>0.007%</td>
<td>0.014%</td>
<td>0.87</td>
<td>3.02</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EGL</td>
<td>0.153%</td>
<td>0.200%</td>
<td>4.76</td>
<td>12.65</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EGL-L</td>
<td>0.645%</td>
<td>1.001%</td>
<td>36.5</td>
<td>59.7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Significantly lower solution quality and higher CPU time when relying on the decomposition.

- Heuristics for the CARP are worth studying...
Addressing problems with turn penalties

- Final experiment about CARP and NEARP with turn penalties
  - A must-have in various sectors of application, but more scarcely studied in the routing community.

- Lack of reasonable benchmark sets, previous instances based on random graphs:
Addressing problems with turn penalties

- Hence, also generating new benchmark instances to investigate the problem

- Extension of DI-NEARP (Bach et al., 2013), adding turn penalties ⇒ 28 instances with 240–833 services.
  - Application of media products distribution in Nordic countries
  - Edge distances are available but no node coordinates

- How to produce realistic turn penalties?
  - Reconstructing a plausible planar layout for each instance, with the FM³ algorithm of Hachul and Jünger (2005) ⇒ efficiently evaluates a force equilibrium, based on desired distances to obtain 2D node coordinates
  - $5\gamma$ for U-turns, $3\gamma$ for left turns, $\gamma$ for intersection crossing
  - $\gamma$ calibrated for turn penalties to scale to 30% of solution cost, (realistic according to analyses of Nielsen et al. 1998)
Addressing problems with turn penalties

- Sample solution with small turn penalties:
  - $\gamma = 0.25$, distance = 4286:
Addressing problems with turn penalties

- Sample solution with slightly larger turn penalties:
  - $\gamma = 0.5$, distance of 4336:
## Addressing problems with turn penalties

<table>
<thead>
<tr>
<th>$\gamma$</th>
<th>Gap (%)</th>
<th>T</th>
<th>Cost</th>
<th>Distance</th>
<th>Nb Turns</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>U-turns</td>
</tr>
<tr>
<td>0</td>
<td>0.141%</td>
<td>50.68</td>
<td>25076.61</td>
<td>25076.61</td>
<td>126.24</td>
</tr>
<tr>
<td>0.25</td>
<td>0.280%</td>
<td>51.32</td>
<td>27500.70</td>
<td>25164.44</td>
<td>119.40</td>
</tr>
<tr>
<td>0.5</td>
<td>0.281%</td>
<td>51.65</td>
<td>29806.22</td>
<td>25250.74</td>
<td>116.79</td>
</tr>
<tr>
<td>1</td>
<td>0.373%</td>
<td>51.74</td>
<td>34339.29</td>
<td>25451.40</td>
<td>113.87</td>
</tr>
<tr>
<td>2</td>
<td>0.511%</td>
<td>51.77</td>
<td>43103.49</td>
<td>25986.19</td>
<td>109.84</td>
</tr>
<tr>
<td>5</td>
<td>0.607%</td>
<td>51.90</td>
<td>68258.91</td>
<td>27243.48</td>
<td>106.31</td>
</tr>
<tr>
<td>10</td>
<td>0.752%</td>
<td>51.92</td>
<td>109011.41</td>
<td>28534.13</td>
<td>105.23</td>
</tr>
</tbody>
</table>

- To assess method performance, Gap(%) measured between average solutions and BKS produced by long runs.
- Gap and standard deviation remain moderate, usually good sign
- CPU time is moderate ($\approx 50\text{min}$ for 833 services).
  - Straightforward parallelization, or reduction of termination criterion if more speed is needed.
Addressing problems with turn penalties

- Turn penalties seem to lead to slightly more difficult problems.
- Remarkable reductions of left turns or U-turns even with very small penalties.
- A few turns cannot be avoided, due to the graph topology.
<table>
<thead>
<tr>
<th>1</th>
<th>Node and edge routing problems</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Combined neighborhoods for arc routing problems</td>
</tr>
<tr>
<td></td>
<td>Work rationale and shortest path formulation</td>
</tr>
<tr>
<td></td>
<td>Cutting off complexity: memories + bidirectional search</td>
</tr>
<tr>
<td></td>
<td>Cutting off complexity: moves filtering via LBs</td>
</tr>
<tr>
<td>3</td>
<td>Problem generalizations</td>
</tr>
<tr>
<td>4</td>
<td>Towards “very very” large neighborhoods</td>
</tr>
<tr>
<td>5</td>
<td>Computational experiments</td>
</tr>
<tr>
<td></td>
<td>Integration into two state-of-the-art metaheuristics</td>
</tr>
<tr>
<td></td>
<td>Comparison with previous literature</td>
</tr>
<tr>
<td></td>
<td>CARP – To reduce or not to reduce</td>
</tr>
<tr>
<td></td>
<td>Problems with turn penalties and delays at intersections</td>
</tr>
<tr>
<td>6</td>
<td>Conclusions/Perspectives</td>
</tr>
</tbody>
</table>
Conclusions

- Studied a neighborhood that was scarcely used in the past ⇒ leads to a decomposition of problem structure, to **conceal arc routing difficulties**
- We made is efficient, systematic and general
- Interesting complexity properties → a kind of “free lunch”.
- Many opportunities of problem generalizations
- State-of-the-art results for all known CARP and NEARP benchmark sets
- Connecting further arc and node routing worlds
Perspectives

- Open doors for research
- New instances for problems with turn penalties, challenging
- Perspectives: look for similar structural decompositions
  ⇒ cases with more resources
  ⇒ other combinatorial optimization problems
  ⇒ further connections with branch-cut-price
Thank you for your attention!

Technical report, instances, detailed results and slides available at:

And references after this slide...


