A SIMULATION-BASED METHODOLOGY TO ASSIST DECISION-MAKERS IN VEHICLE ROUTING PROBLEMS

Angel A. Juan, Javier Faulin, Daniel Riera, David Masip & Josep Jorba

http://ajuanp.wordpress.com | ajuanp@uoc.edu

Dep. of Computer Science, Multimedia and Telecommunication
Open University of Catalonia, Barcelona, SPAIN
Outline

- Introduction: The VRP
- Motivation
- Related work: The CWS heuristic
- Our approach: Randomizing the CWS heuristic
- Our approach: Further improvements
- Software implementation
- Validation
- Future work
- Conclusions
1. Introduction: The VRP

- The Vehicle Routing Problem (VRP) is a c. p. (Golden et al 2008):
  - A set of customers’ demands must be supplied by a fleet of vehicles.
  - Resources are available from a depot.
  - Moving a vehicle from one node $i$ to another $j$ has associated costs $c(i, j)$
  - Additional constraints must be considered: maximum load capacity per vehicle, etc.

- Goal 1 (classical): to obtain an optimal solution, i.e. a set of routes satisfying all constraints with minimum costs

- Different approaches: optimization methods (small-size), heuristics (CWS) and meta-heuristics (GAs, TS, SA, …)
2. Motivation

- Yes, but…: In real-life scenarios is not possible to model all costs, constraints and desirable solution properties in advance (Kant et al 2008)
- Goal 2 (our approach): to develop a method that also provides many ‘good’ alternative solutions, so that the decision-maker can select the one that best fits her utility function.
3. Related work: The CWS heuristic

- Our approach will be based on the Clarke and Wright’s savings (CWS) algorithm (Clarke & Wright 1964).

- **CWS algorithm:**
  - For each pair of nodes $i$ and $j$, calculate the savings, $s(i, j)$, associated to the edge connecting them, where:
    $$ s(i, j) = c(0, i) + c(0, j) - c(i, j) $$
  - Construct a list of edges, sorting the edges according to their associated savings
  - Construct an initial feasible solution by routing a vehicle to each client node
  - Select the first edge in the savings list and, if no constraint is violated, merge the routes that it connects
  - Repeat step 4 until the savings list is empty

- This parallel version of the CWS heuristic usually provides ‘acceptable solutions’ (average gap between 5% and 10%), especially for small and medium-size problems
4. Our approach: Randomizing the CWS algorithm

- **CWS** → the **first edge** (the one with the most savings) is the one selected.

- **SR-GCWS** introduces randomness in this process by using a quasi-geometric statistical distribution → edges with more savings will be more likely to be selected at each step, but all edges in the list are potentially eligible.

- **Notice:** Each time SR-GCWS is run, a random feasible solution is obtained. By construction, chances are that this solution outperforms the CWS one → hundreds of ‘good’ solutions can be obtained after some seconds/minutes.

\[
\forall k = 1, 2, \ldots, s
\]

\[
P(X = k) = a \cdot (1 - a)^{k-1} + \varepsilon
\]

\[
\varepsilon = \sum_{k=1}^{s} a \cdot (1 - a)^{k-1} = 1 - \sum_{k=1}^{s} a \cdot (1 - a)^{k-1}
\]
5. Our approach: Further improvements

- Adding ‘memory’ to our algorithm with a hash table:
  - A hash table is used to save, for each generated route, the best-known sequence of nodes (this will be used to improve new solutions)
  - ‘Fast’ method that provides small improvements on the average

2. Splitting (divide-and-conquer) method:
  - Given a global solution, the instance is sub-divided in smaller instances and then the algorithm is applied on each of these smaller instances
  - ‘Slow’ method that can provide significant improvements
6. Software implementation

- **OO approach** (Java, Eclipse)
- Special attention:
  - RNG (L’Ecuyer 2001) → SSJ library (L’Ecuyer 2002), GenF2W32 period $2^{800}-1$
  - Design of classes (Horstmann 2006)
  - Code accuracy and effectiveness
- Implementation of the CWS heuristic (parallel version) based on:

  [Link](http://web.mit.edu/urban_or_bc)

Both the CWS and the SR-GCWS-CS implementations have been **verified** by using standard benchmarks and independent calculations.
7. Validation (1/2)

- To verify the goodness of our approach and its efficiency, a total of 50 classical VRP benchmark instances were randomly selected from http://www.branchandcut.org (which also contains best-known solutions so far)

Different Scenarios:
- From 45 to 200 nodes
- Different topologies (depot, clusters, etc.)

- Results:
  - 31-out-of-50 instances offer a negative gap – i.e., they outperform the BKS
  - The remaining 19 instances offer a null gap
  - Average gap = -0.21%
  - In most cases → few seconds
7. Validation (2/2)

Intel® Core™2 Duo CPU at 2.4 GHz and 2 GB RAM

A positive gap implies that the CWS solution costs are higher than the ones associated with the best-known-so-far solution.

Our approach improves **31-out-of-50** benchmark solutions, with a global **average gap of -0.21%** for the 50 benchmark instances.

<table>
<thead>
<tr>
<th>Instance</th>
<th>Nodes</th>
<th>CWS-p solution (1)</th>
<th>Gap (1) - (2)</th>
<th>Best-known solution* (2)</th>
<th>SR-GCWS solution (3)</th>
<th>Gap (2) - (3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A-n45-k7</td>
<td>45</td>
<td>1,199.98</td>
<td>4.59%</td>
<td>1,147.28</td>
<td>1,146.91</td>
<td>-0.03%</td>
</tr>
<tr>
<td>A-n60-k9</td>
<td>60</td>
<td>1,421.88</td>
<td>4.87%</td>
<td>1,355.80</td>
<td>1,355.80</td>
<td>0.00%</td>
</tr>
<tr>
<td>A-n80-k10</td>
<td>80</td>
<td>1,860.94</td>
<td>5.35%</td>
<td>1,766.50</td>
<td>1,766.50</td>
<td>0.00%</td>
</tr>
<tr>
<td>B-n50-k7</td>
<td>50</td>
<td>748.80</td>
<td>0.54%</td>
<td>744.78</td>
<td>744.23</td>
<td>-0.07%</td>
</tr>
<tr>
<td>B-n52-k7</td>
<td>52</td>
<td>764.90</td>
<td>1.98%</td>
<td>750.08</td>
<td>749.97</td>
<td>-0.01%</td>
</tr>
<tr>
<td>B-n57-k9</td>
<td>57</td>
<td>1,653.42</td>
<td>3.10%</td>
<td>1,603.63</td>
<td>1,602.29</td>
<td>-0.08%</td>
</tr>
<tr>
<td>B-n78-k10</td>
<td>78</td>
<td>1,264.56</td>
<td>2.87%</td>
<td>1,229.27</td>
<td>1,228.16</td>
<td>-0.09%</td>
</tr>
<tr>
<td>E-n51-k5</td>
<td>51</td>
<td>584.64</td>
<td>11.37%</td>
<td>524.94</td>
<td>524.61</td>
<td>-0.06%</td>
</tr>
<tr>
<td>E-n76-k10</td>
<td>76</td>
<td>900.26</td>
<td>7.51%</td>
<td>837.36</td>
<td>839.13</td>
<td>0.21%</td>
</tr>
<tr>
<td>E-n76-k14**</td>
<td>76</td>
<td>1,073.43</td>
<td>4.55%</td>
<td>1,026.71</td>
<td>1,026.14</td>
<td>-0.06%</td>
</tr>
<tr>
<td>F-n135-k7</td>
<td>135</td>
<td>1,219.32</td>
<td>4.16%</td>
<td>1,170.65</td>
<td>1,170.33</td>
<td>-0.03%</td>
</tr>
<tr>
<td>M-n121-k7</td>
<td>121</td>
<td>1,068.14</td>
<td>2.20%</td>
<td>1,045.16</td>
<td>1,045.60</td>
<td>0.04%</td>
</tr>
<tr>
<td>M-n200-k7</td>
<td>200</td>
<td>1,395.74</td>
<td>6.10%</td>
<td>1,315.43</td>
<td>1,313.71</td>
<td>-0.13%</td>
</tr>
<tr>
<td>P-n70-k10</td>
<td>70</td>
<td>896.86</td>
<td>10.56%</td>
<td>830.02</td>
<td>831.81</td>
<td>0.22%</td>
</tr>
<tr>
<td>P-n101-k4</td>
<td>101</td>
<td>765.38</td>
<td>8.05%</td>
<td>692.28</td>
<td>691.29</td>
<td>-0.14%</td>
</tr>
</tbody>
</table>

**Table 1. Comparison of methodologies for the fifteen selected CVRP instances**

(*) Best-known solution according to the information available at [http://www.branchandcut.org/](http://www.branchandcut.org/)

(**) For this instance our best solution employs a total of 15 routes

A negative gap implies that our solution costs are lower than the ones associated with the best-known-so-far solution.
8. Future work

- Tesla GPUs (CUDA) → about 12,000 threads (for real-time solutions & large-size CVRPs)
- VRPTW, VRPSD, …
- Scheduling problems
- Splitting with AI techniques
- Hybridization with CP techniques
- …
Conclusions

- SR-GCWS-CS is a hybrid algorithm that combines CWS, MCS and splitting techniques to efficiently solve the VRP.
- It also provides a set of alternative ‘good’ solutions with different properties → decision-makers do not receive a single ‘optimal’ solution, but they can choose the solution which best fits their utility function (which usually cannot be modeled in advance).
- The algorithm is relatively simple to understand and to implement. Moreover, it is already designed to be used in parallel processing (different instances of the algorithm using different RNG seeds).
- It is more flexible than other approaches, since it does not require fine-tuning processes. This makes it especially suitable to deal with realistic scenarios with more constraints (VRPTW, VRPSD, etc.)
- Similar algorithms can be developed for NP-hard problems other than the VRP (e.g. Scheduling problems)
A SIMULATION-BASED METHODOLOGY TO ASSIST DECISION-MAKERS IN VEHICLE ROUTING PROBLEMS

Thanks for your attention

Angel A. Juan, Javier Faulin, Daniel Riera, David Masip & Josep Jorba

http://ajuanp.wordpress.com | ajuanp@uoc.edu

Dep. of Computer Science, Multimedia and Telecommunication

Open University of Catalonia, Barcelona, SPAIN