Combining acceleration techniques for pricing in a VRP with time windows

Y. Arda, H. Küçükaydin, S. Michelini

Université de Liège
HEC Liège, QuantOM

ORBEL 30
January 28th, 2016
Table of Contents

1. Introduction
2. Relaxation techniques
3. Observations and future work
The problem

- A variant of the capacitated VRP with time windows
- Additional features:
  - Route cost depends on total route duration
  - Variable starting time for each route
  - Max allotted time for each route
- Minimization of the overall waiting time is part of the objective
- We choose to apply a branch-and-price methodology.
- The pricing problem is an elementary shortest path problem with resource constraints (ESPPRC)\(^1\)

\(^1\)Proven to be NP-Hard (Dror 1994)
Dynamic programming for the ESPPRC

- For every subpath from the source $s$ to a node $i$, we associate a label $L_i = (C_i, R_i, S_i)$, where:
  - $C_i$ is the cumulated cost
  - $R_i$ is the array of resources consumed along the subpath
    - In the case of the classic VRPTW, $R_i = (Q_i, T_i)$, where $Q_i$ is the total demand satisfied and $T_i$ is the total duration of the subpath
    - We impose $Q_i \leq Q_{\text{max}}$ and $a_i \leq T_i \leq b_i$
  - $S_i$ is a 0-1 $n$-sized array that keeps track of the visited nodes
- To extend a subpath $s - \cdots - i$ to a node $j$, simply use $L_i$ to compute the values of a new label $L_j$
- If a resource in $R_j$ is out of bounds or $S^j_i = 1$, the extension is infeasible and $L_j$ is rejected
- After performing all possible extensions, the best label $L_t$ at the sink $t$ is the solution
Dynamic programming: improvements

- Label dominance: given $L_i = (C_i, R_i, S_i)$ and $L'_i = (C'_i, R'_i, S'_i)$, if $C_i \leq C'_i$, $R_i \leq R'_i$, $S_i \leq S'_i$ and at least one inequality is strict, then $L_i$ dominates $L'_i$.

- Bounded bidirectional DP: perform forward extensions from the source and backwards extensions from the sink. Use a resource in $R_i$ to bound the search (e.g. no label with $Q_i > Q_{\text{max}}/2$ is extended).

- If an extension of a Label $L_i$ to node $j$ is infeasible, mark the unreachable node as visited, i.e. put $S'_i = 1$, to increase the number of dominated labels.
Adapting dynamic programming

- For the VRPTW with variable start times, we need to deal with an infinite number of Pareto-optimal states.
- We solve this by adapting the label structure and extension rules.
- We define \( R_i = (Q_i, T_i, -L_i, E_i) \), where:
  - \( T_i \) is the cumulative travel time from \( s \) to \( i \): \( T_j = T_i + t_{ij} \)
  - \( L_i \) is the latest feasible start time from \( s \): \( L_j = \min\{L_i, b_j - T_j\} \)
  - \( E_i \) is the earliest feasible arrival time at \( i \): \( E_j = \max\{a_j, a_i + t_{ij}\} \)
- Furthermore \( C_i = \max\{T_i, E_i - L_i\} - \sum_{k=s}^{i} \eta_k \), where \( \eta_k \) is the dual price associated with \( k \).
- It is then still possible\(^2\) to check \( R_i \leq R'_i \) to see if \( L_i \) dominates \( L'_i \).

\(^2\) Arda, Crama, and Kucukaydin 2014.
Relaxation techniques

- We focus on techniques that relax the elementarity constraints, i.e. manipulate the array $S_i$:
- **Decremental state space relaxation (DSSR)**\(^3\)
- **ng-route relaxation**\(^4\)
- Possible hybrid strategies

---

\(^3\)Righini and Salani 2008.
\(^4\)Baldacci et al. 2010.
Decremental State Space Relaxation

- In **State Space Relaxation**\(^5\), we project the state-space \( S \) used in DP to a lower dimensional space \( T \), so that the new states retain the cost.

- When applying this to the elementarity constraints, the number of states to explore is reduced, at the cost of feasibility.

- **Decremental** State Space Relaxation (DSSR) is a generalization of both this method and DP with elementarity constraints.

- We maintain a set \( \Theta \) of **critical** nodes on which the elementarity constraints are enforced at each iteration of DP.

- If at the end of DP the optimal path is not feasible, we update \( \Theta \) with the nodes that are visited multiple times.

---

\(^5\)Christofides, Mingozzi, and Toth 1981.
DSSR: Initialization strategies

We can initialize the set $\Theta$ with nodes that are likely to be critical

“Cycling attractiveness” $f_{ij}$ of a node $i$ with respect to a vertex $j$:

$$f_{ij} = \eta_i / (\bar{t}_{ij} + \bar{t}_{ji}).$$

Derived measures:

1. Highest cycling attractiveness (HCA): $\max_{j \in V \setminus \{i\}} f_{ij}$;
2. Total cycling attractiveness (TCA): $\sum_{j \in V \setminus \{i\}} f_{ij}$;
3. Weighted HCA (WHCA): $\max_{j \in V \setminus \{i\}} f_{ij} (b_i - a_i)$;
4. Weighted TCA (WTCA): $\sum_{j \in V \setminus \{i\}} f_{ij} (b_i - a_i)$.

We can rank each node according to any of these measures and initialize $\Theta$ with the best $m$ nodes

In a “mixed” strategy, $\Theta = HCA_m \cap TCA_m \cap WHCA_m \cap WTCA_m$

---

$^6$Righini and Salani 2009.
Strategies when enforcing elementarity on the optimal path\textsuperscript{7}

- HMO (highest multiplicity on the optimal path): insert one node at a time, selecting the node that is visited the most. In case of \textit{ex aequo}, choose at random;
- HMO-All: insert all nodes visited the maximum number of times;
- MO-All (multiplicity greater than one on the optimal path): insert all nodes visited more than once in the optimal path.

\textsuperscript{7}Boland, Dethridge, and Dumitrescu 2006.
DSSR: Insertion strategies

- How to generalize and parametrize these strategies?
- At every iteration of column generation we might want to insert up to $N_{col}$ columns
- If the optimal path is not elementary, check violations on:
  1. Only the optimal path
  2. The best $N_{COL}$ paths
  3. The best $k$ paths, $1 \leq k \leq N_{col}$
- For each path $P$ to check, either:
  1. Select the most visited node;
  2. Select all $M_P$ nodes visited multiple times;
  3. Select the $\lceil \alpha M_P \rceil$ most visited nodes, $0 < \alpha < 1$. 
**ng-route relaxation**

- For each node $i$ we define a neighbourhood $N_i$.
- An **ng-route can** contain any cycle of the form $i \rightarrow \cdots \rightarrow j \rightarrow \cdots \rightarrow i$ **only if** it contains a vertex $j$ such that $i \notin N_j$.
- For a subpath $s \rightarrow \cdots \rightarrow i$, $S_i$ represents the “memory” of the visited nodes.
- When extending from $i$ to $j$ we “forget” the nodes that are not in $N_j$.

**Example**

$P = 0 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 4$

$S_3 = \{0, 1, 2, 3\}$

$N_4 = \{2, 3, 4, 5\}$

$S_4 = (S_3 \cap N_4) \cup \{4\} = \{2, 3, 4\}$

$P = 0 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 1$ would be therefore valid.
**ng-route relaxation parameters**

- **Measure according to which we build \( N_i \):**
  1. **Travel time:**
     \[
     D_1(i, j) := t_{ij}, \quad \forall j \neq i;
     \]
  2. **Minimum travel duration:**
     \[
     D_2(i, j) := \max\{D'_ij, D'_ji\}, \text{ where}
     \]
     \[
     D'_ij := \begin{cases} 
     \max\{t_{ij}, a_j - b_i\} & \text{if } a_i + \bar{t}_{ij} \leq b_j \\
     +\infty & \text{otherwise};
     \end{cases}
     \]
  3. **Mixed measure:**
     \[
     D_3(i, j) := \beta D_1(i, j) + (1 - \beta)D_2(i, j), \text{ with } 0 < \beta < 1
     \]

- **The size \( m_{ng} \) of the neighbourhoods,** \( 1 \leq m_{ng} \leq n \)
Hybrid techniques

- Can we combine DSSR and \textit{ng}-route relaxation?
- For a straightforward combination, ignore nodes with multiple visits if they are in a valid \textit{ng}-cycle
- We apply DSSR locally, with respect to each neighbourhood:\footnote{Martinelli, Pecin, and Poggi 2014.}
  - Maintain “applied” neighbourhoods $\hat{N}_i \subseteq N_i \ \forall i$, initialized as empty
  - Use them during label extension instead of $N_i$
  - For every invalid cycle $C = i - \cdots - i$, add $i$ to all $\hat{N}_j$ such that $j \in C$

**Example**

$$P = 0 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 2 \quad N_3 = \{2, 3, 4\}, \quad N_4 = \{2, 3, 4, 5\}$$

$$\hat{N}_3 = \hat{N}_4 = \emptyset \implies \hat{N}_3 = \hat{N}_4 = \{2\}$$
Further possible hybridizations

- In the first hybrid strategy, nodes can be seen as critical in a global sense.
- In the second, nodes are critical with respect to other nodes.
- \(ng\)-routes are not guaranteed to be elementary.

Possible techniques:

- Implement a *local* DSSR, using critical sets \(\Theta_i\); \(\forall i\)
- Corrected \(ng\)-route relaxation: if the desired routes are not elementary, mark the nodes visited multiple times as critical.

We end up with 3 possible \(ng\)-route techniques and 6 exact ones.

- Interesting to compare the best exact technique and the best \(ng\)-route one when applied to branch-and-price, in terms of speed and lower bound quality.
Tuning and a matheuristic

- Decisions are parametrized (numerically and not)
- Use automatic tuning with a tool such as the irace\textsuperscript{9} package to obtain the best configuration on a set of test instances
- Branch-and-price can be used in a matheuristic\textsuperscript{10}
- In particular we can use a Restricted master heuristic
- The 0-1 restricted master problem, when solved exactly can provide a heuristic solution for the original VRP
- Additionally, any metaheuristic can be applied to obtain:
  - new solutions
  - new columns to use in the branch-and-price procedure

\textsuperscript{9}López-Ibáñez et al. 2011.
\textsuperscript{10}“Heuristics algorithms made by the interoperation of metaheuristics and mathematical programming techniques” - Boschetti et al. 2009
Thanks for your attention.


