Column Generation Based Algorithms for a VRP with Time Windows and Variable Departure Times

S. Michelini, Y. Arda, H. Küçükaydın

QuantOM — HEC Liège— Université de Liège

July 5th, 2016

Table of Contents

1 Algorithms for the ESPPRC

2 Branch-and-Price

3 Future research

The problem

- Variant of the VRP with time windows
  - Objective is the total route duration.
  - Route start time is a decision variable.
  - Maximum duration defined for each route.
- Solution method: Branch-and-Price
  - The pricing problem is the Elementary Shortest Path Problem with Resource Constraints (ESPPRC).
- Adapt[1] Feillet's label extension algorithm[2] for the ESPPRC:
  - Label structure
  - Label extension rules
  - Label domination rules

[1] Arda, Crama, and Kucukaydın 2014.
[2] Feillet et al. 2004.

- Each label L_i represents a partial path ending at node i.
- L_i keeps track of the consumption of resources along the partial path.

Label domination

$$L_i = \begin{pmatrix} C_i \\ \mathbf{R}_i \\ \mathbf{E}_i \end{pmatrix} \leq \cancel{\begin{pmatrix} C_i' \\ \mathbf{R}_i' \\ \mathbf{E}_i' \end{pmatrix}} = L_i'$$

- Binary resources for every node ensure elementarity.
- This makes label domination more difficult.
- The following algorithms relax the state space with regard to the elementarity resources.

- Maintain a set Θ of *critical* nodes on which elementarity is enforced.

Algorithm 1 DSSR

```
1: Initialize Θ
2: repeat
3:     P = LabelExtension(Θ)
4:     Φ = MultipleVisits(P)
5:     Θ = Θ ∪ Φ
6: until P is elementary
```

- We manipulate the state space in a *global* way.

[3] Righini and Salani 2008.

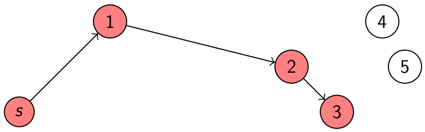
ng-route Relaxation[4]

- Define neighbourhoods $N_i, \ \forall i$.

[4] Baldacci et al. 2010.

ng-route Relaxation[4]

- Define neighbourhoods $N_i, \ \forall i$.
- E_i = set of unreachable nodes for path ending at i.

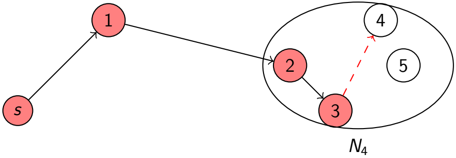


- $\mathbf{E}_3 = \{s, 1, 2, 3\}$

[4] Baldacci et al. 2010.

ng-route Relaxation[4]

- Define neighbourhoods $N_i, \ \forall i$.
- E_i = set of unreachable nodes for path ending at i.
- When extending to j: $\forall k \in E_i$, insert k in E_j only if $k \in N_j$.

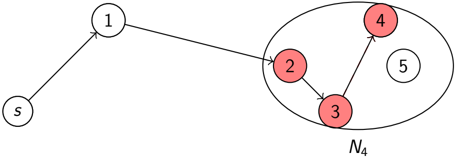


- $\mathbf{E}_3 = \{s, 1, 2, 3\}$
- $j = 4, \ \ N_4 = \{2, 3, 4, 5\}$

[4] Baldacci et al. 2010.

ng-route Relaxation[4]

- Define neighbourhoods $N_i,\ \forall i$.
- E_i = set of unreachable nodes for path ending at i.
- When extending to j: $\forall k \in E_i$, insert k in E_j only if $k \in N_j$.

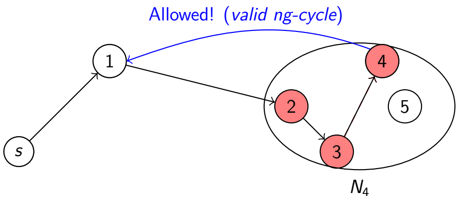


- $\mathbf{E}_3 = \{s, 1, 2, 3\}$
- $j = 4,\ \ N_4 = \{2, 3, 4, 5\}$
- $\mathbf{E}_4 = (\mathbf{E}_3 \cap N_4) \cup \{4\} = \{2, 3, 4\}$

[4] Baldacci et al. 2010.

ng-route Relaxation[4]

- Define neighbourhoods $N_i,\ \forall i$.
- $E_i =$ set of unreachable nodes for path ending at i.
- When extending to j: $\forall k \in E_i$, insert k in E_j only if $k \in N_j$.

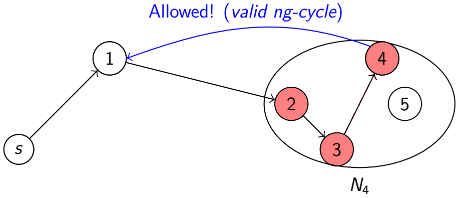


- $\mathbf{E}_3 = \{s, 1, 2, 3\}$
- $j = 4,\ \ N_4 = \{2, 3, 4, 5\}$
- $\mathbf{E}_4 = (\mathbf{E}_3 \cap N_4) \cup \{4\} = \{2, 3, 4\}$

[4] Baldacci et al. 2010.

ng-route Relaxation[4]

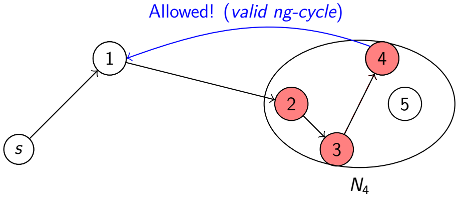
- Define neighbourhoods $N_i, \ \forall i$.
- E_i = set of unreachable nodes for path ending at i.
- When extending to j: $\forall k \in E_i$, insert k in E_j only if $k \in N_j$.
- Resulting path might be **not** elementary.



- $\mathbf{E}_3 = \{s, 1, 2, 3\}$
- $j = 4, \ N_4 = \{2, 3, 4, 5\}$
- $\mathbf{E}_4 = (\mathbf{E}_3 \cap N_4) \cup \{4\} = \{2, 3, 4\}$

[4] Baldacci et al. 2010.

- Define neighbourhoods $N_i, \ \forall i$.
- $E_i =$ set of unreachable nodes for path ending at i.
- When extending to j: $\forall k \in E_i$, insert k in E_j only if $k \in N_j$.
- Resulting path might be **not** elementary.
- We manipulate the state space in a *local* way.



- $\mathbf{E}_3 = \{s, 1, 2, 3\}$
- $j = 4, \ \ N_4 = \{2, 3, 4, 5\}$
- $\mathbf{E}_4 = (\mathbf{E}_3 \cap N_4) \cup \{4\} = \{2, 3, 4\}$

[4] Baldacci et al. 2010.

- We can combine both algorithms in several ways.
- The straightforward combination uses both neighbourhoods and the set of critical nodes.

Algorithm 2 ng-DSSR-global

1: Initialize $\Theta,\ N_i\ \forall i$
2: **repeat**
3: $\quad P = \text{LabelExtension}(\Theta,\ N_i)$
4: $\quad \Phi = \text{MultipleVisits}(P, \{N_i\}_i)$ {Only takes nodes in invalid ng-cycles}
5: $\quad \Theta = \Theta \cup \Phi$
6: **until** P is ng-route

- Let us define *applied* neighbourhoods $\hat{N}_i \subseteq N_i \; \forall \; i$ to use throughout label extension.[5]
- When best path has invalid cycle C, update applied neighbourhoods of nodes in C.

Algorithm 3 ng-DSSR-local

Initialize $N_i, \; \hat{N}_i, \; \forall i$
repeat
    $P = \text{LabelExtension}(\{\hat{N}_i\}_i)$
    $C = \text{InvalidCycle}(P, \{N_i\}_i)$
    $\text{Update}(\{\hat{N}_i\}_i, C)$
until P is ng-route

[5] Dayarian et al. 2015b.

- We can derive a version of DSSR with a local approach[6].
- Let us define local critical sets $\hat{\Theta}_i, \ \forall \ i$.
- If best path has cycle C, update critical sets of nodes in C with the endpoint node of C.

Algorithm 4 DSSR-local

1: Initialize $\hat{\Theta}_i, \ \forall i$
2: **repeat**
3: $\quad P = \text{LabelExtension}(\{\hat{\Theta}_i\}_i)$
4: $\quad C = \text{Cycle}(P)$
5: $\quad \text{Update}(\{\hat{\Theta}_i\}_i, C)$
6: **until** P is elementary

[6] Martinelli, Pecin, and Poggi 2014.

- ng-DSSR-local (global) returns ng-routes.
- We can follow-up with DSSR-local (global) to obtain elementary routes[7].

Algorithm 5 ng-DSSR-local, corrected

1: $P =$ ng-DSSR-local()
2: **if** P is not elementary **then**
3: P = DSSR-local(P)
4: **end if**

[7] Dayarian et al. 2015a.

- *ng*-route relax.
- *ng*-DSSR-global
- *ng*-DSSR-local

} *ng*-route algorithms

- DSSR
- Local DSSR
- *ng*-DSSR-global, corrected
- *ng*-DSSR-local, corrected

} Exact algorithms

Parameters:

Parameters	DSSR_G	DSSR_L	ng-DSSR_G+	ng-DSSR_L+	ng-DSSR_G	ng-DSSR_L	ng-route
NUM_COL	✓	✓	✓	✓	✓	✓	✓
DSSR_INIT_STRATEGY	✓	✓	✓		✓		
DSSR_INIT_AMOUNT	✓	✓	✓		✓		
DSSR_INSERT_PATH_STRATEGY	✓	✓	✓	✓	✓	✓	
DSSR_INSERT_PATH_AMOUNT	✓	✓	✓	✓	✓	✓	
DSSR_INSERT_NODE_STRATEGY	✓		✓		✓		
DSSR_INSERT_NODE_AMOUNT	✓		✓		✓		
NG_SET_TYPE			✓	✓	✓	✓	✓
NG_SET_SIZE			✓	✓	✓	✓	✓
NG_MIX_ALPHA			✓	✓	✓	✓	✓

- After solving the linear relaxation at the root node, solve the integer program with the available columns to obtain an upper bound.
- Branch on the most fractional arc.
- Tree navigation strategies (parametrized for each algorithm):
  - Depth-first search
  - Breadth-first search
  - Best-first search

- Option of using heuristics for the ESPPRC:
  1. Fix the start time of the route, proceed with standard algorithm;
  2. First heuristic domination: compare only $RC = \sum_i \eta_i + T$;
  3. Second heuristic domination: compare separately $\sum_i \eta_i$ and T.

Heuristic configuration	Won instances
None	2
Only 1st heuristic	1
Only 2nd heuristic	5
Both heuristics	2

Only second heuristic

	cpu time	**nodes solved**	**root exact iterations**	**root 2nd heur iter**	**non root exact iterations**	**non root 2nd heur iter**
n=25	cpu time	nodes solved	root exact iterations	root 2nd heur iter	non root exact iterations	non root 2nd heur iter
C103	3918.67	17723	3	18	18372	830
C108	21.89	271	3	11	333	92
C109	144.70	767	3	13	913	182
RC104	3.41	255	2	7	253	44
RC108	4.45	411	1	6	416	34
n=50	cpu time	nodes solved	root exact iterations	root 2nd heur iter	non root exact iterations	non root 2nd heur iter
C107	261.56	771	3	20	913	206
RC102	9.06	391	3	5	418	60
RC105	2554.83	134231	2	5	134287	244
R106	11.14	133	3	12	151	63
R112	4868.92	46065	2	16	46315	949

Heuristic on all nodes vs. only on the root node

	cpu time - ALL NODES	cpu time - ONLY ROOT	nodes solved - ALL NODES	nodes solved - ONLY ROOT
n=25	cpu time	cpu time	nodes solved	nodes solved
C103	3918.67	4112.50	17723	24041
C108	21.89	21.16	271	251
C109	144.70	135.42	767	775
RC104	3.41	2.81	255	269
RC108	4.45	3.28	411	395
n=50	cpu time	cpu time	nodes solved	nodes solved
C107	261.56	268.38	771	799
RC102	9.06	8.19	391	407
RC105	2554.83	2397.34	134231	140755
R106	11.14	8.52	133	99
R112	4868.92	7118.23	46065	68719

Future research: Objectives

- Running the experiments:
  1. Tune the parameters of each ESPPRC algorithm with the `irace` package[8].
     - This will yield a (set of) parameter configurations(s) that is of good quality in a *statistically significant* way.
  2. Compare the performance of the algorithms on the instances used for testing (Gehring & Homberger)[9].
     - We obtain an exact BP algorithm and an *ng*-route-based one.
  3. Run the Branch-and-Price algorithm on the benchmark instances (Solomon)[10].

[8] López-Ibáñez et al. 2011.

[9] Gehring and Homberger 2001.

[10] Solomon 1987.

Parameters: irace tuning

Parameters	TREE_NAV	NUM_COL		DSSR_INIT_STRATEGY		DSSR_INIT_AMOUNT		DSSR_INSERT_PATH_STRATEGY		DSSR_INSERT_PATH_AMOUNT		DSSR_INSERT_NODE_STRATEGY		DSSR_INSERT_NODE_AMOUNT		NG_SET_TYPE		NG_SET_SIZE		NG_MIX_ALPHA	
Algorithms	-	ROOT	NON_RT	ROOT	NON_RT	ROOT	NON_RT	ROOT	NON_RT	ROOT	NON_RT	ROOT	NON_RT	ROOT	NON_RT	ROOT	NON_RT	ROOT	NON_RT	ROOT	NON_RT
DSSR_G	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓						
DSSR_L	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓										
ng-DSSR_G+	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
ng-DSSR_L+	✓	✓	✓					✓	✓	✓	✓					✓	✓	✓	✓	✓	✓
ng-DSSR_G	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
ng-DSSR_L	✓	✓	✓					✓	✓	✓	✓					✓	✓	✓	✓	✓	✓
ng-route	✓	✓	✓													✓	✓	✓	✓	✓	✓

Future research: Objectives

- For the next part of the project:
  1. Study the multi-trip variant of the problem.
  2. Adapt the Branch-and-Price algorithm to the multi-trip variant.
  3. Develop a matheuristic.

Thanks for your attention.