Efficient ng-Route Pricing

Diego Pecin, Rafael Martinelli, and Marcus Poggi
Departamento de Informática, PUC-Rio, Brazil
[dpecin, rmartinelli, poggi]@inf.puc-rio.br

June 12, 2012

- Column generation is present in the current most efficient approaches to routing problems
- Set partitioning formulations model routing problems by considering all possible routes and selecting a subset of them that visits all customers
- This formulation often produces tight linear relaxation lower bounds and requires column generation for its pricing step
- Recently, Baldacci, Mingozzi and Roberti (2011) proposed the ng-routes as compromise between elementary and non-elementary routes, known as q-routes.

- The ng-routes are non-elementary routes with the restriction that following a customer it is not allowed to visit one that was visited before, if it belongs to a dynamically computed set associated with this first customer
- This dynamic set is obtained from ng-sets of given size, associated to each customer, which is usually composed by the closest ones.

- Let $G = (V, E)$ be an undirected connected graph with vertex set $V = \{v_0, v_1, ..., v_k\}$ and edge set $E = \{\{v_i, v_j\} | v_i, v_j \in V\}$
- There is a special vertex $v_0 \in V$ called the depot
- There exists a demand function $q : V \to \mathbb{Z}$ associated with all vertices, in which the depot has demand $q_0 = 0$
- These demands are to be serviced by a set of K identical vehicles with capacity Q, located at the depot

- There exists a traversal cost function $c : E \to \mathbb{Z}$ associated with each edge.
- Let $\mathcal{R}$ be a set of all possible closed routes starting and ending at the depot. The objective of the VRP is to select a subset of K routes from $\mathcal{R}$ which:
  - Minimizes the total traversal cost
  - The demand from every vertex is serviced by a single vehicle
  - The total demand serviced by each route does not exceed the vehicle capacity Q

The Set Partitioning Approach

- The number of possible routes is exponentially large
- Dantzig-Wolfe decomposition of the undirected formulation with an exponential number of constraints
- This decomposition does not enforce the routes to be elementary

Mathematical Notation

- Ω – Set containing all possible routes
- λ_r – Binary variable, 1 if route r is used
- a_r^v – The number of times vertex v is serviced by route r
- b_r^e – The number of times edge e is traversed by route r

$$\text{Minimize} \sum_{r \in \Omega} c_r \lambda_r$$

subject to

$$\sum_{r \in \Omega} \lambda_r = K$$

$$\sum_{r \in \Omega} a_r^v \lambda_r = 1 \qquad \forall v \in V \backslash \{0\}$$

$$\lambda_r \in [0, 1] \quad \forall r \in \Omega$$

- Let γ and β_v be the dual variables associated with the set partitioning constraints respectively:

$$\bar{c}_r = c_r - \gamma - \sum_{v \in V \setminus \{0\}} a_r^v \beta_v$$

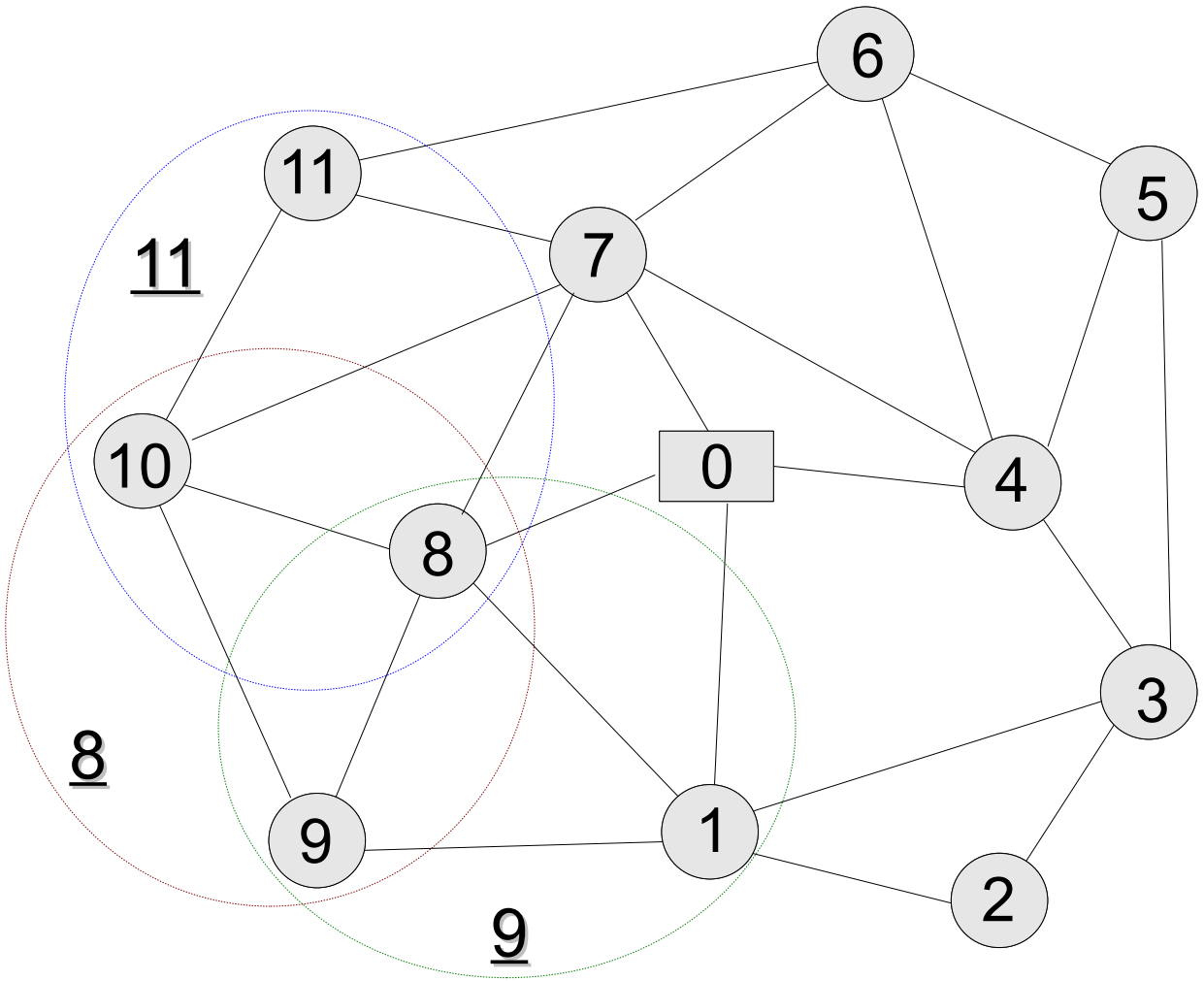
- Given an edge $e = (v_i, v_j) \in E$, the reduced cost $\bar{c}_e$ of this edge can be written as follows:

$$\bar{c}_e = \begin{cases} c_e - \left(\frac{\beta_{v_i} + \beta_{v_j}}{2}\right) & \text{if } e \notin \delta(v_0) \\ c_e - \left(\frac{\beta_{v_i} + \gamma}{2}\right) & \text{if } e \in \delta(v_0) \end{cases}$$

- Since an optimal solution to the VRP does not include routes that visit a vertex more than once, ideally we want to price elementary routes
- This corresponds to solve the Elementary Shortest Path Problem with Resource Constraints (ESPPRC) as a pricing subproblem
- An alternative to deal with this complexity is to relax the elementarily constraint of the path, that means solving the Shortest Path Problem with Resource Constraints (SPPRC), also known as the *q-route problem*
- The SPPRC can be tackled using a pseudo-polynomial dynamic programming algorithm, as described in the seminal work of Christofides et al

- Aiming to have a better compromise between pricing efficiency and lower bounds, Baldacci, Mingozzi and Roberti proposed the ng-route relaxation
- This relaxation defines for each vertex $v_i \in V \backslash \{0\}$ a subset of vertex $N_i \subseteq V \backslash \{0\}$ which have a relationship with vertex v_i
- A possible representation for this relationship can be a neighborhood relationship, i.e., N_i contains the nearest vertex of v_i

ng-Sets



- Given a path $P = (v_0, \ldots, v_i, \ldots, v_p)$, let $V(P)$ be the set of vertices visited by P. A function $\Pi(P)$ of prohibited extensions for the path P can be defined as

$$\Pi(P) = \left\{ v_i \in V(P) : v_i \in \bigcap_{s=i+1}^{p} N_s, i = 1, \ldots, p-1 \right\} \cup \{v_p\}$$

Example 1

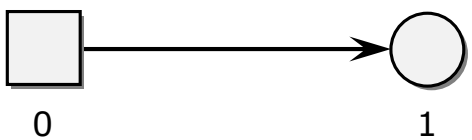
$$N_1 = \{1, 2\},\ N_2 = \{2, 1\},\ N_3 = \{3, 1\}$$

0

$$\pi_0 = \{\}$$

Example 1

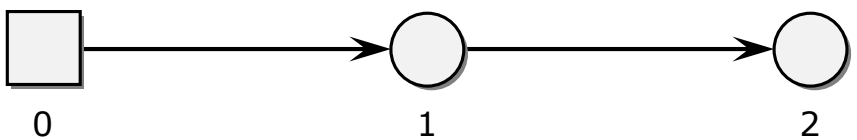
$$N_1 = \{1, 2\},\ N_2 = \{2, 1\},\ N_3 = \{3, 1\}$$



$$\pi_0 = \{\}$$
$$\pi_1 = \pi_0 \cap N_1 \cup \{1\}$$
$$\pi_1 = \{1\}$$

Example 1

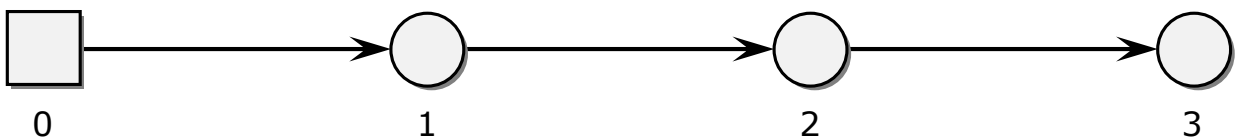
$$N_1 = \{1, 2\},\ N_2 = \{2, 1\},\ N_3 = \{3, 1\}$$



$\pi_1 = \{1\}$
$\pi_2 = \pi_1 \cap N_2 \cup \{2\}$
$\pi_2 = \{1, 2\}$

Example 1

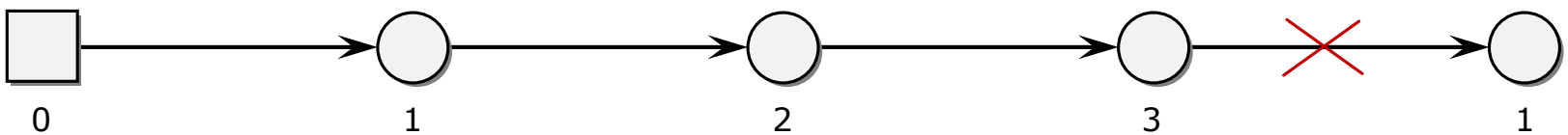
$$N_1 = \{1, 2\},\ N_2 = \{2, 1\},\ N_3 = \{3, 1\}$$



$\pi_2 = \{1, 2\}$
$\pi_3 = \pi_2 \cap N_3 \cup \{3\}$
$\pi_3 = \{1, 3\}$

Example 1

$$N_1 = \{1, 2\},\ N_2 = \{2, 1\},\ N_3 = \{3, 1\}$$



$\pi_3 = \{1, 3\}$

The extension is **not** allowed!

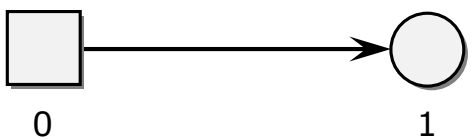
Example 2

$$N_1 = \{1, 2\},\ N_2 = \{2, 1\},\ N_3 = \{3, 2\}$$

$$\pi_0 = \{\}$$

Example 2

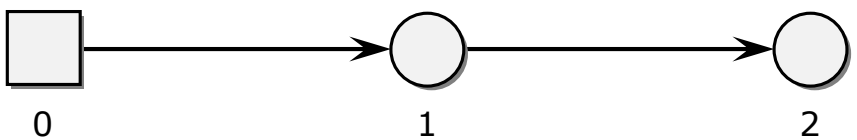
$$N_1 = \{1, 2\},\ N_2 = \{2, 1\},\ N_3 = \{3, 2\}$$



$$\pi_0 = \{\}$$
$$\pi_1 = \pi_0 \cap N_1 \cup \{1\}$$
$$\pi_1 = \{1\}$$

Example 2

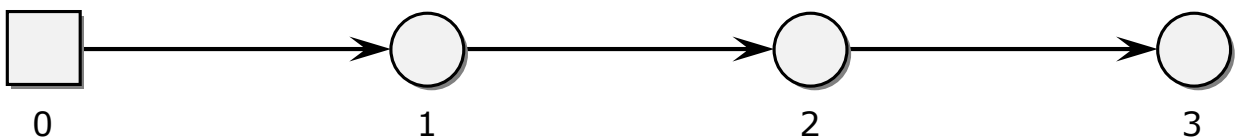
$$N_1 = \{1, 2\},\ N_2 = \{2, 1\},\ N_3 = \{3, 2\}$$



$\pi_1 = \{1\}$
$\pi_2 = \pi_1 \cap N_2 \cup \{2\}$
$\pi_2 = \{1, 2\}$

Example 2

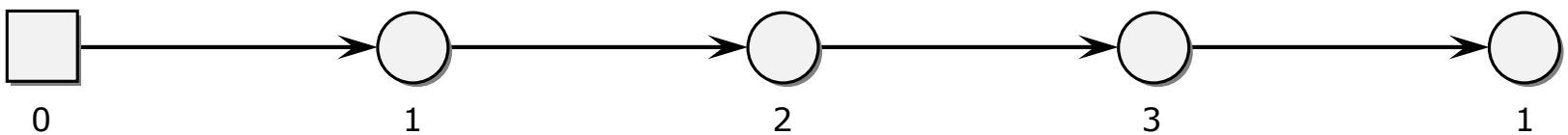
$$N_1 = \{1, 2\},\ N_2 = \{2, 1\},\ N_3 = \{3, 2\}$$



$\pi_2 = \{1, 2\}$
$\pi_3 = \pi_2 \cap N_3 \cup \{3\}$
$\pi_3 = \{2, 3\}$

Example 2

$$N_1 = \{1, 2\},\ N_2 = \{2, 1\},\ N_3 = \{3, 2\}$$



$\pi_3 = \{2, 3\}$

The extension is allowed.

- This pricing is solved exactly using a forward dynamic programming algorithm
- The algorithm stores every possible partial ng-route for each capacity c and end vertex v
- The complexity is still pseudo-polynomial when the size of NGs is bounded by a constant factor
- But it still needs some speed up techniques:
  - Simple heuristic to find ng-routes with negative reduced cost
  - Dominance rules (pricing with elementary routes)
  - Decremental State Space Relaxation (DSSR) (Righini and Salani, 2008)
  - Completion bounds

Heuristic Pricing

- Simple but effective heuristic very similar to our basic dynamic programming algorithm to compute ng-routes
- During the dynamic programming algorithm, we store just the best possible partial ng-route for each capacity c and end vertex v
- The resulting complexity of this algorithm is $\mathcal{O}(n^2 Q)$

Dominance Rule

- Given two paths P_1 and P_2, we say that P_1 dominates P_2 if the following three conditions hold: (i) $q(P_1) \leq q(P_2)$, (ii) $\bar{c}(P_1) \leq \bar{c}(P_2)$ and (iii) $\Pi(P_1) \subseteq \Pi(P_2)$
- It is easy to see that this can be done because any possible extension from P_2, can be done from P_1 with a lower reduced cost

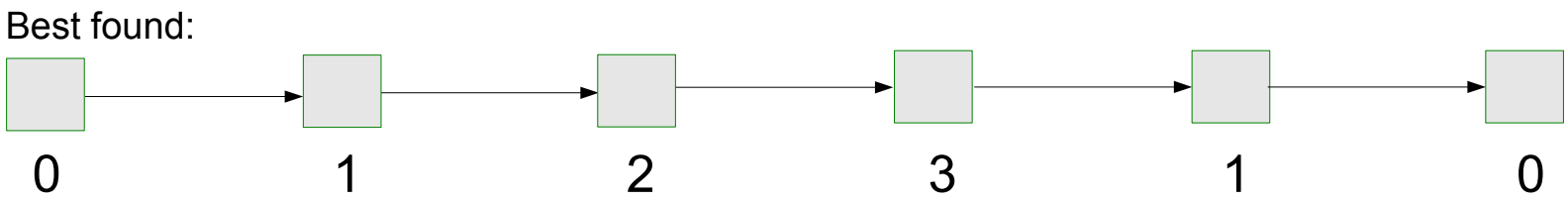
Decremental State-Space Relaxation (DSSR)

- The adapted DSSR algorithm is iterative and works by relaxing the state space of the original neighborhood subsets N_i
- Initially, we define these subsets as empty
- At each iteration, it identifies which repeated vertices violates the original subsets N_i on the best ng-routes
- These vertives are then inserted into the related N_i subsets

ng-Routes with DSSR: Exemple

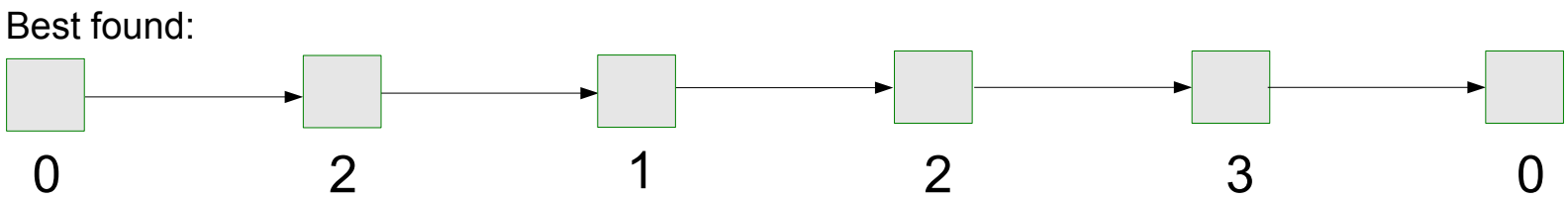
$$N_1 = \{1, 2\},\ N_2 = \{2, 1\},\ N_3 = \{3, 1\}$$

First Iteration: $N_1^0 = \{\},\ N_2^0 = \{\},\ N_3^0 = \{\}$



It's **not** a valid ng-route!

Second Iteration: $N_1^1 = \{1\},\ N_2^1 = \{1\},\ N_3^1 = \{1\}$



It's **not** a valid ng-route!

ng-Routes with DSSR: Exemple

$$N_1 = \{1, 2\},\ N_2 = \{2, 1\},\ N_3 = \{3, 1\}$$

Second Iteration: $N_1^1 = \{1\},\ N_2^1 = \{1\},\ N_3^1 = \{1\}$

Best found:

It's **not** a valid ng-route!

Third Iteration: $N_1^2 = \{1, 2\},\ N_2^2 = \{2, 1\},\ N_3^2 = \{1\}$

Best found:

It's a valid ng-route!

Completion Bounds

- We calculate completion bounds at some DSSR iteration k.
- Let $T_k^*(q, w)$ be the value of the best path which starts at depot v_0 and ends at vertex w with a total demand of q, obtained at the end of iteration k. The completion bound is calculated as

$$\widehat{T}(q, w) = \min_{q' \leq q} \{T_k^*(q', w)\}$$

- These bounds can be used in the subsequent iterations in order to avoid extensions on which the generated label would never result in an optimal ng-route
- The extension of a label $\mathcal{L}(P) = (w_p, q(P), \Pi(P), \bar{c}(P))$ to a vertex w, $w \notin \Pi(P)$, can only be done if

$$\bar{c}(P) + \bar{c}_{w_p w} + \widehat{T}(Q - q(P), w) < 0$$

Computational Experiments

- The algorithms were implemented in C++ using Microsoft Visual C++ 2010 Express
- IBM ILOG CPLEX Optimizer 12.3 was used for solving the formulations
- The experiments were conducted on an Intel Core 2 Duo E7400 2.8 GHz with 4GB RAM running Microsoft Windows Vista Business 32-bits

	Best Bounds		QR-2		NG=8		NG=16		NG=32		NG=64	
Instance	LB	UB	Value	Time	Value	Time	Value	Time	Value	Time	Value	Time
A-n63-k10	1314	1314	1267.42	1.44	1286.58	3.29	1286.81	3.00	1286.83	3.54	1286.83	4.08
A-n64-k9	1401	1401	1353.27	1.89	1368.24	3.89	1374.49	3.69	1376.90	5.07	1376.90	9.25
A-n69-k9	1159	1159	1113.24	2.49	1129.97	3.16	1131.33	3.46	1131.34	4.64	1131.34	6.81
A-n80-k10	1763	1763	1712.17	4.28	1729.81	7.23	1731.45	8.28	1731.58	9.29	1731.58	14.40
B-n50-k8	1312	1312	1217.52	0.74	1266.45	1.99	1266.63	1.95	1266.64	2.05	1266.64	2.97
B-n68-k9	1275	1275	1163.87	1.87	1198.20	3.32	1203.78	5.63	1204.00	5.65	1204.00	14.99
E-n101-k14	1067	1067	1045.11	7.23	1047.20	10.00	1048.45	10.79	1050.42	10.83	1050.42	14.71
E-n101-k8	815	815	786.36	17.67	789.37	25.45	790.71	30.01	790.99	36.13	790.99	54.28
E-n51-k5	521	521	512.92	1.80	517.14	3.05	517.14	4.82	517.14	2.83	517.14	4.57
E-n76-k10	830	830	811.39	3.58	811.77	3.96	811.87	4.59	812.48	5.35	812.48	8.52
E-n76-k14	1021	1021	999.58	2.07	1001.85	2.50	1002.77	3.00	1002.77	4.09	1002.77	4.30
E-n76-k7	682	682	663.31	7.17	664.20	12.48	664.82	12.85	665.59	16.27	665.59	25.56
E-n76-k8	735	735	716.71	5.49	717.82	7.67	718.74	9.15	718.78	8.07	718.78	14.52
P-n50-k8	631	631	612.54	0.82	614.64	1.12	615.55	1.34	615.55	1.29	615.55	2.01
P-n70-k10	827	827	808.31	2.32	809.29	2.78	810.61	4.24	810.94	3.00	810.94	4.64
M-n121-k7	1034	1034	1013.00	26.66	1026.37	167.93	1029.21	2609.38	—	—	—	—
M-n151-k12	1003	1015	991.49	57.15	995.73	96.14	996.64	126.56	997.28	172.54	997.43	3946.30
M-n200-k16	1256.4	1278	1240.44	116.10	1250.23	260.57	1250.87	269.28	1251.36	261.28	1252.05	1528.53
M-n200-k17	1256.8	1275	1243.32	123.11	1252.52	237.15	1252.87	207.09	1253.43	273.82	1254.01	765.77
G-n262-k25	5064.0	5530	5361.79	1242.14	5418.51	1369.33	5425.95	1428.45	5429.20	1587.43	5429.46	1953.33

Concluding Remarks

- We show how to implement an efficient algorithm to price ng-routes to a set partitioning formulation for VRP
- This was done by combining the Decremental State Space Relaxation technique with completion bounds within a basic algorithm for pricing ng-routes
- The resulting algorithm was tested extensively on a large set of VRP instances, where some of them are still far from an optimality proof
- As far as we know, up to this date, the best column generations with ng-route pricing could only run for subsets up to 13

Thank you!