Branch-Cut-and-Price for Routing Problems

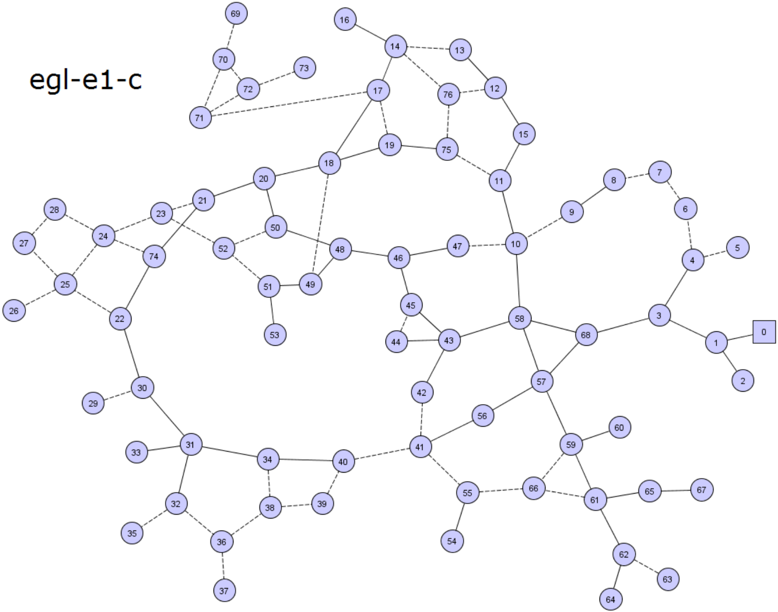
Rafael Martinelli, Diego Pecin, Marcus Poggi
Departamento de Informática, PUC-Rio, Brazil
Eduardo Uchoa, Artur Pessoa
Departamento de Engenharia de Produção, UFF, Brazil

June 13, 2012

Capacitated Arc Routing Problem (CARP)

- Connected undirected graph $G = (V, E)$
- Costs $c : E \rightarrow \mathbb{Z}^+$
- Demands $d : E \rightarrow \mathbb{Z}^+$
- Set I containing k identical vehicles with capacity Q
- Depot vertex labeled 0
- Set $E_R = \{e \in E \mid d_e > 0\}$ of **required** edges

egl-e1-c


69
16
14
13
70
73
72
17
76
12
71
15
19
75
18
20
11
8
7
21
28
23
6
9
50
24
27
52
48
46
47
10
5
4
74
51
49
25
26
45
22
58
3
53
44
43
68
0
1
57
2
30
42
29
56
60
33
31
59
41
34
40
55
66
65
67
61
32
38
39
35
36
54
62
63
37
64

- Golden and Wong, 1981
- Strongly NP-Hard

- Applications:
  - Garbage collection
  - Street sweeping
  - Winter gritting
  - Electric meter reading
  - Airline scheduling

Proved Optimal Solution

- Cutting Planes (Belenguer and Benavent, 2003; Ahr, 2004)
- Branch-and-Cut (Laganá et al., 2007)
- Branch-Cut-and-Price via CVRP (Longo et al., 2006)
- Column Generation (Gómez-Cabrero et al., 2005; Letchford and Oukil, 2009)
- Branch-Cut-and-Price (Martinelli et al., 2011)
- Cut-First Branch-and-Price Second (Bode and Irnich, 2011)
- Cut-and-Price with Elementary Routes (Bartolini et al., 2011)

Hard to solve for more than 30 required edges.

The Set Partitioning Approach

- The number of possible routes is exponentially large
- Dantzig-Wolfe decomposition of flow formulation
- 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^e – The number of times edge e is serviced by route r
- b_r^e – The number of times edge e is deadheaded by route r

Mathematical Formulation

$$
\begin{aligned}
\text{MIN} \quad & \sum_{r \in \Omega} c_r \lambda_r \\
\text{s.t.} \quad & \sum_{r \in \Omega} \lambda_r = k \\
& \sum_{r \in \Omega} a_r^e \lambda_r = 1 \quad \forall e \in E_R \\
& \lambda_r \in \{0, 1\} \quad \forall r \in \Omega
\end{aligned}
$$

Odd Degree Cutset Cuts

$$\sum_{r\in\Omega}\sum_{e\in\delta(S)} b_r^e \lambda_r \geq 1 \quad \forall S \subseteq V\backslash\{0\}, |\delta_R(S)| \text{ odd}$$

Capacity Cuts

$$\sum_{r\in\Omega}\sum_{e\in\delta(S)} b_r^e \lambda_r \geq 2k(S) - |\delta_R(S)| \quad \forall S \subseteq V\backslash\{0\}$$

Column Generation with ng-Routes

$$\begin{aligned}\bar{c}_r &= c_r - \gamma - \sum_{e \in E_R} a_r^e \beta_e - \sum_{S \subseteq V \setminus \{0\}} \sum_{e \in \delta(S)} b_r^e \pi_S \\ &= -\gamma + \sum_{e \in E_R} a_r^e \left(c_e - \beta_e\right) + \sum_{e \in E} b_r^e \left(c_e - \sum_{S \subseteq V \setminus \{0\} : e \in \delta(S)} \pi_S \right)\end{aligned}$$

- This pricing is solved exactly using a forward dynamic programming algorithm
- 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

Branching Rule

- Given z_e, an integer variable which represents the number of times an edge e is deadheaded by *any* vehicle, the branch is done on these variables z_e and then translated to the route variables λ_r
- The algorithm searches for a variable z_e^* with solution closer to 0.5 and then creates two branches

$$\sum_{r \in \Omega} b_r^e \lambda_r \quad \leq \quad \lfloor z_e^* \rfloor$$

$$\sum_{r \in \Omega} b_r^e \lambda_r \quad \geq \quad \lceil z_e^* \rceil$$

$$lb_e \leq \sum_{r \in \Omega} b_r^e \lambda_r \leq ub_e \quad \forall e \in E$$

Pricing

- With the introduction of the bounds constraints, the pricing subproblem does not change, but the reduced cost of a route must consider the dual values associated with these constraints

$$\tilde{c}_r = -\gamma + \sum_{e \in E_R} a_r^e \,(c_e - \beta_e) + \sum_{e \in E} b_r^e \left(c_e - \rho_e - \sum_{S \subseteq V \setminus \{0\} : e \in \delta(S)} \pi_S \right)$$

Strong Branching

- Used in order to obtain better bounds faster
- When a branching variable needs to be chosen, the algorithm selects n candidates to branch ($n = 5$)
- It then runs the column generation algorithm for both branches of candidate c
- The candidate with largest $min\{left_c, right_c\}$ is chosen. In the case of a tie, the candidate with largest $max\{left_c, right_c\}$ is chosen
- If the algorithm finds a candidate with at least one branch infeasible, this one is chosen immediately

Odd Degree Cutset Cuts Separation

- Use the exact algorithm proposed by Padberg and Rao in 1982
- The algorithm builds a Gomory-Hu tree (Gomory and Hu, 1961)
- It can be done in polynomial time running $|V| - 1$ times any max flow algorithm

Capacity Cuts Separation

- Use the exact algorithm proposed by Martinelli et al. in 2011
- Inspired on the exact separation of Chvátal-Gomory Cuts done by Fischetti and Lodi in 2007
- It uses a mixed-integer formulation to find a violated cut

Computational Experiments

- The algorithms were implemented in C++ using Microsoft Visual C++ 2010 Express
- IBM ILOG CPLEX Optimizer 12.4 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

- The cut separation is done before starting the Branch-Cut-and-Price algorithm
- Then it starts the Branch-Cut-and-Price with these cuts
- The separation of the Odd Degree Cutset Cuts is done before the execution of the exact pricing
- The exact separation of the Capacity Cuts is done only when an integer solution is found

Instance Name	Best Known		ng-24				
	LB	UB	Value	Nodes	Routes	Cuts	Time
e1-a	3548	3548	**3548**	1	2883	55	21
e1-b	4498	4498	4496	91	6780	91	470
e1-c	5595	5595	5556	25	2084	70	145
e2-a	5018	5018	**5018**	7	6251	88	1551
e2-b	6305	6317	6301	55	6047	119	572
e2-c	8335	8335	8308	249	4295	104	818
e3-a	5898	5898	**5898**	5	9219	80	617
e3-b	7711	7775	**7722**	515	14590	229	8652
e3-c	10244	10292	10214	339	5676	117	1771
e4-a	6408	6444	6393	19	12118	169	*6h*
e4-b	8935	8961	8908	1663	14871	214	*6h*
e4-c	11493	11550	**11495**	857	5869	230	5775
s1-a	5018	5018	**5018**	3	6948	143	466
s1-b	6388	6388	**6388**	29	7502	154	638
s1-c	8518	8518	8511	81	4489	184	476
s2-a	9825	9884	9818	107	29982	555	*6h*
s2-b	13017	13100	12994	591	14013	611	*6h*
s2-c	16425	16425	16384	1187	9819	363	*6h*
s3-a	10145	10220	**10155**	19	14214	211	*6h*
s3-b	13648	13682	13638	435	15353	593	*6h*
s3-c	17188	17188	17139	983	9523	498	*6h*
s4-a	12143	12268	**12145**	28	14066	252	*6h*
s4-b	16098	16283	**16102**	265	12522	596	*6h*
s4-c	20430	20481	20411	641	7085	373	*6h*

Generalized Vehicle Routing Problem (GVRP)

- Let $G = (V, E)$ be an undirected connected graph with vertex set V and edge set E
- There is a special vertex $v_0 \in V$ called the depot
- The vertices are partitioned into disjoint sets called clusters $C = \{C_0, C_1, ..., C_t\}$
- The cluster $C_0 = \{v_0\}$ contains only the depot
- Given the cluster index set $M = \{0, 1, ..., k\}$, let $\alpha(v) \in M$ be defined as the index of the cluster which contains v
- There exists a demand function $q : M \to \mathbb{Z}$ associated with all clusters, 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

- The GVRP is a generalization of the Capacitated Vehicle Routing Problem (CVRP) and the Generalized Traveling Salesman Problem (GTSP)
- When all the clusters contain only one vertex, it is the CVRP
- When there is only one vehicle, it is the GTSP
- It is clear that, when both conditions are true, it is the Traveling Salesman Problem (TSP)

- This problem is strongly $\mathcal{NP}$-Hard
- As far as we know, the first published work to deal with this problem is the one by Ghiani and Improta
- Recently, the work of Bektaş, Erdoğan and Røpke proposes four formulations for the GVRP and devises a branch-and-cut algorithm using one of these formulations

Mathematical Notation

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

Mathematical Formulation

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

subject to

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

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

$$\lambda_r \in \{0, 1\} \quad \forall r \in \Omega$$

Capacity Cuts

$$\sum_{r\in\Omega}\sum_{e\in\delta(S)} b_r^e\lambda_r \geq 2k(S) \quad \forall S \subseteq C\backslash\{C_0\}$$

Strengthened Comb Cuts

$$\sum_{r\in\Omega}\left(\sum_{e\in\delta(H)} b_r^e\lambda_r + \sum_{j=1}^{t}\sum_{e\in\delta(T_j)} b_r^e\lambda_r\right) \geq S(H, T_1, \ldots, T_t) + 1$$

$$S(H, T_1, \ldots, T_t) := \sum_{j=1}^{t}\left(\tilde{k}(T_j \cap H) + \tilde{k}(T_j\backslash H) + \tilde{k}(T_j)\right)$$

- The cut separation of these constraints is done using the CVRPSep (Lysgaard, 2003)

Column Generation with ng-Routes

- Let γ, β_m and π_S be the dual variables associated with the set partitioning constraints and the capacitated cuts, respectively:

$$\bar{c}_r = c_r - \gamma - \sum_{m \in M \setminus \{0\}} a_r^m \beta_m - \sum_{S \subseteq C \setminus \{C_0\}} \sum_{e \in \delta(S)} b_r^e \pi_S$$

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

$$\bar{c}_e = \begin{cases} c_e - \left(\frac{\beta_{\alpha(v_i)} + \beta_{\alpha(v_j)}}{2}\right) - \sum_{S \subseteq C \setminus \{C_0\} : e \in \delta(S)} \pi_S & \text{if } e \notin \delta(C_0) \\ c_e - \left(\frac{\beta_{\alpha(v_i)} + \gamma}{2}\right) - \sum_{S \subseteq C \setminus \{C_0\} : e \in \delta(S)} \pi_S & \text{if } e \in \delta(C_0) \end{cases}$$

- This pricing is solved exactly using a forward dynamic programming algorithm
- 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

Computational Experiments

- The algorithms were implemented in C++ using Microsoft Visual C++ 2010 Express
- IBM ILOG CPLEX Optimizer 12.4 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
- The instance sets used were recently created by Bektaş et al.
- These instance sets are derived from the CVRP instance sets A, B, P, M and G
- The transformation is made using a method similar to the one by Fischetti et al. which transforms TSP instances into GTSP instances

Instance Name	Best Known		ng-8		ng-16		ng-24		ng-32	
	LB	UB	Value	Time	Value	Time	Value	Time	Value	Time
A-n63-k9-C32-V5	900.3	912	906.8	2.9	907.0	1.6	907.0	2.4	907.0	3.0
A-n63-k9-C21-V3	625.6	642	629.7	1.3	636.3	1.4	636.3	1.0	636.3	1.0
A-n80-k10-C27-V4	679.4	710	706.5	2.1	708.8	2.4	708.8	2.7	708.8	2.4
A-n80-k10-C40-V5	957.4	997	982.4	4.3	982.7	4.8	983.6	4.1	983.0	6.2
P-n50-k8-C25-V4	378.4	392	385.7	0.9	385.8	0.8	385.8	0.8	385.8	1.0
P-n55-k15-C28-V8	545.3	555	555.0	0.4	555.0	0.4	555.0	0.4	555.0	0.4
P-n60-k10-C30-V5	433.0	443	435.4	1.8	435.5	2.4	435.5	2.2	435.5	2.5
P-n60-k15-C20-V5	379.3	382	380.3	0.6	381.5	0.7	381.5	0.7	381.5	0.7
P-n60-k15-C30-V8	553.9	565	564.7	0.6	564.8	0.6	564.8	0.6	564.8	0.6
M-n121-k7-C61-V4	707.7	719	710.4	117.4	710.7	222.4	710.8	1200.1	710.9	1588.3
M-n151-k12-C51-V4	465.6	483	482.3	37.9	483.0	39.1	483.0	64.7	483.0	56.8
M-n151-k12-C76-V6	629.9	659	649.5	52.2	650.0	62.9	650.0	55	650.0	52.7
M-n200-k16-C100-V8	744.9	791	777.8	89.1	778.4	111.3	778.5	115.2	778.8	129.83
M-n200-k16-C67-V6	563.1	605	592.5	79.1	593.6	58.6	594.1	59.1	594.1	68.4
G-n262-k25-C131-V12	2863.5	3249	3176.1	484.12	3,181.2	502.7	3185.2	478.11	3188.1	549.1
G-n262-k25-C88-V9	2102.4	2476	2420.7	271.8	2,425.9	253.1	2426.5	282.1	2427.0	356.2

Instance Name	Best Known		ng-8		ng-16		ng-24		ng-32	
	LB	UB	Value	Time	Value	Time	Value	Time	Value	Time
M-n101-k10	820	820	**820**	10.9	**820**	16.1	**820**	49.7	**820**	60.9
M-n121-k7	1034	1034	1032.5	140.9	1032.6	6275.1	-	-	-	-
M-n151-k12	1003	1015	1000.4	114.0	1001.4	118.3	1001.9	121.2	1002.2	168.9
M-n200-k16	1256.4	1278	1252.5	212.7	1252.8	246.1	1253.2	253.8	1253.3	314.1
M-n200-k17	1256.8	1275	1255.0	231.5	1255.3	226.4	1255.7	292.5	1255.6	320.4
G-n262-k25	5064	5530	**5443.4**	1402.8	**5450.6**	1454.2	**5452.4**	1487.9	**5452.6**	1493.4

Thank you!