MINIMUM WEIGHT ROOTED ARBORESECENCES

Marcus Ritt (joint work with Jordi Pereira)

Outline

1. Overview
2. Problem reductions
3. Mathematical formulation
4. Finding arborescences that span subsets
5. Putting it all together
6. Some results

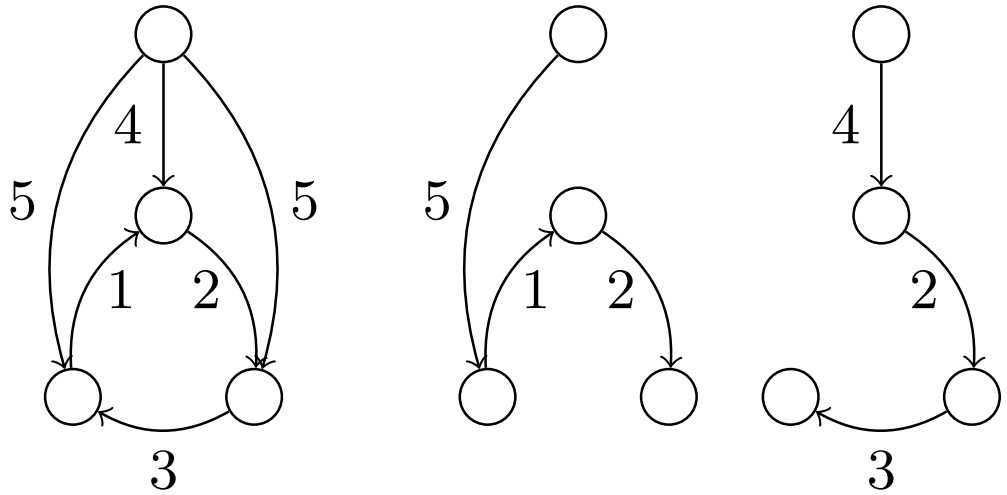
- A directed graph $G = (V, A)$ and arcs weights w_a
- A root $r \in V$

Find an arborescence $G^* = (V^*, A^*)$ rooted in r of minimum total weight.

1. Non-negative case ($w_a \geq 0$): trivial solution $G^* = (\{r\}, \emptyset)$.
2. G^* must be spanning ($V^* = V$): polynomially solvable (Chu & Liu 1965, Edmonds 1967, Bock 1971)
3. Tarjan (1977): $O(n^2)$ for dense, $O(m \log n)$ for sparse graphs
4. Gabow et. al (1986), Fischetti & Toth (1993): link paths, speedups.
5. Unrooted case and branchings: can be made to work, too.

Edmonds' algorithm in a nutshell

while there's an unconnected vertex v
    add lightest in-arc (u, v) to A^*
    if that forms a cycle:
        find lightest cycle arc a
        for all arcs into cycle
            subtract diff. of cycle-arc to w_a
        contract cycle
expand cycles, remove redundant arcs



Uncapacitated facility location: open facilities I at cost $f_i \geq 0$, and connect customers J at cost $c_{ij} \geq 0$.

Reduction to MWRA:

1. Introduce a root r.
2. Connect root to facilities via arcs (r, i) of weight f_i.
3. Connect facilities to clients via arcs (i, j) of weight c_{ij}.
4. Add an auxiliary vertex j', for each client $j \in J$.
5. Connect clients to auxiliary vertices by arcs (j, j') of weight $M_j = -f_{\mu_j} - c_{\mu_j, j} - 1$ where $\mu_j = \operatorname{argmin}_{i \in I}\{f_i + c_{ij}\}$.

Minimum covering arborescence: like MWRA, but a subset of required vertices $R \subseteq V$ must be included.

Reduction to MWRA: (same idea as for UFL):

1. Add an auxiliary vertex v' for each required vertex $v \in R$.
2. Connect required to auxiliary vertices by arcs (v, v') of weight $M_v = -d_{rv} - 1$ where d_{rv} is (some) distance from root to v.

Observations:

- For the MCA $w_a \geq 0, a \in A$ makes sense. In this case we have the Steiner arborescence problem.
- If we have prizes, we can push them into the arcs, so prize-collecting variants can also be solved.
- As for Steiner problems: knowing the optimal vertex set is tantamout to knowing the optimal solution (run Edmonds).

Four simple reduction rules

- Rule 1, Incoming root arcs

  All incoming arcs into the root can be deleted.

- Rule 2, Non-reachable vertices

  All vertices that are not reachable from the root r can be deleted.

- Rule 3, Required vertices of indegree one

  If r is a required vertex with a single incoming arc (v, r), then v is required.

- Rule 4, Non-positive arcs

  All vertices reachable from the root r via a path of non-positive arcs can be made required.

Assume Rule 4 can't be applied any more, but we find a path ruv

- from required vertex $r \in R$
- to non-required vertives $u, v \in \overline{R} = V \setminus R$
- of non-positive weight.

Then v can be made required.
Note: arc (r, u) must be positive, arc (u, v) negative.

This motivates:

- Call a simple path that starts at some vertex in R with arcs in $V \times \overline{R}$ an *outpath*.
- Call R *k-closed*, if for every vertex $v \in \overline{R}$ the shortest outpath P to v of length at most k has positive weight (i.e. $w(P) > 0$).

Note: Applying rule 4 yields a 1-closed set.

- Rule 5, Vertices reachable by non-positive paths

  Let R be a k-closed set. Consider all vertices C that can be reached from some vertex in R by a simple path P of non-positive weight and length $k+1$. Then all vertices C are required.

Reduction algorithm

$R = \{r\}$
apply rules 3 and 4
$k = 1$
while $k < n - 1$ **do**
    `// here` R `is` k`-closed`
    apply Rule 5 to find new required vertices C
    if $C = \emptyset$ **then**
        $k := k + 1$
    else
        $R := R \cup C$
        apply rules 3 and 4
        $k = 1$

Problem: checking k-closedness is NP-complete in general, since checking for a negative-length simple path between two vertices is hard when cycles of negative length can exist.
Solution: heuristic, running Bellman-Ford starting at required vertices R, limited to paths of length at most $k+1$. If any vertex in $\overline{R}$ of non-positive distance root is found, verify that there's is a simple path that witnesses this.
Consequence: we may miss some required vertices.

- Rule 6, Vertices with a single neighbor

  A vertex with a single neighbor and an incoming arc of non-negative weight can be deleted.

- Rule 7, Vertices of non-negative contribution

  Let L be a lower bound on the total weight of the arcs reachable from vertex v and let $\mu = \min_{a \in N^-(v)} w_a$ be the weight of the lightest incoming arc into v. If $\mu + L \geq 0$, vertex v can be deleted, if $L \geq 0$ all outgoing arcs from v can be deleted.

For Rule 7: sum of all negative arcs that can be reached from v is a simple lower bound.

MWRA-a:
$$\min \sum_{(i,j)\in A} w_{ij}x_{ij}, \tag{1}$$
$$\text{s.t.} \sum_{i:(i,j)\in A} x_{ij} \leq 1, \qquad \forall j \in V \setminus \{r\}, \tag{2}$$
$$x_{ij} \leq \sum_{i':(i',i)\in A} x_{i',i} \qquad \forall (i,j) \in A, i \neq r. \tag{3}$$
$$x_{ij} \in \{0,1\}, \qquad \forall (i,j) \in A. \tag{4}$$

- Works in acyclic graphs.
- In general: need to guarantee connectivity, by eliminating subcircuits.

- Subtour elimination

$$\sum_{i \in S} \sum_{j \in S} x_{ij} \leq |S| - 1, \qquad \forall S \subset V, |S| \geq 2. \tag{5}$$

- Cutset inequalities

$$\sum_{i \in S} \sum_{j \notin S} x_{ij} \geq \sum_{i' : (i', j') \in A} x_{i'j'}, \qquad \forall S \subset V, r \in S, j' \notin S. \tag{6}$$

When graph is series-parallel, constraints (2), (5), and $x_{ij} \geq 0$, $\forall (i, j) \in A$ describe the convex hull of the r-arborescences (Goemans 1992, 1994).

- Model usually too large: we want to solve instances with $5K$ vertices, and up to $12.5M$ arcs.
- *Note*: the solution has at most $|V|$ arcs, so most arcs are useless.
- Therefore:
  1. Solve a simple set packing relaxation (1), (2), (4).
  2. Prune arc $a = (i, j)$ if $ub \leq lb + \bar{c}_a$, for red. costs $\bar{c}_a = w_a - \pi_j$.
  3. Solve the linear relaxation of MWRA-a by column generation, with column set $\bar{A}$.
     - Column generation is actually a column and row generation!
     - For each new arc, we need to include a new constraint, too.
  4. Prune arcs as above, for red. costs $\bar{c}_a = w_a - \pi_j - \sum_{a' \in \bar{A}} \pi_{a'}$.
  5. Solve the complete model by a branch-and-cut approach.

- Separate special cutsets (3) as well as general cutsets (6).
- For $m \leq 10K$ include special cutsets (3).

For each integral solution:

- Find the strongly connected components, add

$$\sum_{(i,j)\in A'} x_{ij} \leq |A'| - 1. \tag{7}$$

  to eliminate circuits A'.

For each fractional solution: seperate cutset inequalities

- Special cutsets can be separated by inspection.

Separating general cutsets

- Find integrally connected vertices by DFS
- For non-integrally connected vertices v that are required or have outgoing solutions arcs
  - Solve min-cut/max-flow between root r and v with capacities equal to linear relaxation
  - If flow is less than incoming flow in current solution: add a cut

Edmonds algorithm as of Fischetti & Toth

$A_0 := \emptyset$; $h := 0$
// Phase I
while A_0 *is not* r*-connected* **do**
  $h := h + 1$
(1) find a strong component S_h of $G_0 = (V, A_0)$ such that $r \notin S_h$ and $A_0 \cap K_h = \emptyset$ for the r-cutset $K_h = (V \setminus S_h, S_h)$
(2) find an arc $(u_h, v_h) \in K_h$ such that $w_{(u_h,v_h)} \leq w_{(u,v)}$ for $(u, v) \in K_h$
  $A_0 := A_0 \cup \{(u_h, v_h)\}$
  foreach $(u, v) \in K_h$ **do** $w_{(u,v)} := w_{(u,v)} - w_{(u_h,v_h)}$
// Phase II
for $t = h, h-1, \ldots$ **to** 1 **do**
  if $(u_t, v_t) \in A_0$ **then**
    foreach $q < t$ *such that* $v_t \in S_q$ **do**
      $A_0 := A_0 \setminus \{(u_q, v_q)\}$

Adding required vertices

Strong components are *preferred* if they can reach a required vertex.
Idea: try to connect only if required and to required, or if beneficial.

- Line (1) selects a preferred strong component S_h, if it exists, otherwise any strong component.
- Line (2) selects the lightest cut-arc coming from a preferred strong component; if it does not exist, or S_h is not preferred the overall lightest cut-arc is taken.
- If the selected strong component S_h is not preferred, the cut-arc a is only accepted if $w_a \leq 0$; otherwise S_h is excluded from further consideration.

An iterated local search

Input : An initial solution s.
Output : The best solution found during the search s^*.
$s := \text{localsearch}(s)$
repeat
    $s' := \text{perturb}(s)$
    $s' := \text{localsearch}(s')$
    $s := \text{accept}(s, s')$
until *a stopping criterion is satisfied*

Main idea: search in the space of free vertices S, make some subset $s \subseteq S$ required.

- Initial solution: best of running modified Edmonds algorithm with $s = \emptyset$, $s = S$.
- Local search: flip vertices, first improvement.
- Perturbation: flip $k \in [k_{\min}|S|, k_{\max}|S|]$ vertices randomly.
- Acceptance: with probability $\min\{1, e^{-\alpha r}\}$ for some α and relative deviation of new from old solution value $r = (w(s') - w(s))/w(s)$.
- A twist: search only $L\,\%$ of the neighborhood for improving moves, before accepting, then go to local minimum.

1. Apply the problem reductions. If no free vertices remain, apply Edmonds algorithm to find an optimal solution and stop.
2. Compute a lower bound given by the value of a minimum cost branching.
3. Compute an upper bound by solving different minimum cost arborescence problems: one on all required and free vertices; a second only on the required vertices, giving preference to arcs that connect to preferred components; and a third that also uses only the required vertices, but always chooses the cheapest arc.
4. Solve the first relaxation (set packing). Use the reduced costs of the arcs to exclude those that exceed the upper bound.
5. If the number of free vertices k is small, enumerate all possible 2^k subsets of free vertices, and apply Edmonds algorithm to each of them in order to find the optimal solution and stop.

6. Exclude unpromising arcs heuristically such that the graph remains connected.
   - For $n \to \infty$, $\hat{p} = (\ln n + c)/n$ is a *threshold*: probability of single SCC is $e^{-2e^{-c}}$ (Graham & Pike 2008).
   - Allow max. indegree $\bar{\delta} = 2\lceil \ln n \rceil$, take best arcs only.
7. Repeat step 3 to obtain an alternative upper bound on the reduced graph and apply the ILS metaheuristic starting from the best solution to obtain a tighter upper bound.
8. Apply the second relaxation. Use the reduced costs to exclude arcs that cannot improve the upper bound.
9. Check for the optimality of the incumbent and, if the solution is not optimal, apply the branch-and-cut approach.

Problem	Type	n	p_{arc}	p_{neg}	$\lvert X\rvert$ [%]	N
MWRA	DAG	100, 1000, 3000	0.1, 0.3, 0.5	0.25, 0.5, 0.75	-	10
MWRA	DG	100, 1000, 3000	0.05, 0.15, 0.25	0.25, 0.5, 0.75	-	10
MCA (Set 1)	DAG	500, 1000, 5000	0.1, 0.3, 0.5	0.01	1, 10, 20	10
MCA (Set 2)	DAG	500, 1000, 5000	0.1, 0.3, 0.5	0.001	20	10

n: no. of vertices; p_{arc}: arc probability; p_{neg}: negative weight probability; $|X|$: no. required vertices (% of total vertices); N: instances/group.

Results for MWRA on DAGs

				MathHeur		dapcstp	ILS			Exact
n	p_{arc}	p_{neg}	Opt.	Dev.	Time (s)	Time (s)	Dev.	Time (s)	N	Time (s)
100	0.1	0.25	-2214.7	**0.0**	0.2	0.0	**0.0**	0.0	0	0.0
100	0.1	0.50	-4112.8	**0.0**	0.0	0.0	**0.0**	0.0	0	0.0
100	0.1	0.75	-4628.4	**0.0**	0.1	0.0	**0.0**	0.0	4	0.0
100	0.3	0.25	-5868.7	**0.0**	0.1	0.0	**0.0**	0.0	0	0.0
100	0.3	0.50	-7319.9	**0.0**	0.0	0.0	**0.0**	0.0	5	0.0
100	0.3	0.75	-7999.2	**0.0**	0.0	0.0	**0.0**	0.0	10	—
100	0.5	0.25	-7193.3	**0.0**	0.1	0.0	**0.0**	0.0	1	0.0
100	0.5	0.50	-8332.0	**0.0**	0.0	0.0	**0.0**	0.0	6	0.0
100	0.5	0.75	-8886.6	**0.0**	0.0	0.0	**0.0**	0.0	10	—
1000	0.1	0.25	-80901.3	1.9	138.1	0.2	**0.0**	0.5	0	0.0
1000	0.1	0.50	-88675.5	2.9	73.5	0.2	**0.0**	0.2	0	0.0
1000	0.1	0.75	-91211.2	**0.0**	22.7	0.2	**0.0**	0.1	4	0.0
1000	0.3	0.25	-93225.1	**0.0**	11.5	0.4	**0.0**	0.3	0	0.1
1000	0.3	0.50	-96247.7	**0.0**	14.9	0.3	**0.0**	0.2	3	0.1
1000	0.3	0.75	-96949.9	**0.0**	21.1	0.3	**0.0**	0.2	7	0.1
1000	0.5	0.25	-95930.9	**0.0**	7.4	0.5	**0.0**	0.3	1	0.1
1000	0.5	0.50	-97671.5	**0.0**	2.8	0.4	**0.0**	0.3	6	0.1
1000	0.5	0.75	-98445.7	**0.0**	2.3	0.4	**0.0**	0.2	10	—
3000	0.1	0.25	-278155.4	3.4	593.1	2.2	**0.0**	1.3	0	0.4
3000	0.1	0.50	-286660.4	1.6	434.7	2.8	**0.0**	0.9	0	0.4
3000	0.1	0.75	-290695.2	**0.0**	119.0	2.0	**0.0**	0.8	1	0.3
3000	0.3	0.25	-292479.7	**0.0**	91.2	4.8	**0.0**	1.7	0	0.6
3000	0.3	0.50	-296087.3	**0.0**	148.1	3.0	**0.0**	1.6	3	0.6
3000	0.3	0.75	-297164.3	0.3	171.1	3.0	**0.0**	1.2	8	0.9
3000	0.5	0.25	-295577.6	0.4	193.2	6.6	**0.0**	2.5	2	0.7
3000	0.5	0.50	-297751.0	**0.0**	23.4	5.2	**0.0**	2.0	9	0.8
3000	0.5	0.75	-298465.0	**0.0**	25.5	5.1	**0.0**	1.8	10	—
—	—	—	—	0.4	77.6	1.4	**0.0**	0.6	100	0.2

Results for MCA on Set 2

			CMSA (RP)		dapcstp		ILS		Exact
n	p_{arc}	Opt	Value	Time (s)	Time (s)	N	Value	Time (s)	Time (s)
500	0.1	10172.1	**10172.1**	35.8	0.1	10	10560.1	3.6	0.0
500	0.3	-9174.5	**-9174.5**	17.6	0.2	10	-9081.5	2.3	0.3
500	0.5	-23946.4	**-23946.4**	13.3	0.2	10	-23899.5	3.5	0.0
1000	0.1	-4382.9	**-4375.0**	69.1	1.2	10	-3758.0	2.1	0.4
1000	0.3	-60628.1	**-60628.1**	34.4	1.8	10	-60543.1	4.3	0.1
1000	0.5	-106726.2	**-106726.2**	14.4	2.4	10	-106691.7	3.6	0.1
5000	0.1	-527496.4	**-527400.6**	823.3	219.3	2	-513849.1	2.3	3.7
5000	0.3	-1372718.9	**-1372321.9**	214.6	151.1	7	-1369569.5	4.8	3.9
5000	0.5	-1951128.4	**-1951128.4**	93.6	95.1	10	-1950230.7	8.3	3.4
—	—	—	—	146.2	52.4	79	—	3.9	1.3