Implementation of a matroid-based algorithm for finding optimal branching systems

[1] Waterloo, Ontario. Canada. * nik.dattani@gmail.com

September 9, 2022

Abstract. We implement an algorithm to solve the optimal branching systems problem. This problem generalizes min-cost max-flow and many other problems.

The sections below are organized as follows.

1 Terminology

Directed tree: A directed graph (digraph) that would be a tree if we ignored the directions of the edges.

Arborescence: A directed tree with a "root" such that every node of it has a unique path to it from the root. There's always exactly one root.

Spanning subgraph of $G = (V, E)$: A subgraph (V, B) of G such that $B \subseteq E$.

Branching system of $G = (V, E)$ with $|R|$ specified root nodes: A collection of $|R|$ edge-disjoint arborescences rooted from any of the root nodes, with each arborescence being a spanning subgraph of $(V \backslash (R \backslash r), E)$ with r being the root of the arborescence and R containing all specified roots. By "edge-disjoint" we mean that none of these arborescences share any edges. Also, every node has at most $|R|$ incoming edges (at most one from each arborescence), and multiple spanning arborescenes are allowed to be rooted at the same node.

Optimal branching system of a weighted digraph with k specified root nodes: the branching system with the lowest total weight.

2 Algorithm

2.1 Quick overview

For an arborescence to be a spanning subgraph of $(V \backslash (R \backslash r), E)$, it needs to have $|V| - |R|$ edges. Since branching systems are by definition comprised of $|R|$ edge-disjoint spanning arborescences, all branching systems have exactly $|R|\,(|V| - |R|)$ edges. The *optimal* branching system is the least-weight set of edges with size $|R|\,(|V| - |R|)$ that is independent in both matroid M_1 and in matroid M_2, which are described below:

- $M_1{=}(E, F_1)$ in which $f \in F_1$ has no edges entering any root nodes, and at most $|R|$ edges entering each other node of the graph;
- $M_2{=}(E, F_2)$ in which $f \in F_2$ has an arboricity of at most $|R|$. The **arboricity** is the minimum number of forests into which a set of edges can be partitioned.

2.2 Exponential speed-up by exploiting matroid theory

Comments on the cost of a naive algorithm

For a complete directed graph with $|V| = 20$ nodes and $|R| = 3$ roots, there's $\binom{19}{3}$ different ways to have $|R|$ edges entering each of the 17 non-root nodes, and therefore $\binom{19}{3}^{17} \approx 6 \times 10^{50}$ elements of F_1 that could be constructed in this way. Therefore it would be impractical to fully enumerate F_1 and then to try to find the set of size $|R|\,(|V| - |R|)$ in $F_1 \cap F_2$ that has the optimum weight.

Benefits of using weighted matroid intersection

Since the edges in E are all weighted, we can use one of many oracle-based *weighted matroid intersection* algorithms to find (in polynomial time) the lowest-weight set in $F_1 \cap F_2$ *without* listing all of F_1 or F_2. These algorithms typically start with the empty set, and add elements from $F_1 \cap F_2$ to it one-by-one such that while its growing in its size (number of elements), it always has the optimum total weight out of all sets in $F_1 \cap F_2$ with that specific size; when the size becomes $|R|(|V| - |R|)$ we will know that we have the optimum-weight set of $|R|(|V| - |R|)$ edges that is in $F_1 \cap F_2$. When selecting which element from $F_1 \cap F_2$ to add at each stage, we do not need to know the entirety of F_1

nor F_2 since all edges in E are weighted, making it possible to decide an optimal choice for which element of E is best to add: out of all elements in E that have not yet been included, there will be a preferred choice based on their weights, and we can used the oracles for F_1 and F_2 to determine whether or not our preferred choice is suitable or not, and continue until we find the best edge to add.

Cost of weighted matroid intersection

With the above paragraph considered, the total cost for solving the weighted matroid intersection problem becomes dependent on the number $|E|$ of edges and on the cost of the oracles that check whether or not a set of edges is in F_1 and F_2 respectively. Various polynomial time algorithms exist for solving this problem. In the implementation described here, the cost is $\mathcal{O}\left(|E|^4 + |E|^3\theta\right)$ where θ is the maximum cost of asking the oracle for F_1 or F_2 whether or not a candidate set of edges belong in the corresponding matroid.

The F_1 oracle

Given the incidence matrix for a graph, it is quick and easy to check whether or not a subset of edges is in F_1 (more details about this are given in the Appendix, along with code that accomplishes this task with only 1 line).

The F_2 oracle

Implementing the oracle to check whether or not a set of edges is in F_2 is much harder to do efficiently than it is for F_1, but can be done in polynomial time by solving the *matroid partitioning problem* to see if the graphic matroid corresponding to the graph formed by the set of edges can be partitioned into $|R|$ or fewer partitions. The matroid partitioning procedure requires another oracle, which tells us whether or not a set of edges has any cycles, but this can also be done efficiently since the existence of cycles implies the existence of linearly independent columns in the graph's incidence matrix, which can be determined in polynomial time by Edmonds' algorithm for Gaussian elimination in the Galois field GF(2).

Cost of querying the F_2 oracle

As with the weighted matroid intersection problem, various polynomial time algorihms exist for solving the matroid partitioning problem. In the implementation described here, the cost is $\mathcal{O}\left(|E|^4\varphi\right)$ where φ is the maximum cost of asking the oracle whether or not the candidate set of edges has any cycles.

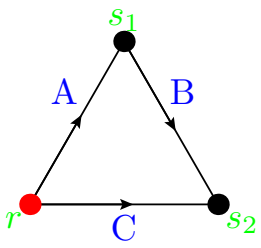
Total cost

Overall we are doing $\mathcal{O}\left(|E|^4 + |E|^3\theta\right)$ operations to solve the weighted matroid intersection problem described above, with $\theta = \mathcal{O}\left(|E|^4\varphi\right)$, and φ being the cost of doing Gaussian elimination in the Galois field GF(2). This leads to an overall cost of $\mathcal{O}\left(|E|^4 + |E|^7\varphi\right)$ or simply $\boldsymbol{\mathcal{O}\left(|E|^7\varphi\right)}$.

3 Pedagogical examples

The following examples are simple enough for us to solve the problem both directly and by using the algorithm described above which takes advantage of matroid theory. These will hopefully help the reader understand the optimal branching systems problem. In all graphs, the red nodes are the specified roots.

3-nodes, 1-root



We can directly observe that only two possible branching systems exist:

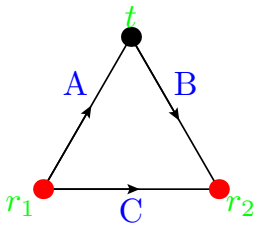
$$\mathscr{B}_1 = \{A, C\} \tag{1}$$
$$\mathscr{B}_2 = \{A, B\} \tag{2}$$

The independence families of F_1 and F_2 can also be listed completely for this graph:

$$J_1 = P(\mathscr{B}_1) \cup P(\mathscr{B}_2) \tag{3}$$
$$J_2 = P(\mathscr{B}_1) \cup P(\mathscr{B}_2) \cup P\{B, C\} \tag{4}$$

Finally, we have that $|R| = 1, |V| = 3, |R|(|V| - |R|) = 2$ so we are only interested in the sets in $J_1 \cap J_2$ that have two edges in them. These are precisely $\mathscr{B}_1$ and $\mathscr{B}_2$. Therefore finding the minimum-weight set with $|R|(|V| - |R|) = 2$ edges in $J_1 \cap J_2$ is equivalent to solving the optimal branching systems problem by finding all possible branching systems directly, and then optimizing over them.

3-nodes, 2-roots



We can directly observe that no branching systems exist for this graph because it's impossible for an arborescence rooted at r_2 to be a spanning subgraph of the original graph (no arborescence rooted at r_2 can contain all of the original graph's non-root nodes), and since there's only one edge connecting r_1 to s, it is not possible to have $|R| = 2$ edge-disjoint arborescences rooted from r_1.

The independence families of F_1 and F_2 can also be listed completely for this graph:

$$F_1 = \{\emptyset, \{A\}\} \tag{5}$$
$$F_2 = P\{A, B; C\} \tag{6}$$

Although $F_1 \cap F_2$ contains the set $\{A\}$, it does not contain any sets that have $|R|(|V| - |R|) = 2$ edges, so $F_1 \cap_2 F_2 = \emptyset$. Therefore again we find that finding the minimum-weight set with $|R|(|V| - |R|) = 2$ edges in $J_1 \cap J_2$ (in this case no such set exists) is equivalent to solving the optimal branching systems problem.

References