é

Optimum Branching Systems [OBS],

a talk celebrating a very special scholar and gentleman,

Gérard Cornuéjols.

May 31, 2022

Carnegie Mellon University

In the early 1960s, about 60 years ago,
optimum matchings in a graph, and optimum matroid intersections,
were found to have polynomial-time algorithms (i.e., to be in P).

The algorithms provided proofs of some "NP-CoNP" ("nice") polytopes,
which appear in a dozen Combinatorial Optimization books.

Around the same time various NP problems were conjectured
to be not in CoNP, and hence not in P.

I presumed that many other optimization problems
would be found to be in P.

Here in my Last Waltz, I am confounded that they have not been.

The blossom algorithm for optimum matchings has been well implemented.
I still have in my back garage Bill Pulleyblank's punch cards of it.
And there have been more implementations of it.
What good is it? Well, it got Bill a job as a VP of Research at IBM.

Knowledge of polynomial-time algorithms is like knowledge
of astrophysics or maybe algebraic topology. i.e., not very practical.

Curiously, no one studies how to solve a quartic equation by radicals
but every pure mathie studies why it can't be done for quintic equations.
Similarly c.s. mathies are obsessed with trying to prove
what can not be done for the traveling salesman.

Maybe the value of books about matchings and matroids
lies in being a sophisticated inspiration for programmers.

My futile dream has been trying to find some smart people
to implement matroid intersections

and/or Optimum Branching Systems (OBS).

Doing it should be simple and lead to great fame and employment.

However it requires some novel sophistication which involves
calling of subroutines for recognizing when a set is independent
in various specified matroids. I guess this scares people.
Maybe another impediment is feelings that implementation
is not spiritual like writing papers.

For many years I have not been in a position to bully students into anything.
Beginning a hundred years ago, my beliefs that I had found help
for implementing OBS have been fiascos.

Optimum Branching Systems (OBS)

One of the concrete discrete problems with a known polynomial-time algorithm is the "Optimum Branching Systems" (OBS) problem:

Given a directed graph with a cost for each edge
and with a set of specified desired root nodes,
find (if they exist) a least expensive set
of edge-disjoint spanning directed trees ('branchings')
rooted at the specified nodes.

It is simple to reduce to the case of finding if they exist
a least expensive set of k edge-disjoint branchings
all rooted at a node r.
For convenience we concentrate this way.

For a single branching at a single root node
there is a beautiful direct easy algorithm,
but for k > 1 the only beautiful algorithm I know is described as
a case of general matroid partitioning and intersections.

My dream is still to persuade some smart people to implement it.

I'll describe here how to do OBS using the matroid methods
which are in many Combinatorial Optimization books.

Curiously none of the books mentions OBS
though it is an easy and I think the most beautiful application
which does not reduce to network flows or the assignment problem.

OBS was a main concrete motivation
for me to study matroid partitioning and intersections.

I did not stress that because I felt I would eventually find a direct algorithm
beautiful like the one I knew for finding a single optimum branching.
However I have still not found an understandable algorithm
except by the matroidal generalization.

There is a 1977 PhD thesis
which proposes a direct implementation of OBS.

The author became a VP of the Bank of Mexico
and then the President of the
National Institute of Statistics, Geography and Informatics (INEGI)
with thousands of employees.

Since the time he became available for the two of us to pursue OBS,
we have not been able to understand his method.
I begged that we cover the problem by getting help to implement
the simple matroidal way, but he chooses to not pursue it beyond his thesis.

We do know a simple good algorithm for OBS as a special case of good algorithms for matroid partitioning, matroid intersections, and simply finding disjoint branchings in a directed graph.

The **'Disjoint Branchings Theorem'** says
The maximum size
of a set B of edge-disjoint r-rooted spanning directed trees in G
= The minimum size
of a set C of edges of G which are directed into a set S of nodes
such that S is non-empty and does not contain root node, r.

There are good algorithms for finding a set B and a set C,
in particular by Lovász and by Tarjan.

A crucial secret corollary:
If a set D of edges in G is such that
(a) D has no edges entering node r,
and exactly k edges entering each other node of G
(a basis of a matroid M_1);
and
(b) D can be partitioned into k edge-disjoint spanning trees of G
(a basis of a matroid M_2)
then
D can be partitioned into k edge-disjoint directed spanning trees of G,
each rooted at r.

The situation is analogous to the case where, with edge-capacities = 1,
a min cost flow algorithm finds an edge set which still needs
to be partitioned into edge disjoint paths from source to sink.

The analogous thing for spanning directed trees rooted at a node r
is not as obvious and not as simple algorithmically.

And so OBS now entails finding a least cost set D of edges
which is both a basis of matroid M_1 and a basis of matroid M_2.
That is the **optimum matroid intersection problem**.

M_1 is the matroid on the set E of edges such that
a subset J of E is independent in matroid M_1 when
(a) J has no edges entering node r,
and at most k edges entering each other node of G.

M_2 is the matroid on the set E of edges such that
a subset J of E is independent in matroid M_2 when
(b) J can be partitioned into at most k forests.

It is obvious that (a) describes a matroid.
That (b) describes a matroid is **a corollary of matroid partitioning**.

Theorem 1.
An $f(\phi)=0$, non-decreasing, submodular set function, f(S),
of the subsets S of a finite set E,
is called a polymatroid function on E.

For any integer valued polymatroid function on E,
let F be the family of subsets J of E
such that for every non-empty subset S of J,
the cardinality of S is at most f(S).
Then M = (E,F) is a matroid.

Its rank function is, for every subset A of E,
r(A) = min[f(S) + cardinality of (A\ S)] for any subset S of A.

What does Theorem 1 have to do with matroid partitioning?

The rank function of a matroid is a polymatroid function,
and hence so is the sum of the rank functions
of any family H of matroids all on the same set E.

Hence the special case of Theorem 1, applied to this sum,
yields **a matroid M on set E as the "sum" of matroids H on set E.**

[I had hoped to understand the prime matroids relative to this sum,
but not much has come of that.]

Suppose we have an oracle
which for an integer polymatroid function, f(S) on E ,
gives the value of f(S) for any subset S of E.
Then the theorem gives an easy way to recognize
when a given subset J of E **is not independent**
in the matroid M determined by Theorem 1:
Observe a subset S of J having cardinality greater than f(S).

Does there exist an easy way to recognize
when a set J **is independent** in M? Yes.

The 'Matroid Partition Theorem' :
Where f is the sum of a given family H of matroid rank functions,
a set J is independent in M if and only if it can be partitioned into
a family of sets which are independent respectively in the matroids of H.

So recognizing independence in M is easy if recognizing independence
in each matroid of H is easy.

To me this good characterization meant that we should prove
the matroid partition theorem by a polynomial-time algorithm.

Having a good characterization relative to an oracle,
and having a good algorithm relative to the oracle
which proves the good characterization,
was a motivation for the subject.

As far as I know, "Matroid Partitioning" is the first time
what is now called NP appears in mathematics.

Because the greedy algorithm works for finding an opt basis in any matroid,
as well as for finding an opt spanning tree in an undirected graph G,
we do not even need an opt matroid intersection algorithm
in order **to find (if it exists)**
an opt system of k edge disjoint spanning trees in G.

Our algorithm needs a subroutine for recognizing forests in G.
I recommend that this be done by a subroutine for recognizing
linearly independent sets of columns in a binary matrix.
This subroutine feeds a matroid partitioning algorithm to recognize
sets of columns which are partitionable into k independent sets.
This partitioning is then a subroutine which feeds the greedy algorithm.

Instead of feeding the greedy algorithm,
to do OBS, matroid partitioning is used as a subroutine
to identify the independent sets of M_2,
one of the two input matroids.of an opt matroid intersection algorithm.

Sons and
Grandson.
Photo by Kathie.

Thank you
for reading.

Please
implement OBS.

Participants in the first matroid theory workshop, August, 1964.