God provides only a few glimpses of heaven: NP and some nice polytopes, i.e., NP ∩ coNP polytopes.

Jack Edmonds

<jack.n2m2m6@gmail.com>

To celebrate the work of Stephen Cook

in 50 Years of Computational Complexity.

Fields Institute, Toronto, May 9, 2019.

Mathematical Programming, Sorbonne, Paris, May 3, 2019

(Lucky to be born a few years before Steve.)

My talk is about some mathematical glimpses of heaven, particularly NP ∩ coNP.

NP is the set of predicates which for the instances that are true there is an easy proof.
i.e., there is a “good description” of the instances that are true

coNP is the set of predicates which for the instances that are not true
there is an easy proof that they are not true.
i.e., there is a “good description” of the instances that are not true.

P is the set of predicates for which there is a “good algorithm”
to decide whether the predicate is true or false.

In order to give the definitions some manageable mathematical meaning
“good” and “easy” means polynomial relative to the size of writing down the instance.

Using terminology that I didn't have at the time:

In 1960 I conjectured that NP ∩ coNP = P.

In 1966 I conjectured that NP ≠ P.

Possibly there are no proofs of them in general, which does not really matter.
What is important about them is they are true and useful in the real World.

Most known NP ∩ coNP predicates have been shown to be in P, including

- linear programming optimality,
- primality,
- **and various problems**
  which have filled books on Combinatorial Optimization
  which I will talk about here.

Combinatorial Optimization books, like most books on algorithms,
only talk about NP as part of NP completeness.

I will speak about the NP ∩ coNP philosophy behind the mathematics
of these books, which the books themselves do not address.

The overwhelming presence of NP predicates being NP-complete
instead of being coNP
has given term NP a bad reputation,
even in these books about NP ∩ coNP.

I did not expect that in the 1960s.

I hope you enjoy the NP ∩ coNP in this talk. However
I should say: the conjecture “NP ∩ coNP = P” is only fairly successful because

(1) it is possibly not true for unknown ‘inhuman instances’;
(2) independent NP ∩ coNP predicates have seemed rather rare; and
(3) there is at least one NP ∩ coNP predicate not yet known to be in P which if shown to be in P will cause great social havoc.

Some public security depends on:

(a) It is easy to find, and recognize as prime, a large random prime integer.
It is known to be easy to determine whether or not a given integer has a factor.
(b) It is difficult to find a factor, even when you know there is one, of a large integer.

By (a), the predicate
p(h,k) = “integer h has a prime factor j such that $j \leq k$” is in NP ∩ coNP.

If this p(h,k) is in P, then there is an easy algorithm for finding a prime factor of any integer h, thus making (b) not true. It is then easy to decipher cryptography which is based on hiding the factors of the product of two primes.

In the real world, algorithmic inefficiency is a good thing.

There is a career advantage to teaching only simplest cases of a theory.
Besides being most convenient for learning,
it provides for more directions of generalization (and more citations).
We follow this way here.
A mathematical subject which is too highly developed becomes ignored.

My favorite theorem is the **matroid-polytope intersection theorem**.

A matroid M is an abstraction of the linearly independent subsets
of the set E of columns of a matrix, M:

For any subset S of E, every (inclusion) maximal M-independent subset J of S
is the same size, called $r_M(S)$, the M-rank of set S.

Since the forest edge-sets of a graph are the independent sets of a matroid,
given by a matrix over the field where 1+1=0, and such that each column has two 1’s,
we see that matroids do have an interesting variety of structure.

The popular ‘greedy algorithm’,
for finding an optimum total weight spanning tree of a graph,
generalizes in two ways:

The Greedy Theorem [1964].
For any matroid M on the set E and any weighting c_e of its elements e,
the “greedy algorithm” finds the 0,1 vector x of an independent set of M
which maximizes cx in the polytope
$P(M) = \{x \geq 0$: for every subset S of E, $\sum\{x_e$: e in S$\} \leq r_M(S)\}$.

Corollary. The vertices of P(M) are the 0,1 vectors of the independent sets of M.

You should be ashamed if, instead of this, you teach only
the Kruskal spanning tree version, presented by Boruvka in 1931.

Suppose we have any two matroids M^1 and M^2 on set E.
In general the 'independence system', $M^1 \cap M^2$,
consisting of sets independent in both M^1 and M^2 is not a matroid.
We want to describe the convex hull, say $P(M^1 \cap M^2)$, of the vectors of sets in $M^1 \cap M^2$.

Every common vertex of any two polyhedra is a vertex of their intersection.
In general however, there will be other vertices as well.
It is rare for two polyhedra to fit together so neatly
that the only vertices of the intersection are the common vertices.

The Amazing Matroid Polytope Intersection Theorem [1965]: **$P(M^1 \cap M^2) = P(M^1) \cap P(M^2)$.**
And the dual optimum is integers when the c of maximizing cx over $P(M^1) \cap P(M^2)$ is integers.

When the c is all ones we have the 'Cardinality Matroid Intersection Theorem':
$\max \{ |J| : J \in M^1 \cap M^2 \} = \min \{ r_{M1}(S) + r_{M2}(E-S) : S \subseteq E \}$.

Proved by proving that a polynomial time algorithm more complicated than the greedy algorithm
maximizes any cx over $P(M^1) \cap P(M^2)$ by a common independent set of M^1 and M^2,
and the dual optimum is integers if the c of the primal objective cx is integers.

The Konig-Egervary Theorem [1931] about the optimum assignment problem is a special case where each matroid, $M^1$and M^2, is a rainbow matroid, which means a subset J is independent in matroid M^k when the members of J have different M^k-colors, where for each k = 1 or 2, the set E of elements is M^k-colored, that is partitioned, in any way.

In other words E is the edge-set of a bipartite graph G
where each edge goes from a boy node to a girl node,
where there is a girl node for each color in M^1, and a boy node for each color in M^2.
A subset J of E is independent in the girl matroid M^1 when the edges of J hit distinct girls
A subset J of E is independent in the boy matroid M^2 when the edges of J hit distinct boys.
$J \in M^1 \cap M^2$ when J is a matching in G, that is, no node of G is hit by more than one edge in J.

The Optimum Branching Systems problem (OBS) is
Given a directed graph G, specified root nodes r(i),
and a cost for each edge of G, find a least cost collection
of edge-disjoint directed spanning trees in G,
rooted respectively at the nodes r(i), i.e., **r(i)-branchings in G**.

By letting M^1 and M^2 be certain matroids, matroid intersection solves OBS.
This should have some use in O.R. It does for a single branching.

The extensively treated mincost network flow problem is a special case of OBS.
However OBS does not reduce to it.
Matroid intersection is the only approach known for solving OBS.
Curiously, the simplest way known to describe an algorithm for OBS
is by matroid intersection for general matroids.

The matroid intersection method for OBS is in the process of being implemented
by a group in Lisbon, which includes José Rui Figueira <figueira@tecnico.ulisboa.pt>.

Matroid algorithms assume subroutine sources of the matroids M
which say when a set is independent in M, or yield $r_M(S)$ for a given set S.

If the source of a matroid M is a matrix M we need a good exact algorithm
for determining the rank of any given subset of the columns in M.
Each time a column is processed in a matrix A by usual Gaussian elimination
the number of bits in each entry of A tends to double, unless it is rounded.
Hence the bit complexity of getting an upper triangular submatrix by row operations is exponential.

Here is a method for exact linear solving which works easily for solving a system of linear equations,
or for getting a submatrix of a matrix A to be upper triangular, over any integral domain.
Even though the algorithm does some division, the quotients stay in the integral domain.
Where each entry of A is some linear combination of a <u>bounded</u> <u>number</u> of indeterminate symbols
the entries of any derived A^k are polynomials of polynomially bounded size.

"Solve" in the following way, any matrix A^0 over any integral domain. $a_{i(0),j(0)}{}^0$ is defined to be 1.
The matrix A^k is obtained from A^{k-1} as follows. Each row i(h), $1 \le h \le k$, is the same in A^k as in A^{k-1}.
Each column j(h) of A^k is all zeros except for entries $a_{i(g)j(h)}{}^k$, $1 \le g \le h$.
For any other row i and column j , let $a_{ij}{}^k = (\, a_{i(k),j(k)}{}^{k-1}\, a_{ij}{}^{k-1} - a_{i,j(k)}{}^{k-1}\, a_{i(k),j}{}^{k-1}\,) \,/\, a_{i(k-1),j(k-1)}{}^{k-1}$

Thm. (Systems of Distinct Representatives and Linear Algebra, 1966). The entries of each matrix A^k magically stay in the integral domain. Where the entries of A, with n rows, are numbers having at most t bits,
the entries of any A^k have at most n(t+log n) bits. (The algorithm turns out to be closely related to the once famous and now forgotten method of Lewis Carroll of 'Alice in Wonderland' for computing determinants.)
The algorithm extends to "Q-pivoting in Q-matrices" for a useful exact lp simplex method,
and for getting from a basis of a matroid given by a matrix to a nearby basis.

For any graph G, a matching J in G means a subset J of the edges
such that each node is hit by at most one edge of J.
Let P(G) be the polytope whose vertices are the 0,1 vectors of the matchings J in G.

The Konig-Egervary Theorem about the optimum assignment problem,
the mentioned prototype of matroid-polytope intersection, implies:
When G is bipartite, the convex hull of the 0,1 vectors of the matchings in G is
P(G) = { $x \geq 0$: for each node v of G, $\sum(x_e$: edges e hitting node v) ≤ 1}

Indeed, I think that the K-E theory [1931] is the first instance of
linear programming duality, and the first instance of NP ∩ coNP.

What about P(G) when G is any graph, not necessarily bipartite?
Matching Polytope Theorem: The convex hull of the 0,1 vectors of the matchings in G
is **P(G) = { $x \geq 0$: for each node v of G, $\sum(x_e$: edges e hitting node v) ≤ 1 }; and for each**
set B of nodes in G, $|B|$odd and ≥ 1, $\sum(x_e$: edges e with both ends in B) $\leq (|B| - 1)/2$ }.

The optimum matching problem, max{ cx : x in P(G)} is a very large linear program –
but optimality is still an instance of NP ∩ coNP.

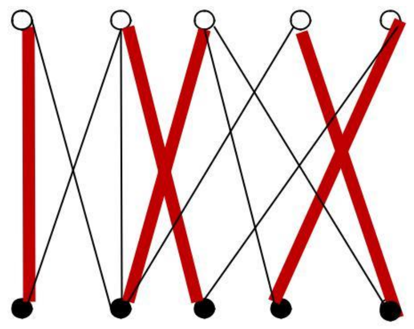
NP and P have origins in “the marriage theorem”,
a corollary of the bipartite K-E matching theory:

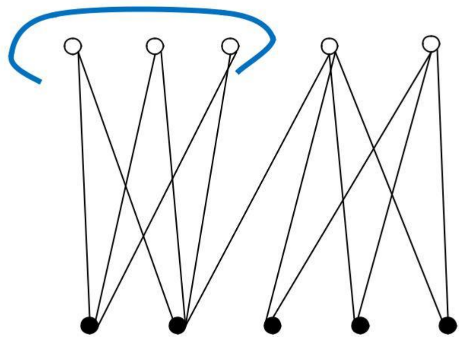
> A matchmaker has as clients the parents of some boys and some girls where some boy-girl pairs love each other. The matchmaker must find a marriage of all the girls to distinct boys they love or else prove to the parents that it is not possible.

The input to this marriage problem is a bipartite graph G with boy nodes, girl nodes, and edges between them representing love. A possible legal marriage of <u>some</u> of the girls to <u>some</u> of the boys is represented by a subset M of the edges of G, which is a matching. The matchmaker’s problem is to find a matching which hits all the girl nodes
Or else prove to the parents that there is none.

The Marriage Theorem: Either $A_1(G)$ = [The girls can marry distinct boys they love] or else
$A_2(G)$ = [there is a subset S of the girls which is bigger than the set of boys who someone in S loves},
Not both.

In other words the Marriage Theorem is: $A_1(G) = \text{not } A_2(G)$.
And so **$A_1(G) \in NP \cap coNP$**, since both $A_1(G)$ and $A_2(G)$ are clearly in NP.





From Matt Baker: https://mattbaker.blog/2014/06/25/the-mathematics-of-marriage/

A gorgeous application of the marriage theorem is the following magic trick:
Deal a deck of cards into 13 piles of 4 cards each.
Select one card from each pile so that no value (Ace through King) is repeated.
This can always been done, no matter how the cards were originally dealt !

Matt deduces this from the **Marriage Theorem** said in a way which pure mathies love:

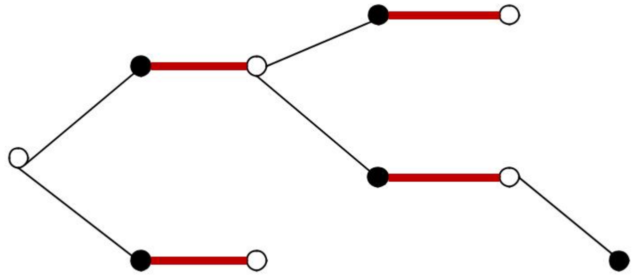
The girls can marry distinct boys they love if and only
for every subset S of the girls, $|S| \geq$ the number of boys which S together loves.

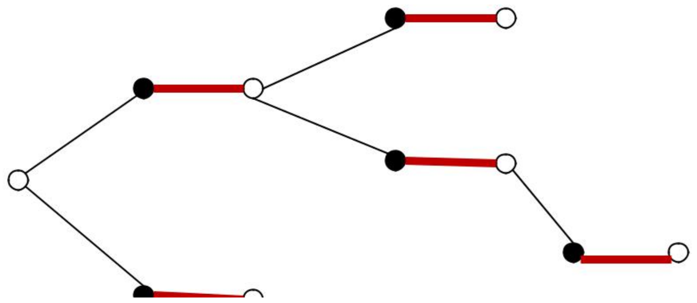
In 1960, while I was working as an operations researcher on civilian projects in Washington,
this way of stating, and an algorithmically inefficient proof of, the Marriage Theorem
blew my mind with puzzlement.

Is looking at every subset of girls easier than looking at every matching?

Pure mathies love proofs which are bad as algorithms and convey the impression
that there is no easy way to actually get what the theorem says exists.
The proof in the book called “Proofs From The Book” is like that.

An Algorithmic Proof of the Marriage Theorem: Let M_1 be any matching in G. If there is some girl node r who is not hit by M_1 grow a tree T such that every path in T starting at node r alternates between edges not in M_1 and edges in M_1. If T reaches boy node t which is not hit by matching M_1, then change edges in the path in T between t and r to get a larger matching, M_2. If T reaches a state of not being able to grow more and not containing a node t not hit by M_1, then the boy nodes of T are the only nodes of G which the girl nodes of T are joined to by edges of G, and there is one more girl node in T than boy nodes in T.
And so [A_1(G) ? or A_2(G) ?] **ϵ P.**





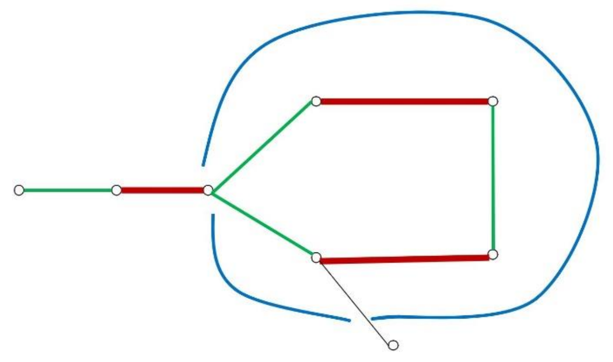
Thinking about all of this in 1960, it occurred to me that maybe a general predicate is in P whenever it is in NP ∩ coNP.

If graph G is not bipartite the Marriage algorithm is messed up
by the discovery of an edge e joining two “girl nodes” of T.

Roughly speaking, the secret to patching up the algorithm for a **non-bipartite** generalization of the K-E ‘Hungarian Method’ and the Marriage Theorem is:

Eureka! You shrink!

T+e contains a cycle C (called a blossom) with an odd number of edges.
Shrink blossom C to become a “pseudo girl node” of T and continue growing T.



The convex hull of any finite set Q of vectors in the space of vectors indexed by the edge-set E of graph G, is a polytope, the bounded solution-set of a finite set L of linear inequalities.
Optimizing cx over members of Q is the same as optimizing cx subject to L.
Normally for an easily described finite set Q of points in R^E,
the number of inequalities needed is exponential relative to $|E|$.
Dantzig's simplex algorithm would then be very exponential time.

An arrangement of the girls and boys at a round banquet table so that each is, on each side, next to someone loved, is the same as a TSP tour in the graph G.
Suppose we want a TSP tour C which optimizes the sum of love-weights on edges in C.

The TSP, in a yes-no form, is given
t = (a graph G, a love-weight c_j for each edge j, a tour H in G),
then where p(t) = "H is not the largest total love-weight tour in G",
is p(t) true or false? This predicate is clearly in NP. Is it in coNP?
TSP is represented by the set Q of the 0-1 vectors x of the TSP tours in G
and a linear function cx to be optimized over Q.
Is there an <u>NP</u> set L of linear inequalities such that
optimizing cx over solutions to L is the same as optimizing cx over Q?

In 1954, George Dantzig, with Ray Fulkerson and Selmer Johnson,
solved a particular 48-city traveling salesman problem
by using linear programming.

Their linear inequality system L’ , satisfied by a 0,1 vector x
if and only if x is the vector of a TSP tour, is NP, though it is exponentially large.
Unfortunately the solution-set of L’ has some extreme points
which are fractional rather than TSP tour vectors,
and so optimizing subject to L’ might not be by a TSP tour.

Their work was a motivation for Ralph Gomory in 1958
to introduce cutting plane methods for integer linear programming.
There was no mention of ‘easy’, meaning polynomially bounded time.
Until my student Vasek Chatal, there was no stated theorem other than that
the algorithm is finite. The fact that cuts and lp optimizations do not commute
created skepticism. What is the “cut-depth” of a cutting plane method?

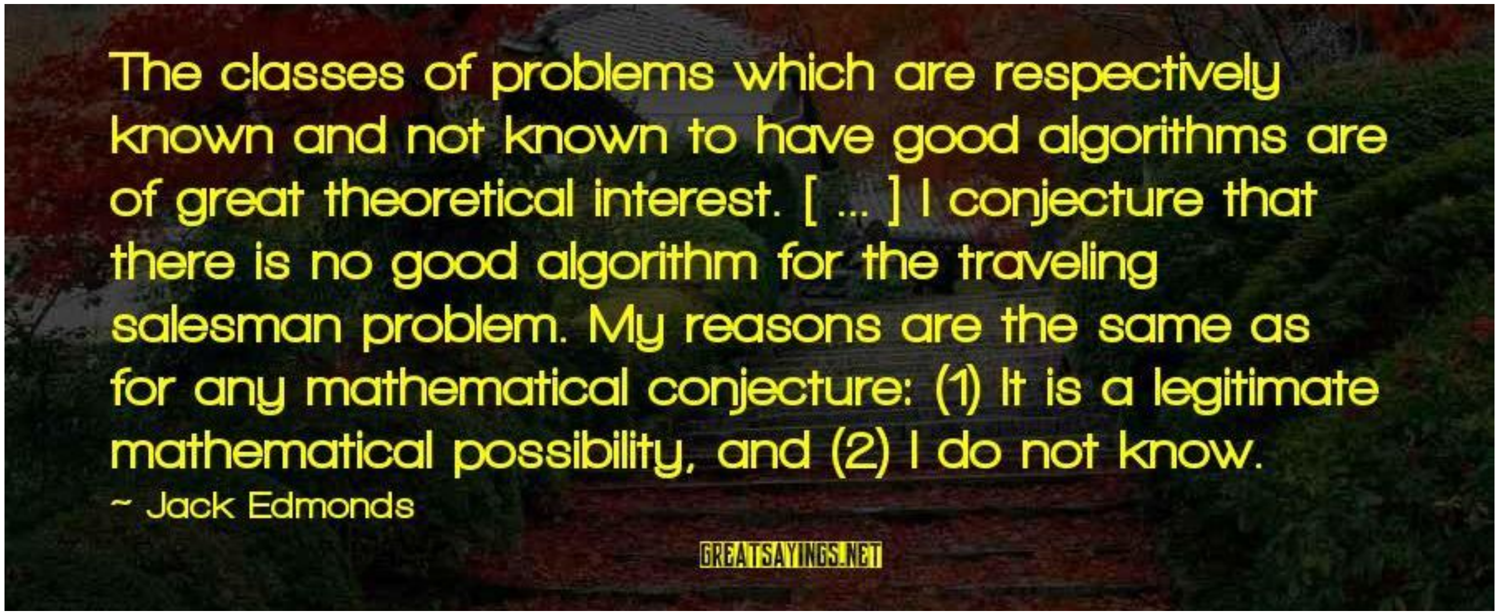
The 1954 TSP paper of Dantzig et al prompted me in 1961
to notice the simple result that

Theorem. If there is an NP description of the set L
of inequalities describing the convex hull of finite point-set Q,
as well as an NP decription of the members of Q, then
it does not matter that L is exponentially large in order to have
the predicate p(x) = [point x of Q is optimum over Q] in NP ∩ coNP.

And <u>maybe</u> that puts "optimize over Q" algorithmically into P.

It seemed plausible that if there is an easy description of Q
then there might be an easy description of a linear system, L,
describing the convex hull of Q. That is an important step
toward an easy general algorithm for optimizing over Q.

I was never able to find an NP description of a linear system L whose solution-set is the convex hull of the vectors of the TSP tours in a complete graph, G, and so I conjectured in **1966** that **NP ≠ P**, in particular that TSP is not in P.



The classes of problems which are respectively known and not known to have good algorithms are of great theoretical interest. [...] I conjecture that there is no good algorithm for the traveling salesman problem. My reasons are the same as for any mathematical conjecture: (1) It is a legitimate mathematical possibility, and (2) I do not know.
~ Jack Edmonds

GREATSAYINGS.NET

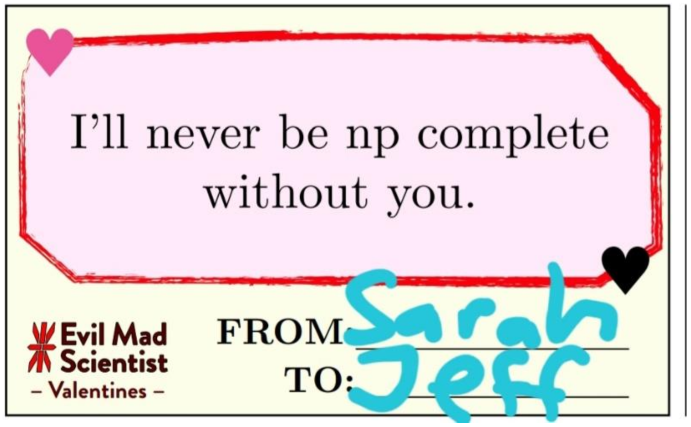
The conjecture **NP ≠ P** is now usefully presumed.
It is at the top of the Clay list of unsolved math problems. Stated there as the “P vs. NP question”, it does not need to be called Edmonds’ conjecture.
However the hypothesis, **NP ≠ P,** is used all the time and its failure would be cataclysmic to mathematical programming.
Finding a mathematical proof has defied enormous attention for over 50 years.

In 1971 Steve Cook showed that there exists an NP predicate, SAT, which is complete. He also showed that Clique is complete.
Dick Karp then showed that many other natural combinatorial problems, including the TSP, are NP-complete.

An easy algorithm for any one of these NP-complete predicates implies an easy algorithm for any NP-complete predicate.
Hence **NP ≠ P** implies that there is no easy algorithm for any of them.

One might as well conjecture
the possibility that there is no mathematical proof of **NP ≠ P**.
Most of true math does not have any mathematical proof.

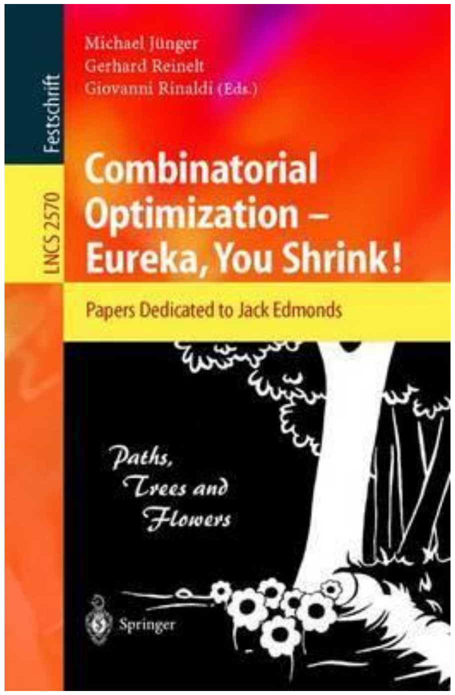
Having such a proof does not really matter since it is true
as long as we do not know a polytime algorithm
for an NP complete problem.


I'll never be np complete without you.
Evil Mad Scientist – Valentines –
FROM: Sarah
TO: Jeff

How about "I'll never be NP ∩ coNP without you." ?

Throughout the 1960s , while looking for easy algorithms of course I found some polytime reductions between problems, but **I never dreamed that there exist NP-<u>complete</u> problems.**
It is now a cosmic tragedy that there being so many NP-complete problems has even given NP a bad name, when NP itself is a wonderfully positive thing.

During the 1960s, **as part of the search for TSP in P**, I did find some NP sets Q of points to have of convex-hull determining NP linear systems, L, and from that P algorithms – like matchings and matroid intersections.
Another was the "Chinese Postman's Problem".
They were all hoped to be ingredients for solving TSP.



In 1961 while working on combinatorial optimization problems in the new Operations Research Section of the Mathematics Division of the U.S. National Bureau of Standards, my wonderful mentor Alan Goldman arranged through his wonderful Princeton mentor, Professor Tucker, thesis supervisor of Nash's equilibria and much else, for me to be a junior participant in a Summer long workshop at the RAND Corporation, across from Muscle Beach, down the street from Hollywood.
Seemingly every known combinatorist participated.

I needed a successful example of the NP idea, such as the non-bipartite matching problem. The day before my scheduled lecture to the eminences I still did not have it. Then Eureka, I did! You shrink blossoms!
So the lecture was a success, and attracted discussion. Professor Tucker offered me a job at Princeton, chairing the Combinatorics and Games Seminar.
So my first glimpse of mathematical heaven was at the notorious RAND Corporation – as exciting as the RAND Corporation's Daniel Ellsberg revealing the Pentagon Papers.

In retrospect I felt slightly foolish to be influenced to conjecture **NP ≠ P**
by my failure to find a <u>nice</u> NP linear-system description L
of the convex hull of TSP tours.

There began to appear easy algorithms for problems of
optimizing cx over a combinatorially described set Q of points x
which did not reveal nice NP descriptions of a system, L, linearizing Q.

Of course a polytime algorithm for optimizing any cx over x in Q
is an NP description of such an L ,
but I had expected some not-formalized nicer description of an L.

A main example of such a loco problem over a set Q with an elegant algorithm in P
and terrible facets in the linearization of Q
was George Minty, and independently Najiba Sbihi for the cardinality version,
finding a polytime algorithm for max cx over Q ≡ {stable sets S of nodes in a claw-free graph}.

An induced subgraph of a graph G means
a subset S of nodes of G plus every edge of G having both ends in S.
A stable set S of G means that the subgraph induced by S has no edges.
A claw means an induced subgraph consisting of 4 nodes and 3 edges which hit one of the 4 nodes.

The line-graph G' of a graph G means that the nodes of G' are the edges of G, and 2 nodes
are joined by an edge in G' when the corresponding 2 edges in G hit a common node in G.
Clearly any line graph G' is claw-free, and the stable sets in G' are the matchings in G.
However there is a rich existence of claw-free graphs which are not line graphs.

The algorithms of Minty and Sbihi generalize an algorithm for optimum matching in G.
In the almost 40 years since their lovely algorithms there have been many papers
about the terrible linearization of the stable sets in a claw-free graph..

Finally **Faenza, Oriolo, and Stauffer [2012]** have usefully shown
that a linearization of the set of stable sets in a claw-free graph G is "nice" after all,
by using certain decompositions of G and **extended formulations**.

Another very nice case , almost not known at all, of a polytime algorithm
minimizing any cx over x in a combinatorially described set Q
where facets look terrible in a linearization of Q, is Huffman's famous algorithm for
getting an optimum prefix coding for an alphabet with any given frequency distribution.
Huffman coding is very valuable for compression.

A prefix coding of an alphabet a ≡ (a1,a2,...,an) is an encoding p ≡ (p1,p2,...,pn)
where the components pk are binary strings such that none is an initial substring of another.
The length vector x ≡ (x1,x2,...,xn) of p is where each xk is the length of string pk.
The frequency vector of a text T using alphabet a is c ≡ (c1,c2,...,cn)
where each ck is the number of times that letter ak appears in text T.

And so for a given prefix encoding p of a text T in alphabet a, cx is the total size of the encoded T.
Huffman [1952] discovered a now well-known beautiful polytime algorithm for, given a text T
using alphabet a, find a prefix encoding p of a which minimizes cx, the total size of the encoded T.
This is the same as minimizing cx over x satisfying a linear inequality system L where
the solution-set of L is the convex hull of the length vectors x of prefix encodings, p.
Jean-Francois Maurras, Thanh Hai Nguyen, and Viet Hung Nguyen, have shown in the paper "On the linear description of the Huffman trees polytope" [2012] that the set of facets of L is terrible.
Volker Kaibel and Kanstantsin Pashkovich have shown in the paper "Constructing Extended Formulations from Reflection Relations" [2013] that an L for Huffman polytopes can be nicely described by certain decompositions and **extended formulations**.

Fulkerson showed that a graph G is perfect if and only if the vertices of P(G) ≡ $\{ x \geq 0 : \text{for every clique } C \text{ in } G, \sum(x_u : u \text{ is a node in } C) \leq 1\}$ are the vectors of stable sets.

G is a Meyniel graph means that it has no induced simple odd cycle with more than 3 nodes and at most one chord.
Meyniel graphs are a rich class of perfect graphs.
There are some papers on algorithms for an optimum stable set in the complement of a Meyniel graph, but even though the Fulkerson linearization is especially nice, finding an optimum stable set has been elusive except by using a seminal decomposition method of Burlet and Fonlupt in “Polynomial Algorithm to Recognize a Meyniel Graph” [1984] and “Polyhedral consequences of the amalgam operation” [1994].

A decomposition method somewhat like that of Burlet and Fonlupt together with certain **extended formulations** has been used by Conforti, Gerards, and Pashkovich, in the paper “Stable Sets and Graphs with no Even Holes” [2018], to get a nice linearization and an algorithm for the stable sets in (not perfect) graphs G with no induced “steamboats”. It is not necessary for us to define steamboat. They are an induced subgraph exclusion which enables the decomposition technique to work.

Extended formulation is an inverse of eliminating, as in Fourier-Motzkin elimination, some of the variables say x’ from a system L of linear inequalities in variables (x,x’) to get a system L’ of linear inequalities in the variables x such that the solutions of L’ are the valuations of x such that for some valuation of x’ , (x.x’) is a solution of L.

L’ generally requires many more inequalities than L and is way more complicated than L.

The algorithm is named after Joseph Fourier and Theodore Motzkin who independently discovered the method in 1827 and in 1936, respectively.

To optimize cx over L’ , it is easier to optimize cx + c’x’ over L with c’ = all zeroes. Hence large and complex systems L’ are sometimes rendered more simple by an extended formulation L of L’.

Sometimes a beautiful but large L’ can be obtained from a beautiful smaller L by elimination – so then L is a ‘nice’ extended formulation of L’. If L has its number of inequalities bounded polynomially by its number of variables, then L is called a **compact formulation** of L’.

Considerable effort was devoted to proving that the NP system L’ for the matching polytope does not have a compact formulation. Proving this was a great achievement by Thomas Rothvoss in the paper “The matching polytope has exponential extension complexity”, which was **Winner of the STOC 2014 Best Paper Prize.**

The subject of compact formulations became popular because of an invalid polynomial time algorithm for TSP presented by an incompetent full Professor, whom we will call Tom, in the University of Waterloo, Department of Combinatorics and Optimization. It was based on a claimed compact extended formulation of TSP.

Tom kept insisting that his TSP algorithm using a compact extended formulation was valid and this caused Mihalis Yannakakis to show that TSP has no compact extended formulation, assuming a symmetry condition:

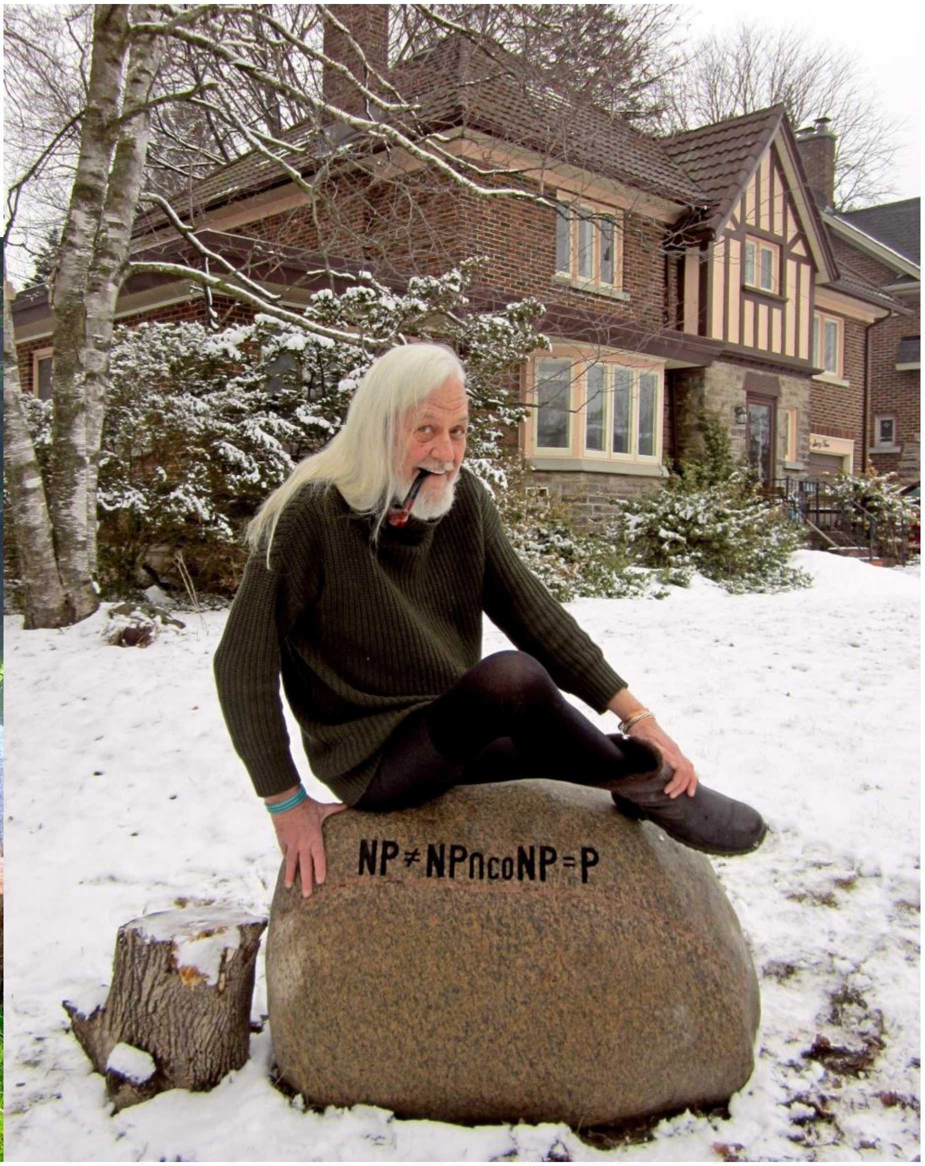
"Expressing combinatorial optimization problems by linear programs" [1991].
The corresponding impossibility result without the symmetry restriction was finally proved by Fiorini, Massar, Pokutta, de Wolf, and Tiwary (2012) in another award winning STOC paper.

Tom's work was probably prompted by the discovery shortly before his work of some beautiful new polyhedra, called coflow polyhedra, by student K. Cameron and me, which were derived from their compact extended formulation to dual min-cost-flow polyhedra. Coflow polyhedra provide a main part of a proof of an old conjecture of Gallai about a lower bound on the size of a largest stable set in a graph.

From the abstract of "CoFlow Polyhedra" [1990]: A linear system L is called box TDI if L together with any inequalities $b \le x \le a$ is totally dual integral (TDI).

(TDI of an L implies the primal integrality of vertices when all the numbers of L are integers.)
For any digraph G with node-set V(G), and for any fixed rational-valued $d = (d_v : v \text{ in } V(G))$, the following system of inequalities in variables $x = (x_v : v \text{ in } V(G))$ is box TDI:

For every directed circuit C of G, $\sum (x_v : v \text{ in } C) \le \sum(d_v : v \text{ in } C)$.



Thanks for reading. Jack

Faster Matroid Intersection

Deeparnab Chakrabarty, Yin Tat Lee, Aaron Sidford,
Sahil Singla, Sam Chiu-wai Wong

April 5, 2019

Abstract
In this paper we consider the classic matroid intersection problem: given two matroids $M_1 = (V, I_1)$ and $M_2 = (V, I_2)$ defined over a common ground set V, compute a set $S \in I_1 \cap I_2$ of largest possible cardinality, denoted by r. We consider this problem both in the setting where each M_i is accessed through an independence oracle, i.e. a routine which returns whether or not a set $S \in I_i$ in T_{ind} time, and the setting where each M_i is accessed through a rank oracle, i.e. a routine which returns the size of the largest independent subset of S in M_i in T_{rank} time. In each setting we provide faster exact and approximate algorithms. Given an independence oracle, we provide an exact $O(nr\log r \cdot T_{ind})$ time algorithm