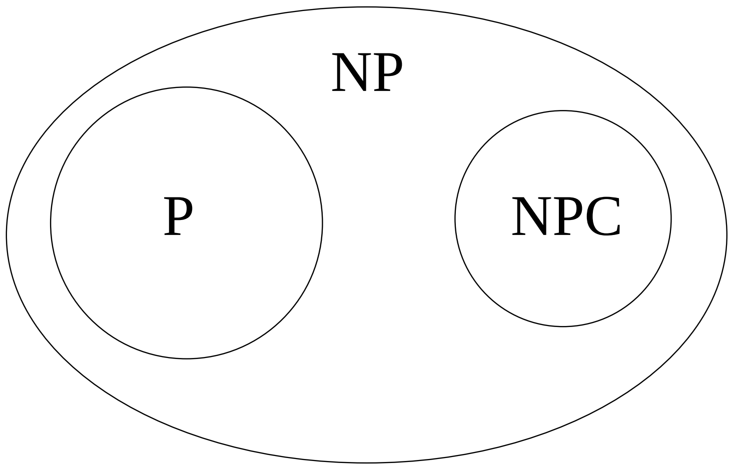
CSE 531 - NP Intermediate: The stuff between P and NP-complete.

Tim James
tjames@cs.washington.edu

1 Overview

The main focus of the class has been on problems that are members of NP-complete or are of P. This may leave you wondering what kinds of problems would be in that region of NP that doesn't include class P or NP-complete problems. Recall the picture of the class NP (assuming P $\neq$ NP):



This paper will discuss the area which consists of NP - (NPC $\cup$ P). This is commonly called NP-intermediate (NPI) because the problems in this class are of suspected to be of intermediate difficulty between P and NPC problems.

So far, problems that experts believe to be in NPI have not been proven to be in NPI. Rather, they tend to have certain characterstics that make them differ from NPC problems and they have not been shown to have a reduction from an NPC problem.

We'll use several potential members of NPI as examples for this paper. We'll also extract text liberally from [1], which you should read if you find this interesting.

2 NPI Background Information

Here, we will state a general theorem proved by Ladner in [5] which we can use to show that NPI is not empty.

Theorem 1 Suppose we have a recursive language B such that $B \notin$ P. There exists a polynomial time recognizable language $D \in$ P such that the language $A = D \cap B$ does not belong to P, $A \propto B$, and yet it is not the case that $B \propto A$. ($\propto$ signifies "is polynomial time mapping reducible to")

We can apply Theorem 1 for our needs as follows: Let B be any NP-complete language. If P $\neq$ NP, then $B \notin$ P, so the hypothesis of the theorem holds. Moreover, the resulting language A belongs to NP because $D \in$ P and $B \in$ NP. The theorem then implies that $B \not\propto A$, so A cannot be NP-complete, and because $A \notin$ P, it follows that $A \in$ NPI.

For example, suppose B were the NP-complete HAMILTONIAN CIRCUIT problem. The theorem states that there exists a polynomial time recognizable class of graphs such that HAMILTONIAN CIRCUIT, when restricted to that class of graphs, is neither NP-complete nor in P (assuming P $\neq$ NP).

The theorem tells us more about the structure of NPI. Assuming P $\neq$ NP, the class NPI must itself be made up of an infinite collection of distinct equivalence classes of languages, for if B is any problem in NPI, Theorem 1 would allow us to construct a new problem that is "easier" than B but still not in P. Ladner[5] also shows that, if P $\neq$ NP, then NPI must contain pairs of languages C and D such that *neither* $C \propto D$ *nor* $D \propto C$. Thus in the partial order according to "hardness" imposed on NP by the relation $\propto$, there must be incomparable elements if P $\neq$ NP.

Now, we could construct problems that are in NPI, but this turns out to be very difficult, so we'll focus on the problems that were discussed by Karp[3] (GRAPH ISOMORPHISM, COMPOSITE NUMBERS, LINEAR PROGRAMMING) as well as some others.

3 NPI Problems

These are problems taken from [3] which suggests them as candidates for NPI.

GRAPH ISOMORPHISM is the problem of determining if two graphs are structurally identical. More formally, given graphs $G = (V, E)$ and $G' = (V, E')$, is there a one-to-one function $f : V \to V$ such that $\{u, v\} \in E$ iff $\{f(u), f(v)\} \in E'$?

COMPOSITE NUMBERS is the problem of determining whether a number is composite. More formally, given a positive integer K, does there exist a pair of integers $m, n > 1$ such that $K = mn$.

LINEAR PROGRAMMING is the problem of determining whether a set of linear inequalities has a rational valued solution. (Usually the goal is to maximize an objective function.) More formally, given integer vectors $V_i = (v_i[1], v_i[2], \ldots, v_i[n])$, $1 \leq i \leq m$, $D = (d_1, d_2, \ldots, d_m)$, $C = (c_1, c_2, \ldots, c_n)$, and an integer B, is there a rational vector $X = (x_1, x_2, \ldots, x_n)$ such that $V_i \cdot X \leq d_i$ for $1 \leq i \leq m$ and such that $C \cdot X \geq B$?

Starting with GRAPH ISOMORPHISM, researchers have noticed that its nature is much more constrained than its NPC counterpart, SUBGRAPH ISOMORPHISM (the problem of finding if, in a source graph, there exists a subgraph which is isomorphic to another target graph). Notice that in SUBGRAPH ISOMORPHISM, you can add and remove edges at will from the source graph without affecting your answer. Contrariliy, in GRAPH ISOMORPHISM, if your source graph is an isomorphism of your target graph, any change in the source graph must be reflected in the target graph. Of course, there is no proof that it is NPI, but no reduction from an NP-Complete problem has been found nor has any polynomial time algorithm been found.

Mathon [6] made another observation that figuring out whether two graphs are isomorphic is polynomially equivalent to the problem of calculating the number of isomorphisms between the two graphs. There is no known NPC problem where checking the existence of a solution and counting the solutions is polynomially equivalent, so this furthers our belief that GRAPH ISOMORPHISM could be in NPI.

Note that the reduction from checking for isomorphisms to counting isomorphisms is trivial since the result of counting, if greater than 0, will imply there is an isomorphism. The reduction from counting isomorphisms to checking for isomorphisms is not so trivial and is left for the reader to pursue.

COMPOSITE NUMBERS and LINEAR PROGRAMMING were both originally suspected to be in NPI, but under different, and possibly more sound reasoning. Recall that co-NP = $\{\Pi^c : \Pi \in$NP$\}$ (Π^c is read "pi complement"). On the grounds that many problems in co-NP do not seem to be in NP, we might guess that NP $\neq$ co-NP. This guess is even stronger than the guess that P $\neq$ NP. The class P is closed under complementation (that is, P = co-P), so NP $\neq$ co-NP would imply P $\neq$ NP, although it might be the case that P $\neq$ NP even though NP = co-NP. This leads us to the following theorem.

Theorem 2 If there exists an NP-complete problem Π such that $\Pi^c \in$ NP, then NP = co-NP.

Any problem in NP can be reduced to a problem in NP-complete. Once this is done, if the complement is in NP, then every item in NP has a complement in NP, thus NP = co-NP.

According to Theorem 2, a problem Π for which both Π and Π^c belong to NP cannot be NP-complete unless NP = co-NP. If we go by the assumption that NP $\neq$ co-NP, the previous statement brings about a contradiction.

Now, if we look at COMPOSITE NUMBERS, we know that it is in NP because we simply need to verify that, given a factor, that factor divides the number. The complementary problem is PRIMES. This has been shown to be verifiable in polynomial time by [7]. Note now that if both COMPOSITE NUMBERS and PRIMES are in NP, neither can be NP-complete unles NP = co-NP.

The fundamental duality theorem of linear programming can be used to show that the complementary problem of LINEAR PROGRAMMING is itself a variant of linear programming that belongs to NP. This put LINEAR PROGRAMMING in the same category as COMPOSITE NUMBERS until a polynomial time algorithm for solving LINEAR PROGRAMMING was discovered by Khachian[4] and improved by Karmarkar[2]. This makes it clear that there is still a definite possibility to finding solutions to other problems that are candidates for NPI.

References

[1] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman, 1979.

[2] N. Karmarkar. A new polynomial time algorithm for linear programming. *Combinatorica*, 4:373-395, 1984.

[3] R. M. Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, 85-103, Plenum Press, 1972.

[4] L. Khachian. A polynomial algorithm in linear programming. *Soviet Math. Dokl.*, 20:191-194, 1979.

[5] R. E. Ladner. On the structure of polynomial time reducibility. *Journal of the ACM*, 22(1):155-171, January 1975.

[6] R. Mathon. A note on the graph isomorphism counting problem, *Information Processing Letters*, 8(3), 131-132, 1979.

[7] V. R. Pratt. Every Prime has a Succinct Certificate. *SIAM Journal on Computing*, 4(3):214-220, 1975.