The borderline between P and NP

Wim H. Hesselink, February 12, 2001

Abstract

A famous problem in theoretical computer science is the question whether P $\neq$ NP? The inequality would mean that certain computational problems (those that belong to the class NP) cannot be solved efficiently. It was recently announced on the internet that A.D. Plotnikov had proved P = NP. We sketch the problem and Plotnikov's approach to the problem.

1 Introduction to NP

Let us start with an example of a so-called NP-complete problem.

An undirected graph is a pair (V, E) where V is a set of vertices and E is a symmetric, antireflexive binary relation of V. A subset K of V is called a clique iff $(x, y) \in E$ for all pairs $x, y \in K$ with $x \neq y$. The problem *CLIQUE* is to decide, given such an undirected graph (V, E) and a number k, whether the graph has a clique of size $\#K = k$.

It is easy to write a computer program to solve this problem. For example, in the functional language Haskell, we can use

```
existsClique 0 f xs = True   -- the empty set is a clique
existsClique k f [] = False  -- the empty set has no larger cliques
existsClique k f (x:xs) =
  existsClique (k-1) f neighbours || existsClique k f xs
  where neighbours = filter (f x) xs
```

Argument `xs` stands for the list of vertices. Function `f` is a boolean function of two arguments such that `(f x y)` is equivalent to $(\mathtt{x}, \mathtt{y}) \in E$. The construction of `f` as a random access function may require some preprocessing of the set E. The Haskell function `filter` makes the sublist of `xs` of the elements that satisfy a given criterion; in this case it makes the list of neighbours of `x`. For $k > 0$ and a nonempty list of vertices, the program tries to extend the first element to a clique or to find a clique in the remainder.

Let us now look at the time complexity of function `existsClique`. Execution of it may require a number of steps that is exponential in the length of list `xs`. The function is quite efficient, however, for special classes of instances. For example, for any constant B, its time complexity is quadratic in the length of `xs` if we restrict to the graphs with degree bounded by B, for then the first alternative of its definition can be computed in linear time. Also, its time complexity is polynomial (of degree $2B$) for instances with $k \leq B$. The function can be made somewhat more efficient by first testing whether $k \leq \#\mathtt{xs}$ and just yielding `False` otherwise. I think that there is no known algorithm for *CLIQUE* more efficient than that.

The characteristic property of problem *CLIQUE* is that it asks for the existence of something, a witness, which may be hard to find but, once proposed,

is easily accepted or rejected. The witness is usually found, as above, by an exhaustive search in a search space of a size exponential in the size of the parameters, which in this case are the graph and k.

The class NP consists of such problems. The class P consists of the problems that can be solved in polynomial time. More precisely,– for which there is an algorithm that solves the problem in a number of steps bounded by a polynomial in the size of the parameters. P is a subclass of NP.

A problem is called NP-complete iff every other problem in NP can be reduced to it (by a polynomial reduction). The class NP and the concept of NP-completeness were introduced by Cook [2] in 1971, but it is still not known whether the classes P and NP really differ.

In order to prove that P = NP, it suffices to give a polynomial solution for one NP-complete problem. There is a long list of such problems, see [5].

2 Cliques and anticliques, chains and antichains

First, some terminology. If T is a set of finite sets, an element $S \in T$ is said to be *largest* in T iff $\#S' \leq \#S$ for every $S' \in T$. It is said to be *smallest* in T iff $\#S \leq \#S'$ for every $S' \in T$. It is said to be *maximal* in T iff $S = S'$ for every $S' \in T$ with $S \subseteq S'$. It is said to be *minimal* in T iff $S = S'$ for every $S' \in T$ with $S' \subseteq S$.

Let $G = (V, E)$ be an undirected graph. Recall that a subset K of V is called a *clique* iff $(x, y) \in E$ for all x, $y \in K$ with $x \neq y$. A subset A of V is called an *anticlique* iff $(x, y) \notin E$ for all x, $y \in K$.

A maximal clique can be constructed in quadratic time:

```
maxClique f [] = []
maxClique f (x:xs) = x: maxClique f (filter (f x) xs)
```

The construction of a largest clique can be done in exponential time. The following Haskell function yields a pair consisting of a largest clique and its size:

```
largestClique f [] = ([], 0)
largestClique f (x:xs) =
  if k0 < k1 then (c1, k1) else (x:c0, 1+k0) where
    (c0, k0) = largestClique f (filter (f x) xs)
    (c1, k1) = largestClique f xs
```

Let $P = (V, \leq)$ be a partially ordered set (poset). A subset C of V is called a *chain* iff $x \leq y$ or $y \leq x$ for all x, $y \in C$. A subset A of V is called an *antichain* iff $\neg\,(x \leq y)$ for all x, $y \in A$.

A collection Q of subsets of V is called a *partition* of V iff they have union V and are nonempty and pairwise disjoint.

A smallest clique partition Q of a graph $G = (V, E)$ is a partition of V that consists of cliques of G with smallest size $\#Q$. Similarly, a smallest chain partition Q of a poset $P = (V, \leq)$ is a partition of V that consists of chains of P with smallest size $\#Q$.

(0) **Lemma.** (a) Let Q be a smallest clique partition of a graph G. Then every anticlique A of G has size $\#A \leq \#Q$.
(b) Let Q be a smallest chain partition of a poset P. Then every antichain A of P has size $\#A \leq \#Q$.

Proof. A clique and an anticlique (a chain and an antichain) can not have more than one element in common. Therefore, in either case, the following argument works. For every member C of Q, we have $\#(A \cap C) \leq 1$. Since A is contained in the union of Q, it follows that $\#A \leq \#Q$. □

Example. It is easy to give a graph in which $\#A < \#Q$ for all clique partitions Q and all anticliques A. Indeed, let G be the pentagon with five vertices and five undirected edges. Then every clique partition of G has at least three members, whereas every anticlique has at most two elements. □

The situation is different for chain partitions and antichains. Indeed, we have

(1) **Theorem** (Dilworth [4, 6, 8]). Let Q be a smallest chain partition of a poset P. Then every largest antichain A of P has size $\#A = \#Q$ and satisfies $\#(A \cap C) = 1$ for every member $C \in Q$.

From our point of view, another difference between graphs and posets is even more important. There is an algorithm to determine a smallest chain partition (or a largest antichain) of a given poset in polynomial time. The problem to determine the size of a smallest clique partition (or a largest anticlique) of a given graph is NP-complete.

Plotnikov's idea [7] is to use the polynomial algorithms for smallest chain partitions and largest antichains of posets to solve the problem to determine the size of a smallest clique partition of the graph.

3 Ordering the graph

Since we want to exploit the algorithms for posets to our graph $G = (V, E)$, we have to order the graph in some way. Let a relation F be called an *arrowing* iff $E = F \cup cv(F)$ and the directed graph (V, F) is acyclic. Here we use $cv(F)$ for the converse of F, i.e., set of pairs (x, y) with $(y, x) \in F$. The easiest way to construct such an arrowing is to introduce a linear order, say $\sqsubseteq$ on V and then define

$$F = \{(x, y) \,|\, (x, y) \in E \ \wedge \ x \sqsubseteq y\} \ .$$

Actually, every arrowing of E can be obtained in this way since a directed acyclic graph always allows a so-called topological ordering.

Let us fix such an arrowing F of E. Then the transitive closure F^* of F is a partial order on V. In this way, we get a poset (V, F^*).

It is now easy to see that every clique of graph (V, E) is a chain of poset (V, F^*). It follows that every clique partition of the graph is a chain partition

of the poset. This implies that the size of the smallest chain partition of the poset is a lower bound for the size of the smallest clique partition of the graph.

Since this holds for every arrowing of E, we may decide to focus on convenient arrowings. For any arrowing F, we define the set of initial vertices

$$\mathit{init}(F) = \{x \in V \mid (\forall\, v :: (v,x) \notin F)\} \ .$$

In other words, $\mathit{init}(F)$ consists of the minimal elements of the poset (V, F^*). The set $\mathit{init}(F)$ is always an antichain of (V, F^*). Let us define $\mathit{dil}(F)$ to be the size of a largest antichain of (V, F^*). Dilworth's theorem implies that it is also the size of a smallest chain partition.

We define the arrowing F to be *init-saturated* iff $\mathit{init}(F)$ is a largest antichain of (V, F^*), or equivalently iff $\#\mathit{init}(F) = \mathit{dil}(F)$.

Remark. This concept is weaker than the concept of vertex saturation proposed in [7], but it is the one that is used below. Note that every antichain of (V, F^*) is an anticlique of (V, E) but a largest antichain need not be a largest anticlique, not even a maximal one. □

The following algorithm can be used to construct an init-saturated arrowing F for any graph (V, E).

(2)
choose a linear order $\sqsubseteq$ of V ;
$F := \{(x,y) \mid (x,y) \in E \ \wedge \ x \sqsubseteq y\}$;
while $\#\mathit{init}(F) < \mathit{dil}(F)$ **do**
  construct an antichain A with $\#A = \mathit{dil}(F)$;
  modify $\sqsubseteq$ such that the elements of A are in front ;
  $F := \{(x,y) \mid (x,y) \in E \ \wedge \ x \sqsubseteq y\}$;
end .

The antichain A constructed is also an anticlique. It follows that the new set F satisfies $A \subseteq \mathit{init}(F)$. This implies that the body increases the size of $\mathit{init}(F)$. This size is bounded by $\#V$. Therefore the loop terminates and its body is executed less than $\#V$ times. Since the computations of the guard and of the body of the loop can be done in polynomial time, the algorithm is polynomial. Upon termination, we have $\mathit{init}(F) = \mathit{dil}(F)$, which implies that F is init-saturated.

Busygin [1] has extracted from [7] the observation that, for an init-saturated arrowing, the problem to decide whether some smallest chain partition of the poset is a clique partition of the graph is NP-complete. This is formalized and proved as follows.

The problem *BUS* has as argument a directed acyclic graph (V, F) such that F is init-saturated. It asks whether some smallest chain partition of the poset (V, F^*) is a clique partition of the graph (V, E) where $E = F \cup \mathit{cv}(F)$.

(3) **Theorem** (Busygin). Problem *BUS* is NP-complete.

Proof. It is easy to prove that *BUS* is in NP. The main point is to prove that an algorithm for *BUS* can be used to determine the size of a smallest clique partition of an arbitrary graph (V, E). This is shown as follows. Use algorithm (2) to determine an init-saturated arrowing F of E. If the answer to *BUS* is positive, the size of a smallest clique partition is $\#init(F)$, otherwise it is larger than $\#init(F)$.

Following Busygin, we define the initial extension (V_1, F_1) of the digraph (V, F) to be obtained by adding one new vertex, say z, to V with new arrows to all vertices $v \in V \setminus init(F)$. We have $init(F_1) = \{z\} \cup init(F)$. It follows that, if Q is a smallest chain partition of (V, F^*), a smallest chain partition of (V_1, F_1^*) is obtained by adding the singleton chain $\{z\}$ to Q. This implies that (V_1, F_1) is init-saturated.

If the size of a smallest clique partition Q of (V, E) is larger than $\#init(F)$, one of the members of Q does not meet $init(F)$ and can therefore be extended with z to a clique in the extended graph. Conversely, every clique of the extended graph intersects the original graph in a clique. It follows that initial extension does not change the size of a smallest clique partition as long as that size is larger than $init(F)$.

Since $\#init(F)$ is incremented with 1 by initial extension, repeated initial extension establishes that the size of a smallest clique partition equals $\#init(F)$. The algorithm for *BUS* is used to determine how long to proceed. So, the size of the smallest clique partition of (V, E) is determined by

> determine an init-saturated arrowing F of E ;
> **while** $\neg\, BUS(V, F)$ **do** extend (V, F) **end** ;
> **return** $\#init(F)$.

The number of steps of the loop is bounded by the size of a smallest clique partition, which is bounded by $\#V$. The algorithm has therefore polynomial complexity. □

Remark. Yet, Plotnikov [7] gives no polynomial algorithm for *BUS*. □

4 Bipartite matching

The intuition is that chains are easier than cliques since one can traverse them in linear order and compare consecutive elements. Such a comparison will be performed by "pairwise matching". This idea is formalized as follows.

A *relation* R between sets X and Y is a subset of the cartesian product $X \times Y$. Alternatively, such a relation is also called a bipartite graph. We define $U \circ R = \{y \mid (\exists\, u \in U :: (u, y) \in R)\}$. We write $cod.R = X \circ R$.

Relation R is called a *matching* iff, for every $x \in X$, there is at most one $y \in Y$ with $(x, y) \in R$ and, for every $y \in Y$, there is at most one $x \in X$ with $(x, y) \in R$. It is clear that every matching R between X and Y satisfies $\#R \leq \#X$ **min** $\#Y$.

The matching problem is to determine, given a relation R between sets X and Y, a largest matching $M \subseteq R$. There exist polynomial algorithms that solve the matching problem, see [3].

We apply the theory of bipartite matching to construct optimal chain partitions. Let $(X, \leq)$ be a finite partially ordered set. Let R be the binary relation $(<)$ on X given by

$$x < y \quad \equiv \quad x \leq y \ \wedge \ x \neq y \ .$$

We claim that every R–matching M induces a chain partition P of X with $\#M + \#P = \#X$, and that every chain partition can be obtained in this way.

First, let M be an R–matching. This means that $M \subseteq R$ and that every $z \in X$ admits at most one element x with $(x, z) \in M$ and at most one element y with $(z, y) \in M$. Let M^* be the reflexive transitive closure of M. Since M is a matching, we have

$$(x, y) \in M^* \quad \wedge \quad (x, z) \in M^* \quad \Rightarrow \quad (y, z) \in M^* \quad \vee \quad (z, y) \in M^* \ .$$

It follows that the relation $M^\#$ given by

$$(x, y) \in M^\# \quad \equiv \quad (x, y) \in M^* \quad \vee \quad (y, x) \in M^*$$

is the least equivalence relation that contains M.

Since $M \subseteq R$, it follows that the equivalence classes for $M^\#$ form a chain partition P of X. The minimal elements of the members of P are the elements of $X \setminus cod.M$. Since every member of P has precisely one minimal element and M is a matching, we have

$$\#P = \#X - \#(cod.M) = \#X - \#M \ .$$

Conversely, a chain partition P of X induces a matching M of R by

$$\begin{aligned}(x, y) \in M \quad \equiv \quad & (\exists\, U \in P :: x \in U \ \wedge \ y \in U \ \wedge \ (x, y) \in R \\ & \wedge \ (\forall\, z \in U : (x, z) \in R : (z, y) \notin R) \ .\end{aligned}$$

Moreover, P consists of the equivalence classes for $M^\#$.

This bijective corresponce between matchings and chain partitions induces a bijective correspondence between the smallest chain partitions of $(X, \leq)$ and the largest matchings in $R = (<)$. Therefore, an algorithm that determines a largest matching in R also determines a smallest chain partition of $(X, \leq)$.

Now, consider a graph (V, E), with an init-saturated arrowing F. Let M be a largest matching of $R = F^+$, the transitive closure of F. The corresponding chain partition of (V, F^*) consists of the equivalence classes for $M^\#$. These equivalence classes form cliques of (V, E) if and only if $M^+ \subseteq F$. This raises the problem to find a largest matching $M \subseteq F^+$ such that $M^+ \subseteq F$. In view of Theorem (3), this problem is NP-complete.

If M is a largest matching within F^+ and $M^+ \subseteq F$, then M is a largest matching within F. We can therefore split the above problem in two parts. Firstly, decide whether the largest matchings in F^+ have the same cardinalities

as the largest matchings in F. Secondly, decide whether some largest matching $M \subseteq F$ satisfies $M^+ \subseteq F$. The first problem can be solved by a polynomial algorithm. It remains to solve the second problem by a polynomial algorithm! It is not clear to me whether Plotnikov [7] claims to give such a solution.

Instead, let me show that the second problem is at least tractable by giving a algorithm for it. Assume that the largest matchings in F^+ have the same cardinality *lam* as the largest matchings in F.

The idea is to construct a largest matching $M \subseteq F$ with $M^+ \subseteq F$ by deleting edges from F to see whether they are needed in M. We thus make a recursive procedure *Tramat* with two parameters G and H, that always satisfy $G \subseteq H \subseteq F$, which returns whether there exists a matching M with $G \subseteq M \subseteq H$ and $M^+ \subseteq F$ and $\#M = lam$. The problem is then solves by a call $Tramat(\emptyset, F)$.

procedure $Tramat(G, H) : Boolean =$
  if not (G is a matching and $G^+ \subseteq F$)
  then return false end ;
  $H := H \cap ((V \setminus dom(G)) \times (V \setminus cod(G)))$;
  construct a largest matching $N \subseteq H$;
  if $\#G + \#N < lam$ **then return false end** ;
  if $(G \cup N)^+ \subseteq F$ **then return true end** ;
  choose edge $e \in N$;
  if $Tramat(G, H \setminus \{e\})$ **then return true**
  else return $Tramat(G \cup \{e\}, H)$ **end** ;
end *Tramat* .

In this algorithm $G \cup N$ is a proposed largest matching. If $(G \cup N)^+ \subseteq F$, it is a witness of a solution. Otherwise, the algorithm chooses an arbitrary edge of N, tries whether it can be missed from F and, if not, adds it to G as an obligatory edge.

Unfortunately (?), due to the two recursive calls in the final conditional statement, this algorithm seems to be exponential. The prelude before the final conditional, however, is polynomial. The paper [7] seems to implement this prelude more efficiently, but I don't see how it can avoid the final conditional.

5 Another point of Plotnikov

Plotnikov's paper [7] starts with an arrowing F of the graph (V, E) which has a stronger property that init-saturation. I don't understand the role of this stronger property, but it does solve the following problem:

Given graph (V, E), construct an arrowing F such that the largest matchings within F are also largest matchings in F^+.

The paper [7] gives an algorithm for this problem and claims that it is polynomial. I do understand (a variation of) this algorithm, but I don't see why it should be polynomial. When I try to find an upper bound, I reduce it to the following problem.

Definition. A partition of a natural number n is an infinite descending of natural numbers with sum n (and hence with infinitely many zeroes). The lexical order of partitions is defined by

$$p < q \quad \equiv \quad p \neq q \quad \wedge \quad p_k < q_k \text{ where } k \text{ is the least index with } p_k \neq q_k.$$

The reduction relation $\rightarrow$ between partitions of n is defined by

$$p \rightarrow q \quad \equiv \quad p < q \quad \wedge \quad (\forall\, k :: p_k \geq q_{k+1})\ .$$

My upper bound would be the length of the largest reduction chain for $\rightarrow$ where n is the number of vertices of V. Renardel has shown there is a reduction path of length 2^m if $n \geq (3m^2 + m)/2$. Unfortunately, this does not show that my algorithm is not polynomial, it only shows that my estimates are not strong enough! Anyway, for the moment, this is just a related problem of potential interest.

References

[1] Busygin, S.: `www.busygin.dp.ua/clipat.html`

[2] Cook, S.A.: The complexity of theorem proving procedures. *Proc. 3rd ACM Symp. on Theory of Computing.* ACM, New York, pp. 151–158, 1971.

[3] Cormen, T.H., Leiserson, C.E., Rivest, R.L.: Introduction to algorithms. MIT Press, 1990.

[4] Dilworth, R.P.: A decomposition theorem for partially ordered sets. Annals of Math. (2) **51** (1950) 161–166.

[5] Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to NP-completeness. W.H. Freeman & Co., San Fransisco, 1979.

[6] Van Lint, J.H., Wilson, R.M.: A course in combinatorics. Cambridge University Press 1992.

[7] Plotnikov, A.D.: An efficient algorithm for the smallest clique partition problem. See [1] and `www.slashdot.org`

[8] Tverberg, H.: On Dilworth's decomposition theorem for partially ordered sets. J. Combinatorial Theory **3** (1967) 305–306.