On the Complexity of Fixed Parameter Problems

(Extended Abstract)

Karl R. Abrahamson
Department of Computer Science
Washington State University
Pullman, WA 99164

John A. Ellis[1]
Department of Computer Science
University of Victoria
Victoria, B.C.
Canada V8W 2Y2

Michael R. Fellows[2]
Department of Computer Science
University of Idaho
Moscow, ID 83843

Manuel E. Mata
Department of Computer Science
University of Victoria
Victoria, B.C.
Canada V8W 2Y2

Abstract. We address the question of why some fixed-parameter problem families solvable in polynomial time seem to be harder than others with respect to *fixed parameter tractability*: whether there is a constant α such that all problems in the family are solvable in time $O(n^\alpha)$. We model the question by considering a class of polynomially indexed relations. Our main results show that (1) this setting supports notions of completeness that can be used to explain the apparent hardness of certain problems with respect to fixed parameter tractability, and (2) some natural problems are complete.

[1]This author's research is supported in part by the Natural Science and Engineering Research Council of Canada.
[2]This author's research is supported in part by the National Science Foundation under grant MIP-8603879, by the Office of Naval Research under contract N00014-88-K-0456, and by the National Aeronautics and Space Administration under engineering research center grant NAGW-1406.

1 Introduction

To illustrate the central questions of this paper, consider the following facts concerning three well-known problems, all of which are NP-complete when the parameter k is part of the input, and trivially solvable in polynomial time for each fixed value of k. In what follows, C is a constant, independent of k, and C_k depends on k.

Fact 1 ([1]): For every fixed k, there is an algorithm that decides in time $Cn + C_k$ whether a graph G of order n has a k-element vertex cover.

Fact 2 ([1]): For every k, there is an algorithm that decides in time $C_k n$ whether a graph G of order n has a k-element feedback vertex set.

Let $\{k\text{-LI}\}$ be the problem of determining whether a system of linear inequalities can be made consistent over the rational numbers by deleting at most k of the inequalities. The name is enclosed in brackets to emphasize that it is a family of problems, one problem for each k.

Fact 3: No algorithm is presently known to decide $\{k\text{-LI}\}$ in time $C_k n^\alpha$ for any fixed α (independent of k), where n is the size of the system.

The above are examples of parameterized problems, where k is the parameter. For many parameterized problems, there is a reasonably small range of the parameter of engineering significance. The examples illustrate the following observation.

Observation 1: The fact that a problem with input (I, k) is NP-complete implies *nothing* about the complexity of the fixed k versions of the problem.

Based only on its NP-completeness, we are unable to say whether $\{k\text{-LI}\}$ is solvable in time n^α for every fixed k, where α is independent of k, even assuming $P \neq NP$. This is true because the family of algorithms might not be polynomially uniform. Alternatively, a polynomially uniform family of algorithms could exist requiring time $C_k n^\alpha$, where C_k grows faster than any polynomial in k, and α is independent of k.

A k-indexed family of decision problems is *absolutely tractable* (*relatively tractable*) if there is are constants C and α (independent of k) such that, for each k, there is an algorithm that solves the k^{th} problem in the family in time $Cn^\alpha + C_k$ (time $C_k n^\alpha$) on inputs of size n.

Part of the motivation of this work is supplied by powerful tools based on well-partially ordered sets [7, 8, 9] that are well adapted to proving relative tractability results. Those, and other tools, have failed to apply to certain fixed parameter problem families. We would like a mechanism for demonstrating that some families of problems are not relatively tractable. As noted above, NP-completeness is not the tool. On the other hand, short of proving $P \neq NP$, we cannot hope to obtain more than supporting evidence, since, if $P = NP$, every parameterized problem in NP can be solved in fixed polynomial time in $n + |k|$. A standard tool for obtaining such evidence is completeness for a class of problems (e.g. [4, 5, 6]). We define a class IP, and a notion of completeness for IP that provides the needed mechanism.

2 Summary of Results

Although precise definitions will be given in later sections, we summarize our results here. Our main results concern completeness and hardness for the class IP.

Theorem 1: If a problem family Π is complete or hard for IP, and if Π is relatively tractable, then every problem in IP is relatively tractable.

Theorem 2: IP has complete problems.

Theorem 3: $\{k\text{-LI}\}$ is hard for IP.

Two other problem families are also shown to be hard for IP. Hence, if either of them is relatively tractable, then every problem in IP is relatively tractable. The problems are as follows.

$\{k$-SHORT SAT$\}$

Input: Propositional formula ϕ in CNF, with n variables.

Question: Is there an assignment to the first $k \lceil \log n \rceil$ variables that causes ϕ to *unravel*? A single step of unraveling a formula is setting a literal ℓ of clause c to true (and its negation to false), provided all other literals in that clause are already assigned false. The formula unravels if a sequence of unraveling steps leads to all clauses holding a literal that has been assigned the value true. This is closely related to proof by unit resolution.

$\{k$-SMALL WEIGHT DEGREE 3 SUBGRAPH$\}$

Input: Undirected graph G with each vertex assigned a nonnegative integer weight.

Question: Is there an induced subgraph H of G of weight at most k such that each vertex has degree at least 3? The weight of H is the sum of the weights of its vertices. The problem remains hard for IP even when all weights are either 0 or 1.

All of the above IP-hard problem families have the property that they are NP-complete for variable k, and P-complete for every fixed k [2, 3]. It is not clear that such "dual completeness" is necessary, but the reductions that we actually perform are particularly simple, and those simple reductions do require dual completeness.

3 Indexed families of problems

Let Σ be an alphabet and $\Pi \subseteq \Sigma^* \times \Sigma^*$ be a relation. The n^{th} *slice* of Π is the relation $\Pi_n = \{(x,y) \in \Pi : |x| = n\}$. An *index* for Π_n is an enumeration of the range of Π_n. An index for Π is a family of indices $(i_n)_{n\geq 1}$ for the slices Π_n of Π. An *indexed relation* is a pair $(\Pi, (i_n)_{n\geq 1})$, although we will suppress the index when it is understood from context.

Intuitively, an index for Π suggests a brute force algorithm for testing membership of x in the domain of Π, namely, try each potential witness in the range of $\Pi_{|x|}$. In order for the search to be economical, certain conditions must be met. IP is the class of indexed relations $(\Pi, (i_n)_{n\geq 1})$ that are

1. *P-bounded*: There is a polynomial p such that if $(x,y) \in \Pi$ then $|y| < p(|x|)$.
2. *P-checkable*: There is a polynomial time algorithm to decide if $(x,y) \in \Pi$.
3. *P-indexed*: There are polynomial time algorithms to compute the functions
   (a) $(1^n, i) \mapsto y$ such that $i_n(y) = i$,
   (b) $(1^n, y) \mapsto i_n(y)$ for $y \in \text{range}(\Pi_n)$.

An example of an indexed relation in IP is one naturally associated with the vertex cover problem. The vertices are given canonical names, and an order n graph uses vertices $v_1, \ldots, v_n$. The potential witnesses for order n graphs are subsets of $\{v_1, \ldots, v_n\}$, enumerated in increasing order of size and lexicographically among subsets of the same size. A potential witness is a genuine witness if it is a vertex cover for the graph. We can let $VC = \{(G,S) : S$ is a vertex cover of $G\}$.

We would like to identify relatively and absolutely tractable problem families in IP. Let q be a polynomial with integer coefficients. The *q-bounded-search* problem $q\Pi$ for indexed relation Π with index $(i_n)_{n\geq 1}$ is as follows.

Input: $x \in \Sigma^*$ and natural number $j \leq q(|x|)$.

Question: Is there a y such that $(x,y) \in \Pi$ and $i_{|x|}(y) \leq j$?

For example, the problem of deciding whether graph G has a vertex cover of size at most k corresponds to the input (G, J) to the q-bounded-search problem for $\Pi = VC$, for $j = j(n) = \Sigma_{i=0}^{k} \binom{n}{i}$, and $q(n)$ a polynomial that exceeds $j(n)$.

Canonically associated with indexed relation Π is the family of all decision problems $\{q\Pi\}_{q \in \mathcal{Z}[n]}$.

4 Many-to-one reductions and completeness

Definition 1: If Π and Π' are indexed relations in IP, then a *many:1 reduction* from Π to Π' is a function $f: \Sigma^* \times \mathcal{N} \rightarrow \Sigma^* \times \mathcal{N}$ that maps $(x,i) \mapsto (x',i')$, such that

1. f is computable in polynomial time in $|x|$ and $\log i$,
2. there are polynomials r, s and t such that $|x| \leq r(|x'|)$ and $|x'| \leq s(|x|)$ and $i' \leq t(i)$,
3. $(\exists y\, (x,y) \in \Pi$ and $i_{|x|}(y) \leq i)$ if and only if $(\exists y'\, (x',y') \in \Pi'$ and $i'_{|x'|}(y') \leq i')$.

The following lemma is straightforward to prove.

Lemma 1: Suppose $\Pi, \Pi' \in IP$ and there is a many:1 reduction from Π to Π'. If the family $\{q\Pi'\}$ is absolutely (relatively) tractable then the family $\{q\Pi\}$ is absolutely (relatively) tractable.

Let GENERATED SAT be the indexed relation $\{(x, y) : x$ is a Boolean expression in 3-CNF and y is a truth value assignment to an initial segment of the variables of x that causes x to unravel$\}$. The indexing is by length of the initial segment, and lexicographically within segments of a given length. Formulas are assumed to be written in a canonical form where a formula with n variables uses variables $v_1, \ldots, v_n$.

A proof of the following theorem is sketched in the appendix.

Theorem 4: GENERATED SAT is complete for *IP* under many:1 reductions.

5 Turing reductions and hard problem families

Although indexed relations are convenient as a formal model of families solvable by brute force search, natural problem families are not presented as indexed relations. We deal with natural problem families by using Turing reductions, which apply both between such families and between natural families and indexed relations. The family of decision problems $\{q\Pi\}$ can be effectively represented by a family $(L_0, L_1, \ldots)$, where L_k is the problem $q\Pi$ for $q(n) = n^k$.

Definition 2: If $F = (L_0, L_1, \ldots)$ and $F' = (L'_0, L'_1 \ldots)$ are families of languages indexed by $\mathcal{N}$, then a *polynomial time Turing reduction* from F to F' is a polynomial time oracle machine M that, on input (x, i), determines whether $x \in L_i$. When (Q, k) is written on the query tape, the information supplied by the oracle is whether $Q \in L'_k$. The operation of M must satisfy: $\forall i \in \mathcal{N}$ there is a finite set of indices $I \subseteq \mathcal{N}$ such that, for every x, on input (x, i), M makes queries only of the form (Q, k) for $k \in I$. We write $F \leq_T F'$ when such a reduction exists.

The following lemma is straightforward.

Lemma 2: If there is a polynomial time Turing reduction from F to F' and if F' is relatively tractable, then F is relatively tractable.

The following Turing reductions are known to exist.

Theorem 5:

(i) $\{q$-GENERATED SAT$\} \leq_T \{k$-SHORT SAT$\}$.

(ii) $\{k$-SHORT SAT$\} \leq_T \{k$-LI$\}$.

(iii) $\{k$-SHORT SAT$\} \leq_T \{k$-SMALL WEIGHT DEGREE 3 SUBGRAPH$\}$.

In this extended abstract, we provide a proof sketch of part (ii) only. It suffices to reduce $(k-1)$-SHORT SAT to k-LI, for each k.

Let ϕ be an instance of $(k-1)$-SHORT SAT with n variables, and let $m = (k-1)\lceil \log n \rceil$. The first step is to simulate the process of choosing the values of the first m variables. Let $x_1, \ldots, x_n$ be rational valued variables, and, for $i = 1, \ldots, n$, write inequalities

$$x_i \leq 0 \qquad (1)$$

$$x_1 + \cdots + x_n \geq k \qquad (2)$$

$$0 \leq x_i \leq 1 \qquad (3)$$

Include $k+1$ copies of each inequality (2) and (3). The system is clearly inconsistent, but can be made consistent by deleting any k of the inequalities (1), with the corresponding variables taking on value 1. Moreover, there is no other way to make the system consistent by deleting only k inequalities. Hence, choosing the k inequalities to delete is equivalent to choosing a size k subset T of $\{x_1, \ldots, x_n\}$.

Order the k-subsets of $\{x_1, \ldots, x_n\}$ in lexicographic order (based on an increasing list of the members). There is a straightforward algorithm *Rank* that maps a k-subset s into its rank in the list, in time polynomial in n and k.

Cook (see [3]) shows how to simulate logic gates using linear inequalities. Using his simulation, we can simulate any polynomial time computation using polynomially many inequalities. By including $k+1$ copies of each inequality, we avoid the possibility of any of them being deleted.

We simulate algorithm Rank. The output is a list of variables whose values will have to be the binary representation of the rank of T. Choose the least significant m bits of the rank to represent the first m variables of ϕ. We require that $m < \lfloor \log \binom{n}{k} \rfloor$, which holds for sufficiently large n. Unraveling can be checked by a polynomial time algorithm, so it can also be simulated with polynomially many inequalities. Again including each inequality $k+1$ times, we end up with a system S that is a yes instance for k-LI if and only if ϕ is a yes instance for k-SHORT SAT. System S can also be created in polynomial time in n and k, from ϕ.

6 Open Questions

For the indexed relations DOMINATING SET and INDEPENDENT SET (where the indexing is just the same as in our example of VERTEX COVER), the following Turing reductions can be shown, based essentially on self-reduction strategies.

(i) $\{q$-DOMINATING SET$\}$ $\leq_T \{k$-DOMINATING SET$\}$.

(ii) $\{q$-INDEPENDENT SET$\}$ $\leq_T \{k$-INDEPENDENT SET$\}$.

These problem families are not known to be relatively tractable, but are not known to be complete for *IP*. Since k-DOMINATING SET and k-INDEPENDENT SET are each in *NC* for fixed k, and hence are probably not *P*-complete, any reduction used to show these problems complete would have to be inherently sequential.

References

[1] S. R. Buss, M. R. Fellows, and B. Monien. Linear-time algorithms for some well-known fixed parameter problems. 1989. Manuscript, Department of Computer Science, University of Idaho.

[2] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-completeness.* W. H. Freeman, San Francisco, 1979.

[3] H. J. Hoover and W. L. Ruzzo. *A Compendium of Problems Complete for P.* Technical Report, Department of Computer Science, University of Washington, 1985.

[4] M. Krentel. The complexity of optimization problems. In *18th ACM Symposium on Theory of Computing*, pages 69–76, 1986.

[5] C. Papadimitriou and D. Wolfe. The complexity of facets resolved. In *26th IEEE Symposium on Foundations of Computer Science*, pages 74–78, 1985.

[6] C. Papadimitriou and Y. Yannikakis. Optimization, approximation and complexity classes. In *20th ACM Symposium on Theory of Computing*, pages 229–234, 1988.

[7] N. Robertson and P. D. Seymour. Disjoint paths, a survey. *SIAM J. Algebraic Discrete Methods*, 6:300–305, 1985.

[8] N. Robertson and P. D. Seymour. Graph minors, a survey. In I. Anderson, editor, *Surveys in Combinatorics*, Cambridge University Press, Cambridge, England, 1985.

[9] N. Robertson and P. D. Seymour. Graph minors XV. Wagner's conjecture. To appear.

Appendix

Theorem 6: GENERATED SAT is complete for *IP* under many:1 reductions.

Proof sketch: Let $\Pi \in IP$. There are polynomials h, k and l so that

1. If $(x, y) \in \Pi$ then $|y| \leq h(|x|)$.
2. It can be determined in time $k(|x|)$ whether $(x, y) \in \Pi$.
3. In time $l(|x| + \log j)$, the potential witness y such that $i_{|x|}(y) = j$ can be produced.

Let $(j_n)_{n\geq 1}$ be the index for Π. Since witnesses for GENERATED SAT concern only an initial segment of the variables, the entire sequence of indices $(i_n)_{n\geq 1}$ for GENERATED SAT is captured by a single index function $i()$.

Given (x, i) with $\log i \leq h(|x|)$, which by (1) is all that is relevant, we show how to compute, in time polynomial in $|x|$, a Boolean expression E_x and an index i' so that the definition of many:1 polynomial-time reduction from Π to GENERATED SAT is satisfied. We take $i' = i$. Let $t = \lceil \log i \rceil$ be the length of a partial truth assignment with index i.

We first argue that we can restrict our attention to truth assignments to the first t variables. Clearly, an assignment to the first $t' < t$ variables that causes and expression to unravel can be extended to all of the first t variables, with the expression still unraveling. Truth assignments to more than t variables lie beyond the search cutoff.

In time polynomial in $|x|$ we can build Boolean circuits that compute the following functions:

f_1: $\{0,1\}^t \rightarrow \{0,1\}$ such that $f(\tau) = 1$ iff $i(\tau) \leq i$.

f_2: $\{0,1\}^t \rightarrow \{0,1\}^*$, such that $y = f(\tau)$ is the potential witness for Π with the property that $i(\tau) = j_{|x|}(y)$.

f_3: $\{0,1\}^* \times \{0,1\}^* \rightarrow \{0,1\}$ such that $f(x, y) = 1$ iff $(x, y) \in \Pi$, for x and y encoded in binary.

E_x consists of clauses representing the gates of the above circuits, together with some auxiliary clauses that tie together the computations of f_1, f_2 and f_3. We may assume that the logic gates of the circuit have at most two inputs and two outputs. Gates are represented as follows. Consider the case of an *and* gate with inputs a, b, and outputs c, d. Corresponding to this gate we have the following clauses in E_x: $(\bar{a} + \bar{b} + c)$, $(\bar{a} + \bar{b} + d)$, $(a + \bar{c})$, $(a + \bar{d})$, $(b + \bar{c})$ and $(b + \bar{d})$. For an *or* gate, we have the following clauses: $(a + b + \bar{c})$, $(a + b + \bar{d})$, $(\bar{a} + c)$, $(\bar{a} + d)$, $(\bar{b} + c)$, and $(\bar{b} + d)$. The essential feature of these clause sets is that, if the input variables a and b have already been given values, then the unraveling process forces the output variables c and d to be assigned appropriate values.

Let $v_1, \ldots, v_t$ be new variables, which will be the first t variables of E_x. These are the inputs to the circuits for f_1 and f_2. Let $u_1, \ldots, u_m$ be the Boolean variables representing the output of the circuit that computes f_2. These, together with Boolean variables $x_1, \ldots, x_n$ representing x are the inputs to the circuit computing f_3. We include in E_x single-literal clauses of the appropriately negated or unnegated variables $x_1, \ldots, x_n$ in order to represent x. We also include the single literal clauses (z) and (w), where z is a Boolean variable representing the output of the circuit computing f_1 and w is a Boolean variable representing the output of the circuit computing f_3.

The resulting expression E_x is clearly polynomially related to x in size. It remains only to check that it functions as required with respect to unraveling. The verification of this is straightforward. □