Complexity with Rod

Lance Fortnow

Georgia Institute of Technology

Abstract. Rod Downey and I have had a fruitful relationship though direct and indirect collaboration. I explore two research directions, the limitations of distillation and instance compression, and whether or not we can create NP-incomplete problems without punching holes in NP-complete problems.

1 Introduction

I first met Rod Downey at the first Dagstuhl seminar on "Structure and Complexity Theory" in February 1992. Even though we heralded from different communities, me as a computer scientist working on computational complexity, and Rod as a mathematician working primarily in computability, our paths have crossed many times on many continents, from Germany to Chicago, from Singapore to Honolulu. While we only have had one joint publication [1], we have profoundly affected each other's research careers.

In 2000 I made my first pilgrimage to New Zealand, to the amazingly beautiful town of Kaikoura on the South Island. Rod Downey had invited me to give three lectures [2] in the summer New Zealand Mathematics Research Institute graduate seminar on Kolmogorov complexity, the algorithmic study of information and randomness. Those lectures helped get Rod and Denis Hirschfeldt interested in Kolmogorov complexity, and their interest spread to much of the computability community. Which led to a US National Science Foundation Foundation Focused Research Group grant among several US researchers in the area including myself. What comes around goes around. In 2010 Rod Downey and Denis Hirschfeldt published an 855 page book *Algorithmic Randomness and Complexity* [3] on this line of research.

In this short paper I recount two other research directions developed with interactions with Rod Downey. In Section 2, I recall how an email from Rod led to a paper with Rahul Santhanam on instance compression [4], easily my most important paper of this century. In Section 3, I discuss my joint paper with Rod on the limitations of Ladner's theorem, that if P is different than NP, there are NP-incomplete sets and that continues to affect my current research.

2 Distillation

In the 1992 Dagstuhl workshop, Rod Downey gave a lecture entitled "A Completeness Theory for Parameterized Intractability," my first taste of fixed-parameter tractability (FPT). FPT looks at NP problems with a parameter, like whether a given graph G with n vertices has a vertex cover of size k. A problem is fixed-parameter tractable if there is an algorithm whose running time is of the form $f(k)n^c$ for an arbitrary f that does not depend on n. Vertex cover has such an algorithm but clique does not seem to. Downey and Michael Fellows had developed a series of complexity classes to capture these questions. I learned much more about FPT in a series of lectures of Michael Fellows at the aforementioned Kaikoura workshop in 2000.

On March 11, 2007 I travelled to Toronto to visit Rahul Santhanam, a former student and then a postdoc at the University of Toronto. On March 12th Rod Downey sent me a question by email (with a lucky typo) that came from a paper "On problems without polynomial kernels" [5] that Downey was working on with Fellows, Hans Bodlaender and Danny Hermelin. This confluence of events would lead my paper with Rahul Santhanam "Infeasibility of Instance Compression and Succinct PCPs for NP" [4]. These two papers would go on to be the co-winners of the 2014 EATCS-IPEC Nerode Prize and would eventually have over 500 combined citations.

Here is a formatted version of the email sent by Rod.

> Say a language L has a distillation algorithm if there is an algorithm A which when applied to a sequence (perhaps exponentially long) $x_1, \ldots, x_n$ outputs in time polynomial in $\sum_i |x_i|$ a single string t such that
>
> 1. t is small: $|t| \leq \max\{x_i : i \leq n\}$, and
> 2. there exists an i, $x_i \in L$ iff $t \in L$
>
> Clearly either all or no NP complete problems have distillation algorithms.
>
> Conjecture: No NP complete L has a distillation algorithm.
>
> Can you think of any classical consequence of the failure of this conjecture? I had thought it implied $\mathrm{NP}^{\mathrm{NP}} \in$ NP/poly, but the proof was flawed.

I discussed the problem with Rahul and responded.

> I believe I can show you get co-NP in NP/poly (thus PH in Σ_3^p) under this condition.
>
> Fix a length m. Let S be the set of all strings not in L of length at most m. We will get a subset V of S with $|V|$ of size at most poly(m) and an $r \leq \text{poly}(m)$ such that for all x in S there are $y_1, \ldots, y_r$ with
>
> 1. $x = y_i$ for some i.
> 2. the procedure on $y_1, \ldots, y_r$ outputs a t in V. Then we have an NP test for x in S with V as the advice.
>
> Let $N = |S|$. There are N^r tuples $y_1, \ldots, y_r$. On each of them the procedure maps to something in S. For some z in S at least N^{r-1} tuples map to z. The number of x's covered by z is at least $N^{(r-1)/r}$ covering a $N^{-1/r}$ fraction of the elements of S. Picking $r = \log N$ (which is poly(m)) makes this a constant fraction. Then we recurse on the remaining strings in S.

Rod Downey conferred with Michael Fellows and the next day realized he had slightly misstated the problem.

> Sorry I realized that I made a mistake in the way that I defined distillation.
>
> t is small means that $|t|$ is polynomial in $\max\{|x_i| : i \leq n\}$, not $\leq \max\{|x_i| : i \leq n\}$. I knew how to do it for $\leq |x_i|$ since then the language would be weakly p-selective (or something similar) and hence $\mathrm{PH} = \Sigma_2^p$.
>
> This is the problem. I cannot see how to fix your proof either, since the recursion goes awry.

Rahul and I looked it over and I responded to Rod

> I worked this out with Rahul Santhanam. The same basic argument does go through if you pick r at least $|t|$.

Rod expressed surprise and when I got back to Chicago I wrote up a quick proof that would become the main lemma in my paper with Rahul [4]. A month later I cleaned up the statement and proof and present that version below (Lemma 1).

We generalized the proof for Rod's first question to get the proof for the question Rod had meant to ask. This two step approach helped us dramatically. If Rod didn't have the typo in the first question, we may never have discovered the proof. Just goes to show the role of pure luck in research.

Lemma 1. *Let L be any language. Suppose there exists a polynomial-time computable function f such that $f(\phi_1, \ldots, \phi_m) = y$ with*
1. *Each $|\phi_i| \leq n$.*
2. *$|y| \leq n^c$ with c independent of m.*
3. *y is in L if and only if there is an i such that ϕ_i is in* SAT*.*

then NP is in co-NP/poly.

Proof Let $A \subseteq \overline{\mathrm{SAT}} \cap \Sigma^{\leq n}$ with $A \neq \emptyset$. Let $B = \overline{L} \cap \Sigma^{\leq n^c}$. Let $N = |A|$ and $M = |B| \leq 2^{n^c}$. Let $m = n^c$.

Claim. There must be some y in B such that for at least half of the ϕ in A, there exists $\phi_1, \ldots, \phi_m$ such that
1. For some i, $\phi = \phi_i$.
2. $f(\phi_1, \ldots, \phi_m) = y$.

Let ϕ be y-good if the above holds. Given y, we have a NP-proof that ϕ is not satisfiable for all y-good ϕ.

Now consider the N^m tuples $(\phi_1, \ldots, \phi_m)$ in A^m. The function f maps these tuples into elements of B. So for some y in B must have $\frac{N^m}{M}$ inverses in A^m.

If there are k y-good ϕ then $\frac{N^m}{M} \leq k^m$, so $k \geq \frac{N}{M^{1/m}}$. Since $m = n^c$, $M^{1/m} \leq 2$ and $k \geq \frac{N}{2}$ which proves our claim.

Now we start with $A = \overline{\mathrm{SAT}} \cap \Sigma^{\leq n}$ and $S = \emptyset$. Applying the claim gives us a y in B. We put y in S, remove all of the y-good ϕ from A and repeat. Since $|A| \leq 2^{n+1}$ we only need to recurse at most $n + 2$ times before A becomes empty.

We then have the following NP algorithm for $\overline{\mathrm{SAT}}$ on input ϕ using advice S:

- Guess $\phi_1, \ldots, \phi_m$.
- If $\phi = \phi_i$ for some i and $f(\phi_1, \ldots, \phi_m)$ is in S then accept.

Every language that is fixed-parameter tractable language can be mapped in polynomial time to an input whose size is a function of the parameter. Vertex cover, for example, has a stronger property that the kernel of an input is polynomial in the size of the parameter, i.e., the size of the vertex cover to check. The paper of Bodlaender, Downey, Fellows and Hermelin [5] would use Lemma 1 to show a number of fixed-parameter tractable problems do not have short kernelizations without complexity consequences.

The paper of Rahul and myself [4] had made some connections also to a paper by Danny Harnik and Moni Naor [6] on instance compression with some connections to cryptography. Later Harry Buhrman and John Hitchcock [7] would build on our lemma to show that NP can't have subexponential-sized complete sets unless the polynomial-time hierarchy collapsed. Andrew Drucker [8] generalized the lemma to problems like AND-SAT (for all i instead of there exists an i in condition 3 of Lemma 1) and to probabilistic and quantum reductions.

3 Punching Holes in SAT

In a 1944 address to the American Mathematical Society, Emil Post [9] laid out the landscape of recursive and recursively enumerable languages (now commonly called computable and computably enumerable), as well as reductions between languages.

> A primary problem in the theory of recursively enumerable sets is the problem of determining the degrees of unsolvability of the unsolvable decision problems thereof. We shall early see that for such problems there is certainly a highest degree of unsolvability. Our whole development largely centers on the single question of whether there is, among these problems, a lower degree of unsolvability than that, or whether they are all of the same degree of unsolvability.

In the paper, Post laid out his program to tackle that question but ultimately leaves it unresolved. It would take a dozen years for Friedberg and Muchnik (see [10]) to show the existence of a computably enumerable set that was not computable and not all other computably enumerable sets reduce to it.

After Steve Cook [11] and Richard Karp [12] defined the complexity classes NP and NP-complete in the early 70s, one could ask a similar question: Is there a problem in NP that is not computable in polynomial-time and not complete?

Unlike in the computability world, we had several natural candidates for those classes including graph-isomorphism and factoring, where factoring as a language problem is the set of tuples (m, r) such that there is a prime factor p of m with $p \leq r$. It took just a couple of years after the introduction of NP-completeness for Richard Ladner [13] in 1975 to answer the question in the affirmative under the assumption that P differs from NP.

Ladner's proof works by "blowing holes in satisfiability". Ladner creates a language that for some input lengths is empty and other input lengths is some NP-complete problem like Boolean formula satisfiability. The lengths are chosen to diagonalize both from every polynomial-time algorithm and every reduction from satisfiability, thus ensuring that the language is not in P or NP-complete. To get the language in NP, Ladner develops a delayed diagonalization technique that doesn't move to the next requirement until it has had time to check that the previous requirement is fulfilled. We present Ladner's full proof as well as an alternate proof in our paper [1]. Both proofs leave large gaps in satisfiability.

I personally find Ladner's proof quite unsatisfying. We don't expect the natural candidates to behave like Ladner's set, as hard as satisfiability on some input lengths and computable in polynomial time on others. Rather every in every length we expect, for example, factoring to be difficult to compute but not complex enough for satisfiability to reduce to it. Is this a necessary factor to prove an intermediate set?

I visited Rod Downey in 1995 during his sabbatical year at Cornell. We discovered our shared concern about Ladner's proof. We formalized the issue by creating a definition of uniformly hard, basically that a set that is hard over polynomially long ranges of the input lengths.

Definition 2. *A language A is* uniformly hard *if for every language B computable in polynomial-time there is a k such that for every integer $n > 1$, A and B differ on some input of length between n and n^k.*

To justify uniformly hard we define an honest reduction with a slight variation to allow for giving a direct answer.

Definition 3. *An* honest reduction *from A to B is a polynomial-time computable function mapping Σ^* to $\Sigma^* \cup \{+, -\}$ such that*

1. *For some integer k, for all $n > 1$ and for all x, either $f(x) \in \{+, -\}$ or $|x| \geq |f(x)|^k$ where $|x|$ is the length of the string x.*
2. *If x is in A then $f(x) \in B \cup \{+\}$.*
3. *If x is not in A then $f(x) \in \overline{B} \cup \{-\}$, where $\overline{B} = \Sigma^* - B$.*

Uniformly hard sets are upwardly closed under these honest reductions.

Downey and I [1] looked at the question: If NP has uniformly-hard sets, is there an incomplete-set that is uniformly-hard under honest reductions? We conjecture such sets exist and in particular factoring should be such an example. However no proof exists that shows there are incomplete uniformly-hard sets. Why is proving such a result so difficult?

To answer that question, Downey and I look at a stronger version of Ladner's Theorem, with essentially the same proof, that there is no computable polynomial-time minimal degree.

Theorem 4. *For every computable B not in P there is a set A such that*

1. *A is not in P*
2. *A polynomial-time honestly reduces to B, in fact the reduction $f(x) \in \{x, -\}$ for all x, and*
3. *B does not even polynomial-time Turing reduce to A.*

Downey and I show that there could be a minimum uniformly-hard set if every problem computable in a polynomial amount of memory can also be computed in a polynomial amount of time.

Theorem 5. *If $P = PSPACE$, there is a computable minimum uniformly-hard set under polynomial-time honest reductions.*

We don't believe P = PSPACE but on the other hand complexity theorists have no approach to separating P and PSPACE. In particular that means we have no known way to avoid the large gaps given by Ladner's proof of Theorem 4.

Years later, Rahul Santhanam and I [14] generalized the uniformly hardness notion into a concept we called robustly-often which led to a new proof of the nondeterministic-time hierarchy. That work led to a paper by Rahul and myself [15] in the 2016 Computational Complexity Conference that gave new lower bounds for non-uniform classes, in particular we showed, for any a, b with $1 \leq a < b$, NTIME(n^b) is not contained in NTIME(n^a) with $n^{1/b}$ bits of advice.

All of these results show that the ideas that Rod Downey helps generate have ripples that continue to push my research today.

Acknowledgments

I'd like to thank the anonymous referee for several helpful comments.

References

1. Downey, R., Fortnow, L.: Uniformly hard languages. Theoretical Computer Science **298**(2) (2003) 303 – 315
2. Fortnow, L.: Kolmogorov complexity. In Downey, R., Hirschfeldt, D., eds.: Aspects of Complexity, Minicourses in Algorithmics, Complexity, and Computational Algebra, NZMRI Mathematics Summer Meeting, Kaikoura, New Zealand, January 7–15, 2000. Volume 4 of de Gruyter Series in Logic and Its Applications. de Gruyter (2001)
3. Downey, R., Hirschfeldt, D.: Algorithmic randomness and complexity. Springer Science & Business Media (2010)
4. Fortnow, L., Santhanam, R.: Infeasibility of instance compression and succinct PCPs for NP. J. Comput. Syst. Sci. **77**(1) (January 2011) 91–106

5. Bodlaender, H., Downey, R., Fellows, M., Hermelin, D.: On problems without polynomial kernels. Journal of Computer and System Sciences **75**(8) (2009) 423–434
6. Harnik, D., Naor, M.: On the compressibility of NP instances and cryptographic applications. SIAM Journal on Computing **39**(5) (2010) 1667–1713
7. Buhrman, H., Hitchcock, J.: NP-hard sets are exponentially dense unless coNP is contained in NP/poly. In: Computational Complexity, 2008. CCC '08. 23rd Annual IEEE Conference on. IEEE, New York (June 2008) 1 –7
8. Drucker, A.: New limits to classical and quantum instance compression. SIAM Journal on Computing **44**(5) (2015) 1443–1479
9. Post, E.: Recursively enumerable sets of positive integers and their decision problems. Bulletin of the American Mathematical Society **50**(5) (1944) 284–316
10. Soare, R.: Recursively Enumerable Sets and Degrees. Springer, Berlin (1987)
11. Cook, S.: The complexity of theorem-proving procedures. In: Proceedings of the 3rd ACM Symposium on the Theory of Computing. ACM, New York (1971) 151–158
12. Karp, R.: Reducibility among combinatorial problems. In Miller, R., Thatcher, J., eds.: Complexity of Computer Computations. Plenum Press (1972) 85–103
13. Ladner, R.: On the structure of polynomial time reducibility. Journal of the ACM **22** (1975) 155–171
14. Fortnow, L., Santhanam, R. In: Robust Simulations and Significant Separations. Springer Berlin Heidelberg, Berlin, Heidelberg (2011) 569–580
15. Fortnow, L., Santhanam, R.: New Non-Uniform Lower Bounds for Uniform Classes. In Raz, R., ed.: 31st Conference on Computational Complexity (CCC 2016). Volume 50 of Leibniz International Proceedings in Informatics (LIPIcs). Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany (2016) 19:1–19:14