The Thirty-Second AAAI Conference
on Artificial Intelligence (AAAI-18)

Computational Social Choice and Computational Complexity: BFFs?*

Lane A. Hemaspaandra
Department of Computer Science
University of Rochester
Rochester, NY 14627, USA

Abstract

We discuss the connection between computational social choice (comsoc) and computational complexity. We stress the work so far on, and urge continued focus on, two less-recognized aspects of this connection. Firstly, this is very much a two-way street: Everyone knows complexity classification is used in comsoc, but we also highlight benefits to complexity that have arisen from its use in comsoc. Secondly, more subtle, less-known complexity tools often can be very productively used in comsoc.

1 Introduction

Even in a work context, friendship is amazingly powerful. We all know how valuable it is to have collaborators who have each other's backs: working closely toward a shared project goal yet each bringing their own skills and perspective to the project, to seek to reach insights and advances that neither researcher could have achieved alone.

Research areas can be like that too. Computational social choice (henceforward, comsoc)—which brings a computational-cost lens to such social-choice issues as preference aggregation/elections/manipulation/fair division—certainly has roots tracing far back; but comsoc's existence as a recognized, distinctive research area within AI/multiagent-systems is quite recent. Despite that, its growth as an area during these recent years has been explosive, and we suggest that a synergy between comsoc and complexity has been partially responsible for that growth. At the 2017 AAMAS conference, for example, there were four sessions devoted to Computational Social Choice; no other topic had that many sessions. Those four sessions included 21 papers, and of those 21, 7 had the word "complexity" in their titles although in the entire conference only two other papers had the word complexity in their titles. In contrast, at the 2003 AAMAS conference the string "social choice" does not even appear in the ACM Digital Library online table of contents; neither does the string "election" or any form of the word "vote," and only three papers in the entire conference have the word "complexity" in their titles.

One thing that those numbers are reflecting is the fact that comsoc and complexity have worked together very well. Comsoc research, from the seminal work of Bartholdi, Orlin, Tovey, and Trick (1989b; 1989a; 1991; 1992) onward, has done an enviably skilled job of interestingly employing complexity classification—which admittedly has strengths and weaknesses, both in its theory and in its application—to reveal a subtle and varied landscape as to what is difficult and what is hard, and how small changes in problems and models can induce seismic changes in complexity.

Yet this article's primary goal is not to praise that praiseworthy achievement. It is to recognize two more subtle things that have already happened, but that we think need to be recognized for what they are, and what more can be done.

Firstly, the comsoc/computational complexity friendship is truly a two-way street. Although everyone knows that complexity classification is widely used in comsoc research, we will stress a direction that is not well-recognized within the AI community or the complexity community: Research in comsoc has often been of great benefit to complexity theory. In particular, complexity classification within comsoc has populated key complexity classes with problems that are undeniably *natural* and compelling. Why is that important? Complexity theorists like creating complexity classes that capture computational capabilities/resources that seem natural and compelling to the complexity theorists. But if those classes then turn out, for example, to have no natural complete problems, it is like hosting a party and having no one show up. It could even make one wonder whether the classes really were important at all. So complexity owes quite a debt to comsoc, for bringing parties to life.

In addition to stressing the two-way street aspect of the comsoc/computational complexity friendship, we will also point out how rather subtle, lesser-known complexity notions and machinery have in the comsoc world found fertile ground to actually do important things. Complexity notions ranging from search-versus-decision issues to one-to-one reductions to the join operator have arguably been used in the comsoc world in more natural and more satisfying ways than they have been used within complexity theory itself.

We present four different issues where the work in comsoc exemplifies one or both of these themes: comsoc work helping complexity, and lesser-known complexity notions

*A version with some additional details/citations appears as (Hemaspaandra 2017).

and machinery being used to address comsoc issues. We will particularly stress examples that fall in the area of the complexity of elections/voting, probably the most intense current focus of comsoc and a remarkably challenging, interesting, nuanced area (for a general survey of comsoc see the excellent articles by Chevaleyre et al., 2007 and Brandt, Conitzer, and Endriss, 2013, and for surveys of elections/voting within comsoc see the surveys by Faliszewski et al., 2009, Faliszewski, Hemaspaandra, and Hemaspaandra, 2010, and Faliszewski and Procaccia, 2010). Due to space/focus limitations, this article can cite just a few examples of these themes, but we try to discuss each of those with some context and care, and in two places our examples will connect complexity with other important AI areas, planning and SAT solvers, once with the comsoc/complexity connection growing from the other AI area and once with it creating an opportunity within the other AI area.

Then in the conclusion we will suggest that the two themes we have been stressing and the work to date on them make clear that comsoc experts and complexity experts working more closely and extendedly together will be to the benefit of both fields.

2 Make It a Party: Populating Lonely Complexity Classes

As mentioned above, complexity classes are typically defined to capture a certain mode/power/model of computing. For example, the important probabilistic complexity class BPP, bounded probabilistic polynomial time, is the class of those sets A for which there is a probabilistic polynomial-time Turing machine that on each input agrees with A with probability at least 75% (or 51%, or 99%; all three of those turn out to define the same class). The class NP is the class of those sets accepted by nondeterministic polynomial-time Turing machines. The class UP, unambiguous polynomial time, is the class of those sets A accepted by nondeterministic polynomial-time Turing machines whose acceptance is always unambiguous; for inputs not in A the machine has no accepting computation paths, and for each input in A, the machine has *exactly one* accepting computation path (in the lingo, "has a unique succinct certificate").

This section will present a few examples of classes that were not wildly filled with natural sets that seem to capture their nature, yet comsoc provided new, or even the first, such examples. By "capture the nature" of a class, in the dream case we would mean "is complete for the class," i.e., our set is in the class and has the property that each set in the class many-one reduces to our set. But if that can't be achieved, finding a set that falls into a given class and does not seem to fall into any smaller class is the natural fallback step to take.

Even readers not very familiar with complexity theory may know that there are thousands of known NP-complete problems and indeed a whole book is devoted to such problems (Garey and Johnson 1979); and that there is an entire book devoted even to problems complete for P, with respect to the natural completeness type there (Greenlaw, Hoover, and Ruzzo 1995); and that even for the levels of the polynomial hierarchy beyond NP and coNP, such as $\mathrm{NP}^{\mathrm{NP}}$, $\mathrm{coNP}^{\mathrm{NP}}$, $\mathrm{NP}^{\mathrm{NP}^{\mathrm{NP}}}$, and $\mathrm{coNP}^{\mathrm{NP}^{\mathrm{NP}}}$, there exists a detailed compendium (Schaefer and Umans 2002). And from that it would be easy to conclude that it is boring and easy to find natural problems complete for virtually any natural complexity class.

But that seems not to be so. For example, $\mathrm{NP} \cap \mathrm{coNP}$ (the class of sets that have succinct, easily-checkable proofs both of membership and nonmembership) and two of the classes mentioned above, BPP and UP, are not known to have even one complete set—not a natural one, and not even an unnatural one ((Sipser 1982; Hartmanis and Hemachandra 1988), see also (Hemaspaandra, Jain, and Vereshchagin 1993)). In fact, the just-cited papers and the work of Regan (1989) in effect are showing that the existence of a complete set for one of these classes would have sweeping consequences for our understanding of the class: that the issue of whether each of these classes has a complete set is in fact a disguised version of the question of whether there exists a nice enumeration of machines that precisely covers the class, i.e., that provides what Regan calls a "constructive programming system." The above papers even provide black boxes (aka oracles) relative to which these class contain no complete sets.

So it is not at all a sure thing that important classes—and BPP is undeniably important, since some would even argue that BPP rather than P should be the working definition of feasibility—have complete sets at all, much less natural complete sets.

And yet some classes that had remained largely or wholly unpopulated by natural sets needing the class's power *have* had natural examples provided by problems from comsoc, and other less empty but far from crowded classes have also turned out to be what pinpoints the complexity of important comsoc problems.

2.1 Θ_2^p: Parallel Access to NP

Probably the most striking such example is the Θ_2^p level of the polynomial hierarchy. This class was introduced by Papadimitriou and Zachos (1983), and is the class of sets that can be accepted by polynomial-time machines allowed to make $\mathcal{O}(\log n)$ sequential queries to an NP oracle; $\mathrm{NP} \cup \mathrm{coNP} \subseteq \Theta_2^p \subseteq \mathrm{P}^{\mathrm{NP}}$. Perhaps more naturally, Θ_2^p is known to be the class of sets that can be accepted by polynomial-time machines allowed to make (an unlimited number of) nonadaptive (i.e., parallel) queries to an NP oracle (Hemachandra 1989). For example, consider the language, ParitySAT, that accepts its input exactly if the input is a list of Boolean formulas and the number of them that is satisfiable is itself an odd number. ParitySAT is clearly in Θ_2^p, since the polynomial-time machine can in parallel ask its oracle about the membership in SAT of each of its input formulas, and then can see whether the number that are satisfiable is odd. We will come back in a moment to the unnaturalness of this example and the search for natural examples.

Θ_2^p is a very important class in complexity theory. Briefly put, it has a large number of equivalent characterizations that are quite natural, e.g., it is the class of sets that can be accepted by logspace machines allowed to make an unlimited number of adaptive queries to an NP oracle (Wag-

ner 1990); it is the class the polynomial hierarchy naturally is shown to collapse to if there are sparse (i.e., having at most a polynomial number of strings at each length) NP-Turing-complete sets (Kadin 1989); and its relationship to P^{NP} is known to completely characterize whether conversations with NP oracles can manufacture time-bounded randomness (Hemachandra and Wechsung 1991).

This at first sounds like quite a party, if one focuses on the results Θ_2^p is central to. But wait. Even when all these results were known, the class was not known to have even one natural complete set. It was known to have a large number of rather unnatural complete sets all having to do with counting the parity of items, e.g., the ParitySAT example given above about telling whether out of a collection of Boolean formulas an odd number of them are satisfiable is known to be complete for Θ_2^p (Wagner 1987).

So in fact, as to known natural complete sets, the party was at that time a (complexity-theoretically important but) empty room. However, comsoc put first one and then many people into that room. In particular, Charles Lutwidge Dodgson (aka Lewis Carroll) in the year 1876 had defined a fascinating election system (Dodgson 1876). He was motivated by the idea that it would be nice for an election system to ensure that (a) if (relative to the votes, which are assumed to each be a tie-free total ordering of the candidates) there is a candidate—what is known as a Condorcet winner—who is preferred in head-on-head contests against each other candidate, then that candidate is chosen the winner, and (b) otherwise, each candidate who is "closest" to being such a winner (in the sense that the number of adjacent exchanges in preference orders needed to make it become a Condorcet winner is the lowest among the candidates) is named a winner. And Dodgson's system then does just that: It counts distance from being a Condorcet winner, and the candidate(s) with the lowest distance wins. In doing that, Dodgson was in the 1800s already using the notation that in modern computer science is central and known as an edit distance.

Although Dodgson's system is mathematically very well defined, one of the seminal papers of comsoc showed that the computational complexity of implementing his system is not low. In particular, Bartholdi, Tovey, and Trick (1989b) showed that it is NP-hard to tell if a given candidate wins such an election. They left open whether the problem is NP-complete, but eight years later the problem was proven to be complete for Θ_2^p (Hemaspaandra, Hemaspaandra, and Rothe 1997a). Θ_2^p-complete problems cannot be NP-complete unless the polynomial hierarchy collapses to $\mathrm{NP} \cap \mathrm{coNP}$, and thus the winner problem for Dodgson elections is highly unlikely to be NP-complete.

But the most important thing to note here is that the complexity of winner-testing for Dodgson elections is *not* a problem that was rigged to provide a Θ_2^p-complete set. The election system was defined in the 1800s for its own interest and importance, long before NP was ever dreamed of, much less Θ_2^p. This problem—from comsoc—thus provides an extremely natural Θ_2^p-complete set.

This result sparked interest in whether other problems in comsoc and beyond might also be Θ_2^p-complete. And the floodgates opened. Other important election systems such as those of Kemeny and Young were proven to also be Θ_2^p-complete (Hemaspaandra, Spakowski, and Vogel 2005; Rothe, Spakowski, and Vogel 2003). General tools were extracted from this approach to try to make it easier to prove such results (Spakowski and Vogel 2000), Θ_2^p-completeness was shown to capture the complexity of how well certain greedy algorithms do (Hemaspaandra and Rothe 1998), and a survey was written looking at the meaning of improving from NP-hardness results to Θ_2^p-completeness results (Hemaspaandra, Hemaspaandra, and Rothe 1997b). Even in the quite different field of automata theory, Θ_2^p-complete problems were found to capture important, natural notions (Holzer and McKenzie 2000). Briefly put, the party was very much on!

2.2 $\mathrm{NP}^{\mathrm{PP}}$ and Other Classes

There are many other classes that comsoc and other work within AI have helped populate. One of the most striking examples is the class $\mathrm{NP}^{\mathrm{PP}}$, i.e., the sets solvable by nondeterministic polynomial-time Turing machines given access to (their choice of a) set from PP (probabilistic polynomial time, the class of sets for which there is a probabilistic polynomial-time Turing machine that on each input is correct with probability greater than 50%). $\mathrm{NP}^{\mathrm{PP}}$ is a class of great complexity, and it is not always easy to work with. In terms of descriptive generality and its "location," Toda's (1991) Theorem says that P^{PP} contains not just the polynomial hierarchy (PH) but even $\mathrm{PP}^{\mathrm{PH}}$. So $\mathrm{NP}^{\mathrm{PP}}$ contains both of those and even $\mathrm{NP}^{\mathrm{PP}^{\mathrm{PH}}}$, while being itself contained in PSPACE.

It was only in the late 1990s that people started finding natural complete sets for $\mathrm{NP}^{\mathrm{PP}}$, and that work came not from complexity issues but from an AI domain, the study of planning and decision processes. There are a number of papers on this and we won't here cite them all, but we mention as a pointer the very impressive work of Goldsmith, Littman, and Mundhenk (1998), which shows versions of both the plan-existence problem and the plan-evaluation problem are, for partially ordered plans, $\mathrm{NP}^{\mathrm{PP}}$-complete, and of Mundhenk et al. (2000) on Markov decision processes, which also yields natural $\mathrm{NP}^{\mathrm{PP}}$-complete problems.

Comsoc also has helped populate that party. Although currently its $\mathrm{NP}^{\mathrm{PP}}$ results are upper bounds—and we commend to the reader the open issue of either proving completeness or of lowering the upper bound—the beautiful work of Mattei, Goldsmith, Klapper, and Mundhenk (2015) shows that $\mathrm{NP}^{\mathrm{PP}}$ plays an important role in the study of bribery and manipulation in tournaments of uncertain information, by showing that many such problems are in $\mathrm{NP}^{\mathrm{PP}}$.[1]

[1] After doing that, that paper comments that "we have actually shown these problems are in $\mathrm{NP}^{\mathrm{PP}[1]}$," which is the class in which the NP machine is allowed at most one query per path to the PP oracle. Simply so that comment in the paper doesn't lead any reader of the present paper to think that $\mathrm{NP}^{\mathrm{PP}}$-completeness for that paper's problems is unlikely due to that paper having achieved the seemingly better upper bound of $\mathrm{NP}^{\mathrm{PP}[1]}$, we claim that in fact $\mathrm{NP}^{\mathrm{PP}[1]} = \mathrm{NP}^{\mathrm{PP}}$, and include a proof in our technical report ver-

There are many other parties, some admittedly not so previously poorly attended as those thrown by Θ_2^p and $\mathrm{NP}^{\mathrm{PP}}$, where comsoc has provided attendees. Here are just a few examples. Nguyen et al. (2014) gave the class DP (all sets that are the symmetric difference of two NP sets) a wide range of new complete sets having to do with social welfare optimization in multiagent resource allocation. The first $\mathrm{P}^{\mathrm{NP}[1]}$-completeness result in social choice (which also is perhaps the first natural completeness result anywhere for $\mathrm{P}^{\mathrm{NP}[1]}$) and the first P^{NP}-completeness result in social choice both come from a study of the complexity of weighted coalitional manipulation of elections, under the voting system, Veto, where each voter vetos one candidate and whoever has the fewest vetos wins; in an online setting (what in political science is called a roll-call vote), the winner problem here is $\mathrm{P}^{\mathrm{NP}[1]}$-complete for 3 candidates and is P^{NP}-complete for 4 or more candidates (Hemaspaandra, Hemaspaandra, and Rothe 2014). Filos-Ratsikas and Goldberg (2017) have recently proven a natural cake-cutting-related problem to be complete for the lonely class PPA. And a huge number of papers in comsoc contribute natural problems—often related to manipulative attacks on elections—to other classes such as $\mathrm{NP}^{\mathrm{NP}}$, $\mathrm{coNP}^{\mathrm{NP}}$, and PSPACE. Also, many counting-based issues in comsoc have been shown to be complete for the counting analogue of NP, the class $\#\mathrm{P}$.

In summary, there is a two-way street of friendship between comsoc and computational complexity. Complexity benefits since its classes—some of which played central roles in key abstract results yet had few or no natural complete problems—are given quite compelling natural problems based in comsoc. Comsoc in turn has benefited since the machinery of complexity-class classification helps clarify how hard or easy many of its problems are.

3 Complexity Machinery and Notions Find Fertile Ground in Comsoc

This section provides a tour of two cases where complexity techniques and notions have found very fertile ground in comsoc research. Although one-to-one (Section 3.2) reductions have been used centrally in complexity theory, in particular in the 1970s, their new use in comsoc is both powerful and quite different than the use they had in complexity. And in another case (Section 3.1), comsoc has given new life to lovely complexity machinery that had been developed in 1976, and yet that even 36 years later had not found a single application in a real-world domain.

Due to space limitations, we will cover each topic but briefly, though we will try to give a sense of how each supports this section's thesis that comsoc has been a fertile ground for complexity techniques and notions, helping comsoc of course, but also in the case of little-used or narrowly used complexity techniques helping establish the value or breadth of the techniques.

sion (Hemaspaandra 2017).

3.1 Search versus Decision for NP Problems

Everyone knows that SAT is a decision problem, and indeed that the vast majority of complexity classes are defined in terms of decision problems, not search problems. This has been the case for so long that it is easy to forget *why* this is so.

The reason why complexity is largely focused on decision problems is because for almost all natural problems the search problem clearly polynomial-time Turing reduces (and often, for example for SAT, even "polynomial-time 2-disjunctive truth-table" reduces) to the problem's decision version. And so the two problems are inherently of the same complexity, give or take composition with a polynomial; it is impossible in these cases for decision to be easy yet for search to be hard. SAT is the classic case: Given a Boolean formula, $F(v_1, v_2, \ldots)$, it is satisfiable exactly if at least one of $F(\textsc{True}, v_2, \ldots)$ and $F(\textsc{False}, v_2, \ldots)$ is satisfiable, so given a decision procedure for SAT, one can ask those two questions to it, and thus find a good value for the first variable (if any exists), and then can redo the process, on the "reduced" formula, to similarly set the second variable and so on. By the end, using as a black-box a decision procedure for SAT, one has found a satisfying assignment in polynomial time, when one exists.

Are all natural NP problems so polite as to allow themselves to be fitted into this straitjacket of linked complexities? For decades there have been hints that the answer might be "no." Most powerfully, in 1976 Borodin and Demers (1976) built lovely, clever machinery that showed that if $\mathrm{P} \neq \mathrm{NP} \cap \mathrm{coNP}$, then there exists an infinite polynomial-time recognizable subset of SAT that has no polynomial-time search algorithm finding solutions for its members. In brief, one could quickly determine with certainty that its members were satisfiable, but one has no idea—it turns out not even a good setting for the first variable of the formula—of what a satisfying assignment would be. (The reason this claim is not precluded by the above discussion of self-reducibility is that the Borodin–Demers subset is so pathological that it has infinitely many members such that the formulas generated at some stage of the above self-reduction process are all not in the subset, and so the above process is rendered invalid on the subset.)

Stunning though that result was, the fact that it was achieving its power not on all of SAT (which is impossible), but rather on a somewhat pathological subset of SAT, kept the result and its lovely machinery from gaining traction; it remained only as the above 1976 technical report, and did not have a conference or a journal version. The work was likely—and in the first and second of these four examples the work is explicitly discussed—the inspiration behind some later related work, e.g., that for unambiguous computation the implication mentioned above becomes an "if and only if" characterization (Hartmanis and Hemachandra 1988), that under the extremely strong assumption that deterministic and nondeterministic double exponential time differ there exists an (artificial) language in NP for which search does not reduce to decision (Bellare and Goldwasser 1994), that if $\mathrm{P} \neq \mathrm{NP}$ then there exist NP-complete languages that are not self-reducible (Faliszewski and Ogihara

2010), and that relative to some oracle there is a search versus decision separation for exponential-time classes (Impagliazzo and Tardos 1989).

Nonetheless, the work found not a single application on natural problems for 36 years. And it was comsoc that finally provided that application. In particular, Hemaspaandra, Hemaspaandra, and Menton (2013) proved that of the standard types of attacks on elections, for about half it holds that if $\mathrm{P} \neq \mathrm{NP} \cap \mathrm{coNP}$ then there is an election system having a polynomial-time winner problem for which the decision problem for that type of attack is in P but the search problem cannot be solved in polynomial time. (For the other half of the attack types, that paper proved unconditionally that search *does* reduce to decision.) Note that the attack types were not created by that paper; they are the long-standard attack types. And so the problems themselves are not tricks, but are the natural, standard ones, and on these, search and decision are being separated under the given complexity-theoretic assumption, which is widely believed to be true.

There are four comments which pretty compellingly need to be made at this point. First, if $\mathrm{P} = \mathrm{NP} \cap \mathrm{coNP}$, then integer factoring can be done in polynomial time, and so RSA and much of the foundation of modern computer security falls. So the above results say that either the foundations of cryptography are deeply flawed, or the approach taken within comsoc to defining election-attack problems—namely, they are defined in their decision versions—is defining as tractable some problems whose search version (the issue of finding the attack that works) is not tractable. Second, the reason this separation of search and decision is possible is that, quite unexpectedly, about half of the natural election-attack problems do not seem to be nicely self-reducible. Third, natural is in the eye of the beholder; although the election-attack problems used in the result are the standard, natural ones, the election systems used are indirectly specified by the hypothesized sets in $(\mathrm{NP} \cap \mathrm{coNP}) - \mathrm{P}$, and so are likely not very natural. And fourth and finally, this entire approach then found application in a different area of AI, namely, the study of backbones and backdoors of Boolean formulas, where it was shown for example that if $\mathrm{P} \neq \mathrm{NP} \cap \mathrm{coNP}$, then search versus decision separations occur for backbones (Hemaspaandra and Narváez 2017a; 2017b).

3.2 Density-of-Hardness Transfer and One-to-One Reductions

When one says a problem is NP-complete, that doesn't prove that it is hard. It just proves that if anything in NP is hard, then that problem is hard. This is generally viewed as much better than having nothing to say about the difficulty of the problem.

Wouldn't it be lovely to have an analogous type of result for frequency of hardness? For example, in Section 3.1, we discussed work showing that if $\mathrm{P} \neq \mathrm{NP} \cap \mathrm{coNP}$, then search and decision separate for many key issues regarding manipulative attacks on elections and backbones of Boolean formulas. But maybe in those examples, for which we know decision is easy, the hardness of search may be a con: Perhaps search is only hard on a very, very, very small portion of the inputs, asymptotically? That is, perhaps it is a worst-case separation that doesn't hold in the typical case.

To try to argue against that possibility, it would be wonderful if we could argue that if *even one* problem A in $\mathrm{NP} \cap \mathrm{coNP}$ is hard with a certain frequency h (i.e., each polynomial-time heuristic is wrong on A's membership problem on $\Omega(h(n))$ strings up to length n) then every polynomial-time heuristic for our search problem fails with almost that same frequency, namely, for each polynomial-time heuristic there is an $\epsilon > 0$ such that on our given search problem it fails on $\Omega(h(n^\epsilon))$ of the strings up to length n).

This would say that if *any* set in $\mathrm{P} \neq \mathrm{NP} \cap \mathrm{coNP}$ is frequently hard (relative to heuristic attacks), then our search versus decision separations are not cons. After all, it is widely believed, most crucially in the cryptography community (due in part to the issue of factoring), that there are sets in $\mathrm{NP} \cap \mathrm{coNP}$ that *are* frequently hard with respect to polynomial-time heuristics. And if one believes that, then the hypothetical machinery discussed above would immediately convert that into a claim of the frequent hardness of the created search problems whose decision problems are easy.

In fact, machinery that can do this exists, although it was developed for a completely different purpose in the 1970s. So this is an example where comsoc—and the study of SAT solvers—is giving a fresh use to a complexity-theoretic notion.

In particular, every CS undergraduate learns why claims of NP-completeness are crucially tied to the tool of many-one polynomial-time reductions; it is how we establish them. In the 1970s, though, there was a sudden moment of focus in theoretical computer science on the notion of *one-to-one* (aka injective) polynomial-time reductions, i.e., polynomial-time reductions that have no collisions.

This moment of intense focus on one-to-one reductions came about to support one of the great complexity goals of the 1970s: to prove that all NP-complete sets were the same set in a transparent disguise, namely, that all NP-complete sets were polynomial-time isomorphic. This is known as the Isomorphism Conjecture. To this day it remains open. However, using one-to-one reductions, in the 1970s Berman and Hartmanis (1977) proved that all familiar NP-complete sets indeed were polynomial-time isomorphic, which was a huge revelation. However, due to challenges such as the potential existence of one-way functions, that 1970s result has not only never been expanded to (under the inherently required assumption that $\mathrm{P} \neq \mathrm{NP}$) include all NP-complete sets, but indeed there is evidence that there may well exist nonisomorphic NP-complete sets (Joseph and Young 1985; Kurtz, Mahaney, and Royer 1995), i.e., that the Isomorphism Conjecture may fail.

These days in complexity theory, many-one reductions remain the standard, and one-to-one reductions are not often discussed. However, one thing that one-to-one reductions do fiendishly well is preserve density. If one has a bunch of strings with a certain behavior, their image when pushed through the reduction cannot possibly jumble them on top of each other, because one-to-one reductions don't jumble anything on top of anything else. It is admittedly true that

such reductions can leave "holes" (though injective, they need not be surjective), and can polynomially stretch (and superpolynomially contract, though inherently not too frequently) lengths; and that in fact is what is behind the "ϵ" mentioned above. But one-to-one reductions by definition never have collisions.

Exploiting that, the "if anything in NP $\cap$ coNP is frequently hard relative to polynomial-time heuristics then our search problems are almost as frequently hard relative to polynomial-time heuristics as those" results mentioned above are achieved by ensuring that the entire proof structure in those papers can be made to yield polynomial-time one-to-one reductions. (And the additional related paper mentioned earlier, which adds in results on the case of back*doors* of Boolean formulas, and has some related results about backbones of Boolean formulas, in both cases under the weaker assumption P $\neq$ NP, also works by achieving polynomial-time one-to-one reductions.)

Thus, to summarize, one-to-one reductions are a tool that was briefly extremely important in complexity in the 1970s in a brilliant but even forty years later not yet successful attempt to show that there is in effect just one NP-complete set. But comsoc—and also, related to SAT solvers, the study of backbones and backdoors of Boolean formulas—has brought the notion of one-to-one reductions again front-and-center, providing an "if anything in the class is frequently hard then *this* is frequently hard" analogue of the "if anything in the class is hard then *this* is hard" argument that many-reductions have provided for almost half a century in the case of NP-complete sets.

This case makes very clear the two-way street that exists between complexity and comsoc (and also the study of SAT solvers). Complexity benefits in that one of its notions that was a bit covered in cobwebs turned out to yield important frequency-of-hardness results in comsoc, and also relatedly in the study of SAT solvers. And comsoc benefits in that it now has these results, which are quite powerful evidence that the hardness of search that these speak of is in fact not some con job that happens only extremely infrequently.

4 Conclusions

We have discussed just a few of the rich collection of interactions between comsoc and computational complexity. Due to space limitations, many others—from how the complexity-theoretic join operation allows proofs in comsoc of the impossibility of proving certain impossibility theorems (Hemaspaandra, Hemaspaandra, and Rothe 2009) to how work on online control gives an unexpected new quantifier-alternation characterization of the class coNP (Hemaspaandra, Hemaspaandra, and Rothe 2017) to the rich interactions with approximation (see as just one of many examples Faliszewski, Skowron, and Talmon, 2017) to the power of dichotomy results to the insights given by parameterized complexity (both these last are on view simultaneously, for example, in Dey, Misra, and Narahari, 2017) to much more—are not even touched on here.

What are the take-aways? The benefits of the friendship between comsoc and computational complexity are a two-way street; both comsoc and complexity have benefited. Classes from complexity have been populated, cobweb-covered techniques have been given surprising new life, and interesting results have been obtained in comsoc from the use of those classes and techniques. In light of this, we urge researchers in comsoc and complexity to reach out and work with each other more and more, for the benefit of both fields. And we hope that more Ph.D. programs will encourage and create young researchers who are trained in and simultaneously expert in *both* areas; that will allow advances far beyond even those that so far have come from this wonderful synergy between areas.

Let us hope that the areas become best friends forever.

Acknowledgments. Warm thanks to the reviewers, Aris Filos-Ratsikas, and Paul Goldberg for helpful comments.

References

Bartholdi, III, J., and Orlin, J. 1991. Single transferable vote resists strategic voting. *Social Choice and Welfare* 8(4):341–354.

Bartholdi, III, J.; Tovey, C.; and Trick, M. 1989a. The computational difficulty of manipulating an election. *Social Choice and Welfare* 6(3):227–241.

Bartholdi, III, J.; Tovey, C.; and Trick, M. 1989b. Voting schemes for which it can be difficult to tell who won the election. *Social Choice and Welfare* 6(2):157–165.

Bartholdi, III, J.; Tovey, C.; and Trick, M. 1992. How hard is it to control an election? *Mathematical and Computer Modeling* 16(8/9):27–40.

Bellare, M., and Goldwasser, S. 1994. The complexity of decision versus search. *SIAM Journal on Computing* 23(1):97–119.

Berman, L., and Hartmanis, J. 1977. On isomorphisms and density of NP and other complete sets. *SIAM Journal on Computing* 6(2):305–322.

Borodin, A., and Demers, A. 1976. Some comments on functional self-reducibility and the NP hierarchy. Technical Report TR 76-284, Dept. of Computer Science, Cornell University.

Brandt, F.; Conitzer, V.; and Endriss, U. 2013. Computational social choice. In Weiss, G., ed., *Multiagent Systems*. MIT Press, 2nd edition. 213–284.

Chevaleyre, Y.; Endriss, U.; Lang, J.; and Maudet, N. 2007. A short introduction to computational social choice. In *Proceedings of the 33rd International Conference on Current Trends in Theory and Practice of Computer Science*, 51–69. Springer-Verlag *Lecture Notes in Computer Science #4362*.

Dey, P.; Misra, N.; and Narahari, Y. 2017. Parameterized dichotomy of choosing committees based on approval votes in the presence of outliers. In *Proceedings of the 16th International Conference on Autonomous Agents and Multiagent Systems*, 42–50.

Dodgson, C. 1876. A method of taking votes on more than two issues. Pamphlet printed by the Clarendon Press, Oxford, and headed "not yet published".

Faliszewski, P., and Ogihara, M. 2010. On the autoreducibility of functions. *Theory of Computing Systems* 46(2):222–245.

Faliszewski, P., and Procaccia, A. 2010. AI's war on manipulation: Are we winning? *AI Magazine* 31(4):53–64.

Faliszewski, P.; Hemaspaandra, E.; Hemaspaandra, L.; and Rothe, J. 2009. A richer understanding of the complexity of election systems. In Ravi, S., and Shukla, S., eds., *Fundamental Problems in Computing*. Springer. 375–406.

Faliszewski, P.; Hemaspaandra, E.; and Hemaspaandra, L. 2010. Using complexity to protect elections. *Communications of the ACM* 53(11):74–82.

Faliszewski, P.; Skowron, P.; and Talmon, N. 2017. Bribery as a measure of candidate success: Complexity results for approval-based multiwinner rules. In *Proceedings of the 16th International Conference on Autonomous Agents and Multiagent Systems*, 6–14.

Filos-Ratsikas, A., and Goldberg, P. 2017. Consensus Halving is PPA-complete. Technical Report arXiv:1711.04503 [cs.CC], Computing Research Repository, arXiv.org/corr/.

Garey, M., and Johnson, D. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman.

Greenlaw, R.; Hoover, H.; and Ruzzo, W. 1995. *Limits to Parallel Computation: P-Completeness Theory*. Oxford University Press.

Hartmanis, J., and Hemachandra, L. 1988. Complexity classes without machines: On complete languages for UP. *Theoretical Computer Science* 58(1–3):129–142.

Hemachandra, L., and Wechsung, G. 1991. Kolmogorov characterizations of complexity classes. *Theoretical Computer Science* 83:313–322.

Hemachandra, L. 1989. The strong exponential hierarchy collapses. *Journal of Computer and System Sciences* 39(3):299–322.

Hemaspaandra, L., and Narváez, D. 2017a. The opacity of backbones. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence*, 3900–3906. AAAI Press.

Hemaspaandra, L., and Narváez, D. 2017b. The opacity of backbones and backdoors under a weak assumption. Technical Report arXiv:1706.04582 [cs.AI], Computing Research Repository, arXiv.org/corr/.

Hemaspaandra, E., and Rothe, J. 1998. Recognizing when greed can approximate maximum independent sets is complete for parallel access to NP. *Information Processing Letters* 65(3):151–156.

Hemaspaandra, E.; Hemaspaandra, L.; and Menton, C. 2013. Search versus decision for election manipulation problems. In *Proceedings of the 30th Annual Symposium on Theoretical Aspects of Computer Science*, 377–388.

Hemaspaandra, E.; Hemaspaandra, L.; and Rothe, J. 1997a. Exact analysis of Dodgson elections: Lewis Carroll's 1876 voting system is complete for parallel access to NP. *Journal of the ACM* 44(6):806–825.

Hemaspaandra, E.; Hemaspaandra, L.; and Rothe, J. 1997b. Raising NP lower bounds to parallel NP lower bounds. *SIGACT News* 28(2):2–13.

Hemaspaandra, E.; Hemaspaandra, L.; and Rothe, J. 2009. Hybrid elections broaden complexity-theoretic resistance to control. *Mathematical Logic Quarterly* 55(4):397–424.

Hemaspaandra, E.; Hemaspaandra, L.; and Rothe, J. 2014. The complexity of online manipulation of sequential elections. *Journal of Computer and System Sciences* 80(4):697–710.

Hemaspaandra, E.; Hemaspaandra, L.; and Rothe, J. 2017. Online voter control in sequential elections. *Autonomous Agents and Multi-Agent Systems* 31(5):1055–1076.

Hemaspaandra, L.; Jain, S.; and Vereshchagin, N. 1993. Banishing robust Turing completeness. *International Journal of Foundations of Computer Science* 4(3):245–265.

Hemaspaandra, E.; Spakowski, H.; and Vogel, J. 2005. The complexity of Kemeny elections. *Theoretical Computer Science* 349(3):382–391.

Hemaspaandra, L. 2017. Computational social choice and computational complexity: BFFs? Technical Report arXiv:1710.10753 [cs.MA], Computing Research Repository, arXiv.org/corr/.

Holzer, M., and McKenzie, P. 2000. Alternating and empty alternating auxiliary stack automata. In *Proc. of the 25th Intl. Symposium on Mathematical Foundations of Computer Science*, 415–425. Springer-Verlag *Lecture Notes in Computer Science #1893*.

Impagliazzo, R., and Tardos, G. 1989. Decision versus search problems in super-polynomial time. In *Proceedings of the 30th IEEE Symposium on Foundations of Computer Science*, 222–227. IEEE Computer Society Press.

Joseph, D., and Young, P. 1985. Some remarks on witness functions for non-polynomial and non-complete sets in NP. *Theoretical Computer Science* 39(2–3):225–237.

Kadin, J. 1989. $\mathrm{P}^{\mathrm{NP}[\log n]}$ and sparse Turing-complete sets for NP. *Journal of Computer and System Sciences* 39(3):282–298.

Kurtz, S.; Mahaney, S.; and Royer, J. 1995. The Isomorphism Conjecture fails relative to a random oracle. *Journal of the ACM* 42(2):401–420.

Littman, M.; Goldsmith, J.; and Mundhenk, M. 1998. The computational complexity of probabilistic planning. *Journal of Artificial Intelligence Research* 9:1–36.

Mattei, N.; Goldsmith, J.; Klapper, A.; and Mundhenk, M. 2015. On the complexity of bribery and manipulation in tournaments with uncertain information. *J. of Applied Logic* 13(4, Part 2):557–581.

Mundhenk, M.; Goldsmith, J.; Lusena, C.; and Allender, E. 2000. Complexity of finite-horizon Markov decision process problems. *Journal of the ACM* 47(4):681–720.

Nguyen, N.; Nguyen, T.; Roos, M.; and Rothe, J. 2014. Computational complexity and approximability of social welfare optimization in multiagent resource allocation. *Autonomous Agents and Multi-Agent Systems* 28(2):256–289.

Papadimitriou, C., and Zachos, S. 1983. Two remarks on the power of counting. In *Proceedings of the 6th GI Conference on Theoretical Computer Science*, 269–276. Springer-Verlag *Lecture Notes in Computer Science #145*.

Regan, K. 1989. Provable complexity properties and constructive reasoning. Manuscript.

Rothe, J.; Spakowski, H.; and Vogel, J. 2003. Exact complexity of the winner problem for Young elections. *Theory of Computing Systems* 36(4):375–386.

Schaefer, M., and Umans, C. 2002. Completeness in the polynomial-time hierarchy: Part I: A compendium. *SIGACT News* 33(3):32–49. Updated version available online at `ovid.cs.depaul.edu/documents/phcom.pdf`.

Sipser, M. 1982. On relativization and the existence of complete sets. In *Proceedings of the 9th International Colloquium on Automata, Languages, and Programming*. Springer-Verlag *Lecture Notes in Computer Science #140*. 523–531.

Spakowski, H., and Vogel, J. 2000. Θ_2^p-completeness: A classical approach for new results. In *Proceedings of the 20th Conference on Foundations of Software Technology and Theoretical Computer Science*, 348–360. Springer-Verlag *Lecture Notes in Computer Science #1974*.

Toda, S. 1991. PP is as hard as the polynomial-time hierarchy. *SIAM Journal on Computing* 20(5):865–877.

Wagner, K. 1987. More complicated questions about maxima and minima, and some closures of NP. *Theoretical Computer Science* 51(1–2):53–80.

Wagner, K. 1990. Bounded query classes. *SIAM Journal on Computing* 19(5):833–846.