On the Complexity of Second-Best Abductive Explanations

Paolo Liberatore[a,*], Marco Schaerf[a]

[a]DIAG, Sapienza University of Rome, Via Ariosto 25, 00185 Rome

Abstract

When looking for a propositional abductive explanation of a given set of manifestations, an ordering between possible solutions is often assumed. While the complexity of computing optimal solutions is already known, in this paper we consider second-best solutions with respect to different orderings, and different definitions of what a second-best solution is: an optimal solution not already found, or a solution that is optimal among the ones not previously found.

Keywords: Abduction, Propositional logic, Knowledge representation techniques, Knowledge-based systems

1. Introduction

The three basic reasoning mechanisms used in computational logic are deduction, induction, and abduction [1]. Deduction is the process of drawing conclusions from information and assumptions representing our knowledge of the world, so that the fact “battery is down” together with the rule “if the battery is down, the car will not start” allows concluding “the car will not start”. Induction, on the other hand, derives rules from the facts: from the facts that the battery is down and that the car is not starting up, we may conclude the rule relating these two facts. Abduction is the inverse of deduction (to some extent [2]): from the fact that the car is not starting up, we conclude that the battery is down. In a more complete formalization of this environment there are many explanations for a car not starting up. This is an important difference between abduction and deduction, making the former, in general, computationally harder.

A given problem of abduction may have one, none, or even many possible solutions (explanations). Moreover, we need to perform both a consistency check and an inference just to verify an explanation. These facts intuitively explain why abduction is to be expected to be computationally harder than deduction.

[*] Corresponding author

Email addresses: liberato@dis.uniroma1.it (Paolo Liberatore), Marco.Schaerf@uniroma1.it (Marco Schaerf)

This observation has indeed been confirmed by theoretical results. Selman and Levesque [3, 4] and Bylander et al. [5, 6] proved the first results about fragments of abductive reasoning, Eiter and Gottlob [7] presented an extensive analysis, Creignou and Zanuttini [8] and Creignou, Schmidt, and Thomas [9] classified the complexity under two kinds of restrictions, Nordh and Zanuttini [10] located the tractability/intractability frontier, Eiter and Makino [11, 12, 13] studied the complexity of computing all abductive explanations, Hermann and Pichler [14] considered the complexity of counting the number of solutions, Fellow et al. [15] analyzed the problem from the point of view of parametrized complexity. All these studies proved that abduction is, in general, harder than deduction. The analysis has also shown that several problems are of interest in abduction. Not only the problem of finding an explanation is relevant, but also the problems of checking an explanation, or whether a hypothesis is in some, or all, of the explanations (relevance). Some work on the complexity of abduction from non-classical theories has also been done [16, 17, 18].

Abduction is also related to the ATMS [19, 20] and to the set of prime implicates of a propositional formula. Indeed, Levesque [21] has proved that ATMS and prime implicates can be used to find the abductive explanations of a literal from a Horn theory. As a result, ATMS and algorithms for finding prime implicates of a formula can be seen as algorithms that solve the problem of abduction; moreover, finding the prime implicates can be seen as a preprocessing phase. Kernel resolution [22] exploits the particular literals of the observation to drive the clause generation process. Using this algorithm, Del Val derived upper bounds on the number of generated clauses, and proved that some restricted classes of abduction problems are polynomial [23, 24].

Contrarily to deduction, abduction is driven by heuristic principles to best explain the given observations. This means that even if the best possible solution to a given problem is found, there is no warranty that it represents the actual state. As an example, a light bulb may not turn on because it is broken, but also because a complex set of circumstances caused a black out in the whole town; while the first explanation is more likely and should therefore be preferred, it may still be wrong. Therefore, it makes sense not to stop at the first explanation, or even at the set of all possible best explanations, but continue the search for other, less likely, solutions.

Other works studied the complexity of finding a solution for a problem of abduction [3, 4, 5, 6, 7, 8, 9, 10]; this one considers the problem of finding *another* solution after some have been found. The difference is that:

- in previous works, a problem of abduction is given and the task is to find a solution;
- in this article, a problem of abduction and a set of its solutions are given, and the aim is to find another solution.

The difference is that the solution to be found has to be different from the previous ones. Whenever an ordering of likeliness of explanations is given, these solutions are assumed to be among the best ones, and the task is to find another best explanation. The meaning of "another best" in this definition may take two meanings: in the first one, we exclude the given solutions and search for a best one among the remaining ones; in the second, we search for another best solution of the original (unrestricted) problem. These problem are characterized in the framework of the complexity classes, which contain decision problems (those having yes/no solutions). The decision problems considered in this article are: check if a set of hypothesis is a solution, and check if a specific hypothesis is in some solution.

2. Definitions

The process of abduction starts from three elements: a propositional formula T formalizing the domain of interest, a set of variables M representing the current manifestations, and another set of variables H representing their possible explanations. In this article, abduction is formally defined as follows.

Definition 1. *A* problem of abduction *is a triple $\langle H, M, T \rangle$, where T is a propositional formula, M is a set of propositional variables called manifestations and H is a set of propositional variables called hypotheses, with $H \cap M = \emptyset$.*

Intuitively, T describes how the assumptions and manifestations are related. We know that the manifestations M occur, and we want their most likely explanation, where an explanation is a set of assumptions $A \subseteq H$ that implies M and is consistent with T.

Definition 2. *The* set of solutions *or* explanations *of a problem of abduction $\langle H, M, T \rangle$ is the set of all sets of assumptions $A \subseteq H$ such that $A \cup \{T\}$ is consistent and $A \cup \{T\} \models M$:*

$$SOL(\langle H, M, T \rangle) = \{A \subseteq H \mid A \cup \{T\} \textit{ is consistent and } A \cup \{T\} \models M\}$$

It is easy to show instances having exponentially many solutions. Ideally, each instance should have a single solution, the assumptions that have – in the real world – caused the manifestations. At least, there should be a way for eliminating solutions that are known to be less likely than other ones.

This is achieved by employing a preorder $\preceq$ over the subsets of H. Given two subsets $A, A' \subseteq H$, they are related by $A \preceq A'$ if A is considered more likely than A'. The three preorders considered in this article are:

- the cardinality-based preorder: $A \leq A'$ if and only if $|A| \leq |A'|$, where $|.|$ denotes the cardinality of a set; in other words, A is preferred if it contains fewer assumptions than A';
- the subset-based preorder: $A \subseteq A'$; a set of assumptions contained in another one is more likely than it;
- the void preorder: $A \trianglelefteq A'$ for no pair $A, A' \subseteq H$; it captures the case of no assumption about the relative likeliness of the candidate solutions.

Instead of considering all solutions to a problem of abduction, one may restrict attention to the most likely ones. Since likeliness is formalized by $\preceq$, this amounts to consider only the minimal solutions.

Definition 3. *The set of* minimal solutions *of a problem of abduction $\langle H, M, T\rangle$ with respect to the preorder $\preceq$ is:*

$$SOL_{\preceq}(\langle H, M, T\rangle) = \min(SOL(\langle H, M, T\rangle), \preceq)$$

In this definition, $\min(R, \preceq)$ is the set of elements of R that are minimal with respect to $\preceq$, that is, the elements $r \in R$ such that no r' exists with $r' \preceq r$ and $r \not\preceq r'$.

The void preorder makes all solutions minimal: $SOL_{\trianglelefteq}(\langle H, M, T\rangle) = SOL(\langle H, M, T\rangle)$. This allows for the notational simplification of considering only minimal solutions, where the preorder may be $\trianglelefteq$, $\leq$ or $\subseteq$.

2.1. Second-Best Solution

In the conditions of perfect knowledge, the set of minimal solutions of a problem of abduction would always contain a single element: the hypotheses that actually caused the manifestations to happen. Unfortunately, such complete information may not be available, leading to more than one minimal solution. Once one is found, it makes sense to continue the search for other ones. This process is formalized as follows.

Definition 4. *Given a nonempty set of minimal solutions $\{A_1, \ldots, A_m\} \subseteq SOL_{\preceq}(\langle H, M, T\rangle)$ of a problem of abduction, the set of second-best solutions is:*

$$\begin{aligned} & NEXT\_SOL_{\preceq}(\langle H, M, T\rangle, \{A_1, \ldots, A_m\}) \\ = \;& \min(SOL(\langle H, M, T\rangle) \backslash \{A_1, \ldots, A_m\}), \preceq) \end{aligned}$$

The case of empty set of given minimal solutions $\{A_1, \ldots, A_m\}$ is excluded from consideration because it makes the second-best solutions the same as the minimal solutions. This definition can be extended by allowing A_i's to be non-minimal if they contain another A_j. The technical results in this article are unaffected by the change; the original definition is chosen because it is simpler.

2.2. Other Best Solutions

A second-best solution may not be a minimal solution of the original problem. For example, if $\{A_1, \ldots, A_m\}$ includes all minimal solutions, all second-best solutions are not minimal. This is because the definition first excludes $\{A_1, \ldots, A_m\}$ from the set of solutions, and then takes the minimal ones among the remaining ones. If only minimal solutions are of interest, a different definition is more appropriate: given a set of minimal solution, an other-best solution is a minimal solution not in the set of the given ones.

Definition 5. *Given a nonempty set of minimal solutions $\{A_1, \ldots, A_m\} \subseteq SOL_{\preceq}(\langle H, M, T\rangle)$ of a problem of abduction, the set of other-best solutions is:*

$$MIN\_SOL_{\preceq}(P, \{A_1, \ldots, A_m\}) = SOL_{\preceq}(P) \backslash \{A_1, \ldots, A_m\}$$

2.3. Computational Problems of Abduction

There are several computational problems that are relevant for abduction, here we list the ones considered in this article.

- Existence: Decide whether a problem of abduction $P = \langle H, M, T \rangle$ admits a (minimal) solution, that is, $SOL(\langle H, M, T \rangle)$ is non-empty;

- Checking: Decide whether a set of hypotheses A is a minimal explanation, that is, whether $A \in SOL_{\preceq}(\langle H, M, T \rangle)$;

- Relevance: Decide whether a hypothesis h belongs to at least a minimal solution of a problem of abduction $P = \langle H, M, T \rangle$, that is, $\exists A \in SOL_{\preceq}(\langle H, M, T \rangle)$ such that $h \in A$;

A solution can be iteratively found using the Relevance problem: for every $h \in H$, if it is relevant then add it to T, and remove it from H regardless of its relevance. The set of the relevant hypotheses iteratively found in this manner is a solution for the abduction problem. This is therefore a Turing reduction from solution finding to relevance checking, and gives an upper bound to the former problem.

2.4. Computational complexity

The complexity analysis of the problems of second-best explanation is done in the framework of the polynomial hierarchy and many-one polynomial reductions. A number of books on the topic exist [25, 26, 27]. Decision problems (problems having a yes/no answer) are partitioned into classes of increasing complexity. In summary, the class P contains all problems solved by some algorithms that run in time polynomial in the size of their inputs. The class NP is defined in a similar way with the algorithm running on a nondeterministic Turing machine. The class coNP contains all problems whose complement (the problem with reverse yes/no answer) is in NP. The class DP contains all problems that can be split into a subproblem in NP and one in coNP, so that the answer is yes if and only if the answers of the two subproblems are yes. The other classes of the polynomial hierarchy considered in this article are defined in terms of oracles, which are subroutines whose running time is not counted. In particular, the class Σ_2^p contains all problems that are in NP assuming the availability of an oracle solving a subproblem in NP. The class containing all complementary problems is Π_2^p. The class of problems solvable in polynomial time with a logarithmic number of calls to an oracle for Σ_2^p is $\Delta_3^p[\log n]$.

While membership to a complexity class is established by showing an appropriate algorithm (running on deterministic or nondeterministic machines, using oracles or not), proving non-membership is a more difficult task. Currently, even the existence of problems in NP that are not in P has never been proved, only that P≠NPimplies that a problem is not in P if every other problem in NP can be reduced to it via a polynomial-time reduction. Such problems are called NP-hard. If they also belong to NP, they are NP complete. The same definitions apply to DP and Π_2^p. More details about complexity classes and reductions can be found in the cited books on computational complexity [25, 26, 27].

The analysis of complexity is performed by turning search problems into decision problems: from finding a solution to verifying one and to checking the existence of a solution containing a given hypothesis. The second problem is particularly significant regarding the complexity of finding, as a solution can be determined by repeatedly solving it. This and the corresponding problem of dispensability (no minimal solution contains h) have been analyzed by Eiter and Gottlob [7]. In this article, the problem of relevance is considered with the additional assumption that some solutions are already known, possibly with additional information attached.

Most hardness results in this article are proved by translating a problem of abduction into another: for example, the problem of checking a solution to that of checking a second-best solution. This involves proving that certain solutions of the first are turned into solutions of the second. Since being a solution is defined in terms of satisfiability and unsatisfiability, the proofs employ modifications that do not affect these conditions:

1. if a set implies a formula, the formula can be added to the set;
2. a formula entailed by the rest of a set can be removed from the set;
3. if a set contains a literal l and a clause containing l, the latter can be removed; clauses containing the negation of l can be removed this literal; when considering the sign of a literal, a clause written $l \rightarrow s$ is actually $\neg l \vee s$; therefore, l is negated in it;
4. if a variable b only occurs in formulae that are clauses, and is negated in all of them, these can be removed; the same if b only occurs unnegated;
5. in particular, if a variable only occurs in a single clause, that clause can be removed;
6. if a set can be partitioned in subsets not sharing variables, it is satisfiable if and only if each of the subsets is;
7. renaming variables does not affect satisfiability: if X and X' are two sets of variables in bijective correspondence and T a formula, the formula $T[X'/X]$ obtained from T by replacing each variable in X with its corresponding variable in X' is satisfiable if and only if T is.

3. Second-Best Solution

In this section we consider the problem of the second-best solutions, as formalized by Definition 4: given a set of minimal solutions, find one that is minimal among the other ones. As common in computational complexity studies, this search problem is turned into a verification problem in order to evaluate its complexity: given an instance of abduction, a set of solutions and a candidate solution, check whether the latter is a second-best solution. A solution can be found by repeatedly solving problems of relevance, which are also analyzed.

The technical means to prove the hardness of these problems is the following lemma, showing how to introduce a new minimal solution to a problem of abduction.

Lemma 1. *For every problem of abduction P not containing variables s and r, another problem P' can be built in polynomial time such that:*

$$SOL(P') = \{s\} \cup \{A \cup \{r\} \mid A \in SOL(P)\}$$

Proof. Let $P = \langle H, M, T \rangle$ be the original problem of abduction not containing the variables s and r. The problem $P' = \langle H', M', T' \rangle$ is defined as follows, where t is a fresh variable and H'' is a set of fresh variables in bijective correspondence to H:

$$\begin{aligned} H' &= H \cup \{r, s\} \\ M' &= \{t\} \\ T' &= (T[H''/H] \vee \neg r) \wedge \bigwedge\{h \rightarrow h'' \mid h \in H\} \wedge ((r \wedge \bigwedge M) \rightarrow t) \wedge \\ &\quad\; (\neg s \vee t) \wedge (\neg s \vee \neg r) \wedge \bigwedge\{\neg s \vee \neg h \mid h \in H\} \end{aligned}$$

Intuitively, the claim holds because the manifestation t of this instance is only implied by either s or $r \wedge \bigwedge M$, which cannot both be true because of $\neg s \vee \neg r$. In turn, $s, \neg s \vee \neg h$ is inconsistent with every non-empty subset of H. This shows that the solutions of this instance are $\{s\}$ and $\{r\} \cup A$ where A is a solution of the original instance.

The claim can be formally proved in three steps; only the main ideas are shown, as the details are long but tedious, and can be found in a technical report [28].

$\{s\}$ is a solution of P': this is proved by adding $\{s\}$ to T' and simplifying the resulting formula. In particular, t and $\neg r$ are entailed. By applying the rules described in the previous section, what results is the equivalent formula $s \wedge t \wedge \neg r \wedge \bigwedge\{\neg h \mid h \in H\}$. Since this formula is satisfiable and entails t, the claim follows.

every solution of P is also a solution of P' with the addition of r: given $A \in SOL(P)$, the set $A \cup \{r, T'\}$ implies $\neg s$. This set can therefore by simplified, resulting in the formula $\bigwedge A \wedge r \wedge T[H''/H] \wedge \bigwedge\{h \rightarrow h'' \mid h \in H\} \wedge \neg s \wedge ((\bigwedge M) \rightarrow t)$. Since A is a solution of the original problem this formula is satisfiable and entails M; therefore, it also entails t.

every solution of P' is either s or a solution of P with r added to it: since T' contains $\neg s \vee \neg r$ no solution contains both variables. Since T' includes $\neg s \vee \neg r$ and $\neg s \vee \neg h$ for every $h \in H$, it follows that $\{s\} \cup \{T'\}$ entails the negation of every variable in H' but s; therefore, no solution contains s except $\{s\}$.

Regarding the other solutions, a subset $A' \subset H'$ that is satisfiable with T' but contains neither s nor r is not a solution. Indeed, simplifying $A' \cup \{T'\}$ using this assumption results in a formula not containing t.

This proves that every solutions contain either s or r. Since no solution contain both variables thanks to $\neg s \vee \neg r$, a solution not containing s is in the form $A \cup \{r\}$ with $A \subseteq H$. Remains to be proved that A is a solution of P, in this case.

Since T' contains $\neg s \vee \neg r$, it follows that $A \cup \{r, T'\}$ implies $\neg s$. Therefore, all clauses containing $\neg s$ can be removed, leading to the following formula.

$$\bigwedge A \wedge r \wedge \neg s \wedge T[H''/H] \wedge$$
$$\bigwedge\{h'' \mid h \in A\} \wedge \bigwedge\{h \rightarrow h'' \mid h \in H \backslash A\} \wedge ((\bigwedge M) \rightarrow t)$$

Since renaming does not affect satisfiability or entailment, variables H and H'' can be swapped, making $\{h'' \mid h \in A\}$ become A and $T[H''/H]$ become T. This results in a set that is satisfiable (because $A \cup \{T\}$ is so) and entails t (because $A \cup \{T\}$ entails M). □

This lemma shows how to add the new solution $\{s\}$ to a given problem of abduction. This addition makes the problem of finding a solution in the old instance equivalent to finding a solution different from $\{s\}$ in the new one. The solution $\{s\}$ is minimal with respect to the three considered orderings, since no solution of the form $\{r\} \cup A$ is contained in it or has less literals than it. Since the problem modification can be performed in polynomial time, it shows that if the problem of checking a minimal solution is hard for some complexity class, then the corresponding problem of second-best solution checking is hard for the same class. As a result, in the following complexity characterizations of the second-best solution problems the hardness parts are all proved by a simple reference to this lemma.

This lemma provides a reduction from the problem of checking whether $H \in SOL_{\trianglelefteq}(\langle H, M, T\rangle)$ to that of checking whether $H \in NEXT\_SOL_{\trianglelefteq}(\langle H, M, T\rangle, \{A_1, \ldots, A_m\})$, therefore proving the hardness of the second problem from the hardness of the first. Verifying a solution with the empty preorder is mentioned to be DP-hard by Eiter and Gottlob [7], but as far as it was possible to verify no formal proof was published to date. The claim is proved for the particular candidate solution $\emptyset$; since this is minimal if it is a solution, hardness holds for all considered orderings.

Lemma 2. *Checking whether $\emptyset \in SOL(\langle H, M, T\rangle)$ is* DP*-hard.*

Proof. This property is stated by Eiter and Gottlob [7] for an arbitrary candidate solution as an easy corollary of their results, but as far as we know, no proof has been published, possibly because of its extreme simplicity: formulae F and G over variables X translate into the problem of abduction $\langle \emptyset, \{m\}, T\rangle$, where $T = F \wedge (\neg G[X'/X] \rightarrow m)$, X' is a set of fresh variables in one-to-one correspondence with X and m a fresh variable. This is a reduction from the *sat-unsat* problem of checking whether F is satisfiable and G is unsatisfiable to the problem of checking whether $\emptyset$ is a solution of $\langle \emptyset, \{m\}, T \rangle$. Indeed, $\emptyset \cup \{T\}$ is equivalent to $F \wedge (\neg G[X'/X] \rightarrow m)$. This formula is satisfiable if and only if F is satisfiable, since the rest is satisfied by the model where m is true. This means that $\emptyset$ is a solution if and only if F is satisfiable and $\emptyset \cup \{T\} \models m$. The latter condition is equivalent to the unsatisfiability of $F \wedge (\neg G[X'/X] \rightarrow m) \wedge \neg m$, which is equivalent to $F \wedge G[X'/X] \wedge \neg m$. Since F is satisfiable and does not share variables with the rest of the formula, and the same for $\neg m$, the formula is unsatisfiable if and only if $G[X'/X]$ is unsatisfiable. Since satisfiability is unaffected by variable name change, this proves that $\emptyset$ is a solution of $\langle \emptyset, \{m\}, T \rangle$ if and only if F is satisfiable and G is unsatisfiable. This reduction proves that the problem is DP-hard. □

The complexity of checking whether a set of hypotheses is a solution is an easy consequence of this lemma.

Theorem 1. *Checking whether $A \in SOL(\langle H, M, T \rangle)$ is* DP*-complete.*

Proof. Membership follows from the problem being defined as the satisfiability of $A \cup \{T\}$ and the unsatisfiability of $A \cup \{T, \neg \bigwedge M\}$. Lemma 2 proves that the problem is hard even in the particular case $A = \emptyset$. □

Together with Lemma 1, this result proves that the second-best solution problem is DP-hard for $\trianglelefteq$. It is also a member of this class, as the following theorem proves.

Theorem 2. *Checking whether $A \in NEXT\_SOL_{\trianglelefteq}(\langle H, M, T \rangle, \{A_1, \ldots, A_m\})$ is* DP*-complete.*

Proof. By definition, $\trianglelefteq$ is the empty preorder: $A \trianglelefteq A'$ never holds. All solutions are minimal according to this preorder. Reworded: the set of minimal solutions coincides with the set of all solutions.

The problem is in DP because it can be solved by first checking whether $A \cup \{T\} \models M$ and then whether $A \cup \{T\}$ is consistent and $A \not\in \{A_1, \ldots, A_m\}$. The subproblem $A \cup \{T\} \models M$ is in coNP. The rest of the problem can be solved by nondeterministically generating every possible propositional model over the considered variables and checking whether it satisfies $A \cup \{T\}$ and whether A is different from each element of $\{A_1, \ldots, A_m\}$; both steps can be done in polynomial time; as a result, the problem is in DP.

Hardness is a consequence of Lemma 1 and Lemma 2: the former proves that $\{r\} \in NEXT\_SOL_{\trianglelefteq}(P', \{s\})$ if and only if $\emptyset \in SOL(P)$, the latter proves that $\emptyset \in SOL(P)$ is DP-hard. □

Relevance is harder than verification. Intuitively, the complexity increase is due to the necessity of searching for a solution, among the possibly many ones, that contains the hypothesis h to be checked for relevance.

Theorem 3. *Checking the existence of a solution in $NEXT\_SOL_{\trianglelefteq}(\langle H, M, T \rangle, \{A_1, \ldots, A_m\})$ containing a given $h \in H$ is Σ_2^p-complete.*

Proof. The problem can be solved by a nondeterministic algorithm employing an oracle for the propositional satisfiability problem. The algorithm nondeterministically generates each possible $A \subseteq H$ and calls the oracle for the satisfiability of $A \cup \{T\}$ and of $A \cup \{T, \neg \bigwedge M\}$. If the first is satisfiable, the second is unsatisfiable, $h \in A$ and $A \not\in \{A_1, \ldots, A_m\}$, the algorithm returns yes: h is relevant.

By Lemma 1, from a problem of abduction P can build a second problem P' that has the same solutions of P with $\{r\}$ added to each, plus the single new solution $\{s\}$. The hypothesis h is in some solutions of P if and only if h is in some solutions of P' different from $\{s\}$. Since the first problem is Σ_2^p-hard [7, Theorem 4.1.1], the latter is Σ_2^p-hard as well. □

Requiring set-containment minimality does not increase the cost of verifying a solution, which remains DP-complete as for the case of the empty preorder.

Theorem 4. *Checking whether* $A \in SOL_{\subseteq}(\langle H, M, T\rangle)$ *is* DP*-complete.*

Proof. The problem is in DP because it can be solved by a number of parallel satisfiability and unsatisfiability checks. Indeed, that A is a solution is equivalent to the satisfiability of $A \cup \{T\}$ and the unsatisfiability of $A \cup \{T, \neg \bigwedge M\}$. The first condition implies the satisfiability of $A' \cup \{T\}$ for every $A' \subseteq A$. As a result, A is not a minimal solution only if there exists $A' \subset A$ such that $A' \cup \{T\} \models M$. This implies $A \backslash \{h\} \cup \{T\} \models M$ for every $h \in A \backslash A'$ by monotonicity of $\models$. The converse also holds: A is not minimal if such h exists, since $A \backslash \{h\} \subset A$ for every $h \in A$. As a result, A is a minimal solution if and only if:

- $A \cup \{T\}$ is consistent;
- $A \backslash \{h\} \cup \{T, \neg \bigwedge M\}$ is consistent for every $h \in A$;
- $A \cup \{T, \neg \bigwedge M\}$ is inconsistent.

These tests are in polynomial number and can be done in parallel by renaming the variables. As a result, the whole problem amounts to checking whether a formula is satisfiable and another is not.

Hardness is a direct consequence of Lemma 2, which proves that establishing whether $\emptyset \in SOL(\langle H, M, T\rangle)$ is DP-hard. Since $\emptyset$ is contained in every other subset of H, if any, it is a minimal solution if and only if it is a solution. As a result, $\emptyset \in SOL_{\subseteq}(\langle H, M, T\rangle)$ is DP-hard. □

Checking a second-best solution can be proved to be complete for the same class.

Theorem 5. *Checking whether* $A \in NEXT\_SOL_{\subseteq}(\langle H, M, T\rangle, \{A_1, \ldots, A_m\})$ *is* DP*-complete.*

Proof. Membership is proved as in the previous theorem, with two variants. First, A is not a second-best solution if it is in $\{A_1, \ldots, A_m\}$. Second, the check for consistency of $A \backslash \{h\} \cup \{T, \neg \bigwedge M\}$ is skipped if $A \backslash \{h\}$ is in $\{A_1, \ldots, A_m\}$.

Hardness is proved by Lemma 1 and the previous theorem, showing the problem with no given solution DP-hard. The lemma proves that A' is in $SOL(\langle H', M', T' \rangle)$ if and only if either $A' = \{s\}$ or $A' = A \cup \{r\}$ with $A \in SOL(\langle H, M, T \rangle)$. This also implies that $\{s\}$ is a minimal solution. Since s is not in $\langle H, M, T \rangle$, a solution $A \cup \{r\}$ does not contain s, which means that it is minimal if and only if A is minimal. This is therefore a reduction from checking a minimal solution of $\langle H, M, T \rangle$ to that of checking a second-best solution in $NEXT\_SOL_{\subseteq}(\langle H', M', T' \rangle, \{\{s\}\})$. Since the former is DP-hard by the previous theorem, the latter is hard for the same class. □

This result establishes the complexity of verifying a solution of an abduction problem in presence of other minimal solutions. Searching for a solution can be turned into the decision problem of relevance (checking the existence of solutions with a given $h \in H$) as already explained. Relevance for the subset preorder is Σ_2^p-complete [7, Theorem 4.2.1]. Lemma 1 shows how to carry the hardness part of this result to the case where other minimal solutions are known.

Theorem 6. *Checking the existence of a solution in $NEXT\_SOL_{\subseteq}(\langle H, M, T \rangle, \{A_1, \ldots, A_m\})$ containing a given $h \in H$ is Σ_2^p-complete.*

Proof. Membership can be proved by nondeterministically generating all possible subsets A of H and then checking (possibly using the oracle) whether $h \in A$, whether $A \not\in \{A_1, \ldots, A_m\}$, whether $A \cup \{T\}$ is consistent, whether $A \cup \{T\} \models M$ and whether $A \backslash \{h'\} \cup \{T\} \not\models M$ for all $A \backslash \{h'\} \not\in \{A_1, \ldots, A_m\}$ with $h' \in A$.

Hardness is a consequence of the hardness result without the given solutions $\{A_1, \ldots, A_m\}$, since Lemma 1 implies that $A \in SOL_{\subseteq}(\langle H, M, T \rangle)$ if and only if $A \cup \{r\} \in NEXT\_SOL_{\subseteq}(\langle H', M', T' \rangle, \{\{s\}\})$. As a result, h is in some element of $SOL_{\subseteq}(\langle H, M, T \rangle)$ if and only if it is in some element of $NEXT\_SOL_{\subseteq}(\langle H', M', T' \rangle, \{\{s\}\})$. This is a reduction from relevance without given solutions to relevance for second-best solutions, proving the Σ_2^p-hardness of the latter. □

As for $\trianglelefteq$ and $\subseteq$, the hardness of the problems of verification and relevance for the cardinality-based preorder $\leq$ is proved by reducing to them the corresponding problems without the given solutions. The following theorem shows the complexity of the verification problem.

Theorem 7. *Checking whether $A \in SOL_{\leq}(\langle H, M, T \rangle)$ is Π_2^p-complete.*

Proof. Non-membership can be verified with a nondeterministic algorithm employing an oracle for solving the satisfiability problem. Given an abduction problem and a subset $A \subseteq H$, the algorithm nondeterministically generates each possible $A' \subseteq H$. After this A' is produced, the following checks are done, with the help of the oracle: either $A \cup \{T\}$ is unsatisfiable, or $A \cup \{T, \neg \bigwedge M\}$ is satisfiable, or the following three conditions hold: $|A'| < |A|$, $A' \cup \{T\}$ is consistent and $A' \cup \{T, \neg \bigwedge M\}$ is inconsistent. If all these hold, then either A is not a solution or smaller solution A' exists.

Hardness is proved by reduction from the problem of non-relevance, which Eiter and Gottlob [7, Theorem 4.2.1] proved to be Σ_2^p-complete even if the formula T is consistent [7, Definition 2.1.1]. Given a problem of abduction $\langle H, M, T \rangle$ and $h \in H$, a $\leq$-minimal solution of $\langle H, M, T \rangle$ containing h exists if and only if S is not a $\leq$-minimal solution of the problem $\langle H', M', T' \rangle$ defined as follows.

$$
\begin{aligned}
H' &= H \cup S \\
M' &= M \cup \{w\} \\
T' &= T[h''/h][M''/M] \wedge \bigwedge \{m'' \rightarrow m \mid m \in M\} \wedge \\
& \quad (h \rightarrow h'') \wedge (h \rightarrow w) \wedge (\bigwedge S \rightarrow \bigwedge M')
\end{aligned}
$$

If $|H| = n$, then S is a set of $n+1$ fresh variables. Also h'' and w are fresh variables and M'' is a set of fresh variables in one-to-one correspondence with M.

The proof is articulated in two parts: first, regardless of the original instance, S is a solution of the new one; second, the other solutions of the new instance are the solutions of $\langle H, M, T \rangle$ that contain h. Since S has size $|H| + 1$, it is size-minimal only if no solution of the original problem contains h.

The first part is easy to prove: since T' contains $\bigwedge S \rightarrow \bigwedge M'$, all manifestations are entailed by S. The set $S \cup \{T'\}$ can be simplified by removing all clauses entailed by $\bigwedge S \wedge \bigwedge M'$ and their consequences, resulting into $T[h''/h][M''/M] \wedge (h \rightarrow h'')$, which is consistent because T is consistent by assumption.

The second part of the proof requires further elaboration. If A' is a solution of $\langle H', M', T' \rangle$ not containing all of S then the rest of S can be removed from it. Indeed, if A' does not contain s_i then this variable only occurs negated in $A' \cup \{T'\}$; therefore, it can be removed without affecting consistency and entailment of M', which by definition does not contain s_i. The result is a set not containing any of S.

Intuitively, M' is entailed by having in A' either the whole of S, via the subformula $\bigwedge S \rightarrow \bigwedge M'$, or a solution of $\langle H, M, T \rangle$ containing h, thanks to $m'' \rightarrow m$ and $h \rightarrow w$. If A' is a solution of $\langle H', M', T' \rangle$ not intersecting S then $A \cap H$ is a solution of $\langle H, M, T \rangle$. The details of this final claim are omitted because of their simplicity [28], and consist in manipulations of $A' \cup \{T'\}$ via the transformations in Section 2, such as the removal of literals occurring only with one sign in a formula and swapping variables such as m and m''. $\square$

The following theorem shows the complexity of the second best solution verification problem with the cardinality-based preorder.

Theorem 8. *Checking whether $A \in NEXT\_SOL_{\leq}(\langle H, M, T \rangle, \{A_1, \ldots, A_m\})$ is Π_2^p-complete.*

Proof. Membership is proved as follows: A is a second-best solution if it is in $SOL(\langle H, M, R\rangle)$ and for every $A' \subseteq H$ such that $|A'| < |A|$ it holds that either $A' \cup \{T\}$ is inconsistent, $A' \cup \{T\} \not\models M$ or $A' \in \{A_1, \ldots, A_m\}$. All these checks can be done with an NP oracle once a subset $A' \subseteq H$ is nondeterministically generated.

Hardness is proved by Lemma 1: $\{s\}$ is not only a solution but also minimal because all other ones (if any) have the form $H \cup \{r\}$, so their cardinality is larger than or equal to one. Every solution of the original problem is translated into a solution of the new one. This reduction preserves the relative size of explanations, as they are all added one element. Therefore, $A \cup \{r\} \in NEXT\_SOL_{\leq}(\langle H', M', T'\rangle, \{\{s\}\})$ holds if and only if $A \in SOL_{\leq}(\langle H, M, T\rangle)$ holds. □

The problem of existence of a second-best solution containing a given hypothesis can be shown to be $\Delta_3^p[\log n]$-complete.

Theorem 9. *Checking the existence of a solution in* $NEXT\_SOL_{\leq}(\langle H, M, T\rangle, \{A_1, \ldots, A_m\})$ *containing a given* $h \in H$ *is* $\Delta_3^p[\log n]$*-complete.*

Proof. The problem of checking for the existence of a solution A with size bounded by a number k and containing h is in Σ_2^p, as it amounts to nondeterministically generating a solution and then checking it for being a second best-solution and for its size being less than or equal to k. The problem of relevance can be therefore solved by a binary search for the minimal size of solutions [7, Theorem 4.3.2]: start with $k = |H|/2$, and if the result is positive set $k = |H|3/4$, otherwise $k = |H|/4$. Once the minimal size is found, the problem can be solved by nondeterministically generating all solutions of this size not being in $\{A_1, \ldots, A_m\}$ and then checking whether h is in some of them.

Hardness follows from Lemma 1: h is $\leq$-relevant to $\langle H, M, T\rangle$ if and only if a solution of $NEXT\_SOL_{\leq}(\langle H', M', T'\rangle, \{\{s\}\})$ containing h exists; this is proved like in the previous theorem. Since $\leq$-relevance is $\Delta_3^p[\log n]$-hard [7, Theorem 4.3.2], also checking for solutions of $NEXT\_SOL_{\leq}(\langle H, M, T\rangle, \{A_1, \ldots, A_m\})$ containing a given $h \in H$ is $\Delta_3^p[\log n]$-hard. □

4. Other Minimal Solution

The implicit assumption in second-best solutions is that non-minimal solutions are taken into account once all minimal ones have been considered. Indeed, the definition of $NEXT\_SOL(\langle H, M, T\rangle, \{A_1, \ldots, A_m\})$ includes all solutions that are minimal once $A_1, \ldots, A_m$ are removed from consideration. A different approach is to only allow minimal solutions. This is different in that:

- second-best solutions are solutions that are minimal among the ones different from the given ones;

- other minimal solutions are solutions that are minimal and are not among the given ones.

The first definition allows non-minimal solutions if the minimal ones are all among the given ones, the second does not. Only non-minimal solutions are affected. Therefore, the difference disappears when the void preorder $\trianglelefteq$ is considered, as no solution is non-minimal according to it.

When using $\subseteq$ or $\leq$, the two definitions may lead to different results, like for the problem:

$$\begin{array}{rcl} H & = & \{s, r\} \\ M & = & \{t\} \\ T & = & \{s \rightarrow t\} \end{array}$$

The problem $\langle H, M, T\rangle$ has two explanations: $\{s\}$ and $\{s, r\}$. Only the first one is minimal in the two preorders; this is also intuitively correct, as r does not really contribute to entail t. However, the second-best solutions include $\{s, r\}$. Such a possibility is excluded when considering the other minimal solutions: none exists but $\{s\}$.

When $\subseteq$ is used as the preorder, the complexity of checking another minimal solution is the same as that for a second-best solution. This can be proved as for Theorem 5 with minimal changes: for membership, the sets $A\backslash\{h\}$ are checked even if they are in $\{A_1, \ldots, A_m\}$; hardness is proved with the very same reduction, which maps minimal solutions of the original problem into solutions of the new problems that are both second-best solutions and other solutions.

Other best solutions are easier than second-best, if using $\leq$: DP-complete. The following lemma shows how to relate the solutions of a problem to the minimal solutions of another problem. This property will be used to prove that we can reduce the problem of checking a solution to the problem of checking another minimal solution.

Lemma 3. *Let $P = \langle H, M, T\rangle$ be a problem of abduction, where $H = \{h_1, \ldots, h_n\}$. Let $P' = \langle H', M', T'\rangle$ be the problem defined as follows, where C, D, and E are sets of n fresh variables each.*

$$\begin{array}{rcl} H' & = & C \cup D \\ M' & = & M \cup E \\ T' & = & T \cup \{c_i \rightarrow h_i, c_i \rightarrow e_i, d_i \rightarrow e_i \mid 1 \leq i \leq n\} \end{array}$$

It holds:

$$SOL_{\leq}(\langle H', M', T'\rangle) = \{\{c_i \mid h_i \in A\} \cup \{d_i \mid h_i \not\in A\} \mid A \in SOL(\langle H, M, T\rangle)\}$$

Proof. Intuitively, $e_i \in M'$ enforces either c_i or d_i to be in every solution, and minimization excludes solutions containing both. Since c_i entails h_i, M is entailed only if the c_i's correspond to the original solution. Since a solution not containing c_i contains d_i, each solution of P is mapped into a minimal solution of P'.

Given a solution A of P, the corresponding minimal solution of P' is obtained by adding c_i if $h_i \in A$ and d_i otherwise. In the other way around, given a minimal solution A' of P', the solution of the original problem is obtained by selecting the h_i's such that $c_i \in A'$. Indeed, for every i the two clauses $c_i \rightarrow e_i$ and $d_i \rightarrow e_i$ ensure that every solution contains at least one between c_i and d_i; minimal solutions cannot contain both. The clauses $c_i \rightarrow h_i$ establish the correspondence between A and A': since T contains h_i but none of the new variables, consistency and entailment of M is the same as in the original problem.

The claim is formally be proved in three steps: in the first, every solution of P is proved to be translatable into a solution of P'; in the second, every solution of P' can be translated back to a solution of P; in the third, every minimal solution of P' is shown to contain d_i if and only if it does not contain c_i. The full proof of these three steps is long but tedious, and is therefore omitted [28]. In all cases, the conclusions is obtained by modifying formulae like $A \cup \{T\}$, $A' \cup \{T'\}$, $A \cup \{T, \neg m\}$, etc. using the consistency-preserving transformations of Section 2. □

This lemma maps each solution of P into a $\leq$-minimal solution of P', and vice versa. It therefore provides a reduction from the problem of second-best solutions with the void preorder $\trianglelefteq$ to the problem of other minimal solution with the cardinality preorder $\leq$.

Theorem 10. *Checking whether $A \in MIN\_SOL_{\leq}(\langle H, M, T\rangle, \{A_1, \ldots, A_m\})$ is* DP*-complete.*

Proof. Given $\{A_1, \ldots, A_m\}$ with $m \geq 1$, one can check whether A is another minimal solution by expressing $|A| = |A_1|$ as a propositional formula F using fresh variables; this can be done because checking $|A| = |A_1|$ can be done in propositional time, which means that it can be expressed as a circuit of polynomial size [29]. Then, the problem amounts to the satisfiability of $A \cup \{T, F\}$ and the unsatisfiability of $A \cup \{T, \neg \bigwedge M\}$.

Hardness follows the DP-hardness of the problem of verifying $A \in NEXT\_SOL_{\trianglelefteq}(\langle H, M, T\rangle, \{A_1, \ldots, A_m\})$. Indeed, Lemma 3 proves that the solutions $A, A_1, \ldots, A_m$ of $\langle H, M, T\rangle$ can be turned into $\leq$-minimal solutions $A', A'_1, \ldots, A'_m$ of $\langle H', M', T'\rangle$. As a result, A is a solution of $\langle H, M, T\rangle$ not in $\{A_1, \ldots, A_m\}$ if and only if A' is a minimal solution of $\langle H', M', T'\rangle$ not in $\{A'_1, \ldots, A'_m\}$. Since the first problem is DP-hard, the second is DP-hard as well. □

5. Conclusions

In this article, we have investigated the problem of finding a solution to a given abduction problem when some solutions have already been found. The results show that the analyzed problems are computationally intractable, but this does not rule out the possibility of tackling them. It only suggests the most appropriate tools to use. Polynomial problems are best attacked using deterministic polynomial algorithms, while problems in NP can be solved using reduction to the propositional satisfiability problem (SAT) and then passed to a state of the art SAT solver (for example, one of the contestants in the SAT competition `http://www.satcompetition.org/`). Problems in higher classes of the polynomial hierarchy (such as all the problems shown in the paper) can be solved by a reduction to the Quantified Boolean Formulae problem (QBF) and the use of QBF solvers (`http://qbf.satisfiability.org/gallery/`). Problems higher up in the polynomial hierarchy are more complex to solve, but, by identifying their precise complexity, we can better take advantage of the solvers.

There are some open questions and some possible future directions of work. It makes sense to establish the complexity of finding a k-th best solution, at least in the case of the ordering based on cardinality. This can be seen as a variant of the problems studied in this article where the given solutions are not known.

Another question left open by this article is to find a reduction from the problem of second-best solutions to simple abductions that preserve the explanations. What is needed is the opposite of Lemma 1, which shows how to add a given explanation to an abduction problem: a reduction that eliminates some given solutions from an abduction problem while leaving the other ones unchanged.

A further problem worth investigating is whether additional information found during the search for the first solutions may be useful for finding the others. The problems of second-best solutions is the restriction of this one where the additional information is limited to the solutions found, discarding everything else. The extended problem cannot be framed in the standard complexity classes because in general the additional information can only be assumed to be polynomial in size. The compilability classes [30, 31] characterize this kind of problems.

References

[1] C. S. Peirce, Abduction and induction, in: J. Buchler (Ed.), Philosophical Writings of Peirce, Dover, New York, 1955, Ch. 11, pp. 150–156.

[2] M. Cialdea Mayer, F. Pirri, Abduction is not deduction-in-reverse, Journal of the IGPL 4 (1) (1996) 86–104.

[3] B. Selman, H. J. Levesque, Abductive and default reasoning: A computational core, in: Proceedings of the Eighth National Conference on Artificial Intelligence (AAAI'90), 1990, pp. 343–348.

[4] B. Selman, H. Levesque, Support set selection for abductive and default reasoning, Artificial Intelligence 82 (1996) 259–272.

[5] T. Bylander, D. Allemang, M. C. Tanner, J. R. Josephson, Some results concerning the computational complexity of abduction, in: Proceedings of the First International Conference on the Principles of Knowledge Representation and Reasoning (KR'89), 1989, pp. 44–54.

[6] T. Bylander, D. Allemang, M. C. Tanner, J. R. Josephson, The computational complexity of abduction, Artificial Intelligence 49 (1991) 25–60.

[7] T. Eiter, G. Gottlob, The complexity of logic-based abduction, Journal of the ACM 42 (1) (1995) 3–42.

[8] N. Creignou, B. Zanuttini, A complete classification of the complexity of propositional abduction, SIAM J. Comput. 36 (1) (2006) 207–229. doi:10.1137/S0097539704446311.

[9] N. Creignou, S. J., M. Thomas, Complexity classifications for propositional abduction in post's framework, J. Log. Comput. 22 (5) (2012) 1145–1170. doi:10.1093/logcom/exr012.

[10] G. Nordh, B. Zanuttini, What makes propositional abduction tractable, Artificial Intelligence 172 (2008) 1245–1284.

[11] T. Eiter, K. Makino, On computing all abductive explanations, in: Proceedings of the Eighteenth National Conference on Artificial Intelligence (AAAI 2002), 2002, pp. 62–67.

[12] T. Eiter, K. Makino, Abduction and the dualization problem, in: Discovery Science, 2003, pp. 1–20.

[13] T. Eiter, K. Makino, Generating all abductive propositional horn theories, in: Seventeenth International Workshop on Computer Science Logic, 2003, pp. 197–211.

[14] M. Hermann, R. Pichler, Counting complexity of propositional abduction, Journal of Computer and System Sciences 76 (2010) 634–649.

[15] M. Fellows, A. Pfandler, F. Rosamond, S. Rümmele, The parameterized complexity of abduction, in: Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence, 2012.

[16] T. Eiter, G. Gottlob, N. Leone, Semantics and complexity of abduction from default theories, Artificial Intelligence 90 (1997) 177–223.

[17] T. Eiter, G. Gottlob, N. Leone, Abduction from logic programs: Semantics and complexity, Theoretical Computer Science 189 (1997) 129–177.

[18] D. Calvanese, M. Ortiz, M. Simkus, G. Stefanoni, The complexity of conjunctive query abduction in DL-Lite, in: Proceedings of the twentyfourth International Workshop on Description Logics (DL 2011), 2011.

[19] J. de Kleer, An assumption-based TMS, Artificial Intelligence 28 (1986) 127–162.

[20] R. Reiter, J. de Kleer, Foundations of assumption-based truth maintenace systems: Preliminary report, in: Proceedings of the Sixth National Conference on Artificial Intelligence (AAAI'87), 1987, pp. 183–188.

[21] H. J. Levesque, A knowledge-level account of abduction, in: Proceedings of the Eleventh International Joint Conference on Artificial Intelligence (IJCAI'89), 1989, pp. 1061–1067.

[22] A. Del Val, A new method for consequence finding and compilation in restricted languages, in: Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI'99), 1999, pp. 259–264.

[23] A. Del Val, On some tractable classes in deduction and abduction, Artificial Intelligence 116 (2000) 297–313.

[24] A. Del Val, The complexity of restricted consequence finding and abduction, in: Proceedings of the Seventeenth National Conference on Artificial Intelligence (AAAI 2000), 2000, pp. 337–342.

[25] D. Bovet, P. Crescenzi, Introduction to the Theory of Complexity, Prentice-Hall international series in computer science, Prentice Hall, 1994.

[26] M. Sipser, Introduction to the Theory of Computation, International Thomson Publishing, 1996.

[27] S. Arora, B. Barak, Computational Complexity: A Modern Approach, Cambridge University Press, 2009.

[28] P. Liberatore, M. Schaerf, On the complexity of finding second-best abductive explanations, Computing Research Repository (CoRR) abs/1204.5859. URL http://arxiv.org/abs/1204.5859

[29] R. Boppana, M. Sipser, The complexity of finite functions, in: J. van Leeuwen (Ed.), Handbook of Theoretical Computer Science, Vol. A, Elsevier Science Publishers (North-Holland), Amsterdam, 1990, Ch. 14, pp. 757–804.

[30] M. Cadoli, F. M. Donini, P. Liberatore, M. Schaerf, Preprocessing of intractable problems, Information and Computation 176 (2) (2002) 89–120.

[31] P. Liberatore, Monotonic reductions, representative equivalence, and compilation of intractable problems, Journal of the ACM 48 (6) (2001) 1091–1125.