The Complexity of Determining Critical Sets

Fermi Ma, Ariel Schvartzman, and Erik Waingarten

MIT,
77 Mass Ave., Cambridge, MA 02139, USA,
{fermima,arielsc,eaw}@mit.edu

1 Introduction

In 2012, McGuire, Tugermann, and Civario confirmed the conjecture that there are no uniquely solvable 16-clue Sudoku puzzles [**?**]. It had long been known that 17 clues was sufficient, and there exists an online database containing over 50,000 such puzzles. Thus, the result by McGuire et al. completely solves the problem of determining the minimum number of Sudoku clues needed.

We consider a following generalization of the above Sudoku question: for any given problem, what is the fewest number of clues necessary to make it uniquely solvable? For example, if the problem is to find a Hamiltonian cycle in a given graph $G = (V, E)$, our question would be to determine the minimum number of edges you can specify before only one Hamiltonian cycle contains all of these edges. This general problem is motivated by possible applications to puzzle-making. A puzzle-maker for a newspaper might want to make a puzzle that contains as few clues as possible, but is still uniquely solvable (as the puzzle-maker might want to publish the answer in the next day's paper).

We formalize this problem by defining the "Fewest Clues Problem" (FCP). We treat FCP as a new complexity class and we consider the idea of FCP-hard problems. We use this framework to analyze the FCP versions of classic NP-hard problems such as SAT and 3SAT, as well as NP-hard puzzle problems such as Latin square completion and Sudoku completion.

The paper is organized as follows. In Section 1, we discuss previous research done on this sort of problem for the specific cases of Sudoku, Latin Squares, and graph coloring. We note that while this specific problem has been studied extensively, it is almost always done so with respect to a certain problem. In Section 2, we motivate and formally define the complexity class FCP. We state what it means for a problem to be FCP, and provide a framework for FCP reductions. In Section 3, we consider FCP versions of the SAT, 3SAT, Triangle Partition, Latin Squares and Sudoku completion problems, and show why these problems are FCP-hard. Then in Section 4, we show how the complexity class FCP fits into the landscape of well-known classes such as P, NP, Σ_2^P, and PSPACE. Finally, in Section 5 we give concluding remarks and ideas for further research.

Related Work. This problem has been investigated before under a variety of different names. The problem of determining the smallest Sudoku has been

referred to as the "minimum number of clues problem" in by McGuire et al. in [**?**], as well as the "minimal fair puzzle" problem. The problem of finding the smallest partial Latin Square with a unique solution has been called the "minimum critical set" problem by Ghandehari et al. in [**?**]. They present upper and lower bounds for this problem.

In 2013, Cooper and Kirkpatrick [**?**] extended this problem beyond the realms of puzzles. They introduce a similar notion for the graph coloring problem. Given a graph and a proper coloring of the graph, they refer to a *critical set* as the smallest set of vertices such that when the coloring is restricted to them, there is a unique proper coloring for the rest of the graph. In order to understand the complexity of this problem, they define problems that are natural upper and lower bounds of it. They show that computing the upper and lower bounds is NP-hard.

To the knowledge of the authors, there aren't any significant efforts to generalize this concept to NP search problems.

2 FCP-: Fewest Clues Problem

2.1 Formalizing the Question

In this section, we formalize the question of determining the fewest number of clues for a given instance of a problem. We will be working with NP search problems, this is the class of problems whose decision is in NP, but we ask for a particular solution. In the case of Hamiltonian Path, instead of asking does there exists a Hamiltonian path, we are asking to present a Hamiltonian path. It is clear that NP search problems are "harder" than decision problems, since one can answer the decision problem once one has found a solution.

In particular, we will have an NP search problem be comprised of a set of *instances*, which will be strings x over some finite alphabet, and for an instance a set of possible *solutions* to be strings s of polynomial size with respect to $|x|$ over some finite alphabet. Each problem also comes with an algorithm A that checks whether a string s is in fact a solution for an instance x.

A *clue* will be an indexed character of a solution. So a clue indicates what character lies at some position of the solution.

Definition 1. *Given some* NP *search problem A, the "fewest clues" variation of A,* (FCPA) *asks for the minimum number of clues corresponding to a unique solution for any given instance.*

This set-up allows us to ask FCP variations to the common puzzles like Sudoku quiet easily. For the classical 9×9 Sudoku, we let the alphabet be $\Sigma = \{1, ..., 9\} \cup \{\square\}$, and we let an instance be a string of length 81 with elements from Σ, where each position in the string corresponds to a position on the 9×9 Sudoku board. The presence of $\square$ indicates that position is unknown. A solution will be a string of length 81 with no $\square$ characters.

The algorithm A checks whether a solution is correct first checks that in every row, every column, and every 3×3 square contains all the numbers from $\{1, ..., 9\}$ and that the solution contains the same numbers as the instance at the specified positions. Figure **??** shows a blank Sudoku puzzle indicating the rows, columns, and 3×3 squares.

2	6	4	7	1	5	8	3	9
1	3	7	8	9	2	6	4	5
5	9	8	4	3	6	2	7	1
4	2	3	1	7	8	5	9	6
8	1	6	5	4	9	7	2	3
7	5	9	6	2	3	4	1	8
3	7	5	2	8	1	9	6	4
9	8	2	3	6	4	1	5	7
6	4	1	9	5	7	3	8	2

Fig. 1. 17-Clue 9×9 Sudoku, taken from [**?**]

Therefore, the FCP Sudoku puzzle asks which is the fewest number of squares we must fill in order for the Sudoku to be uniquely solvable. A solution for the 9×9 case where the instance is initially empty is given in Figure 2.1. The fewest clues are given by the yellow squares.

We can also ask the corresponding decision question for some fewest clues problem: does there exist less than k clues which uniquely specify a solution for a given instance? By being able to answer this question, we can solve for the minimum number of clues necessary with a binary search through k.

2.2 The FCP Class

We now present definitions that give rise to a complexity class, which we call FCP (Fewest Clues Problem). This will sound very familiar to the above notions of solution and clue. We generalize the FCP variation to all NP search problems without having to explicitly specify the instance, solution and checker algorithm by framing the problem in terms of non-deterministic Turing machines. This generalization will give us an easy extension of the Cook-Levin theorem, which will show that FCP SAT is FCP-complete.

We can assume without loss of generality that non-deterministic Turing machines (NTMs) have a branching factor of 2 at each non-deterministic step. In

addition, when the NTM performs a non-deterministic step where the computation branches, we assume that we have a canonical ordering of the two branches. We can call one the "left" branch, and the other the "right" branch. We also assume without loss of generality that all branches of the computation have length equal to a fixed polynomial since we are only dealing with NP- search problems.

Definition 2. *A partial certificate is a read-only tape which contains at each tape cell one of three possible characters, 0, 1, or* $\perp$*.*

Definition 3. *The size of a partial certificate is the number of non-null entries.*

For any NP search problem, a partial certificate is the generalization of a certificate. Also, any certificate of an NP problem can be thought of as a partial certificate. We think of partial certificates as ordinary certificates with some tape cells erased.

Definition 4. *A modified non-deterministic Turing machine (mNTM) is an NTM where there is an additional input of a partial certificate. At each step of the computation, an mNTM reads the next tape cell of the partial certificate and if the partial certificate contains a 0, it takes the left branch of the computation. If it contains a 1, it takes the right branch of the computation. If it contains* $\perp$*, then it behaves non-deterministically. Other than that, an mNTM behaves exactly like an NTM.*

In some sense, a partial certificate makes an mNTM more deterministic. If a partial certificate contains no $\perp$ characters, then the mNTM behaves like a deterministic Turing machine. Alternatively, we can think of the partial certificate as feeding in clues to the mNTM in order to guide its computation. It is clear that given any NTM, we can construct a corresponding mNTM which runs in exactly the same manner, but has an additional input of a partial certificate. With these definitions, we are ready to define the class FCP.

Definition 5. FCP *is the class of function problems of the following form:*

> *Given an NTM running in polynomial time along with an input* x*, compute the smallest a partial certificate such that the corresponding mNTM running on* x *with the partial certificate has only one accepting branch?*

Proposition 1. *Any problem in* FCP *can be phrased as a fewest clues problem.*

Proof. Suppose we are given NTM M which runs in polynomial time with respect to the length of the input. We create a fewest clues problem FCPM:

- Instances: these are the instances of the NTM
- Solutions: these are strings with characters in $\Sigma = \{0, 1\}$.
- Checker algorithm: Given an instance x and a solution s, simulate an mNTM which corresponds to the NTM M with partial certificate s. If the runtime of the mNTM is not equal to $|s|$, then reject. If it is equal, and so the mNTM behaves deterministically and accepts, then accept.

1. Each clue in FCPM corresponds to a character in the certificate.
   Proof: This is because each clue is an indexed character of the solution, which is in turn an indexed character of a certificate.
2. A set of clues corresponds to a partial certificate for M.
   Proof: This follows from 1 by taking many of characters in the certificate.
3. If C is a set of clues with a unique solution, then the corresponding partial certificate will have a unique accepting branch.
   Proof: Suppose there is another accepting branch in the mNTM. The non-deterministic turns at each step specify two possible solutions for the set of clues C.
4. If C is the smallest set of clues with a unique solution, then the corresponding partial certificate is the smallest partial certificate with mNTM having a unique accepting branch.
   Proof: Follows from 3 and the fact that the size of the partial certificate is the same as the number of clues.
5. Therefore, a problem in FCP can be phrased as a fewest clues problem.
   Proof: follows from 4 and because each partial certificate corresponds to a set of clues.

□

Proposition 2. *Any fewest clues problem with solution alphabet $\Sigma = \{0, 1\}$ is in* FCP*.*

Proof. We assume that we have some fewest clues problem, FCPA, where:

- The set of instances is I.
- The set of solutions is $S(x)$ for each $x \in I$ where each string is a character from some alphabet Σ and each $s \in S(x)$ has $|s| \leq p(|x|)$.
- We have an algorithm A with inputs $x \in I$, $s \in S$ that decides in polynomial time whether s is a solution to x.

We will construct a NTM M such that the smallest partial certificate for M will correspond to the fewest clues in FCPA.
NTM M: Given an instance x, non-deterministically, for $p(|x|)$ steps, generate a solution s by writing a character in each step, where the left branch writes 0 in the solution and the right branch writes 1 in the solution. Then simulate A with inputs x and s. Accept if A accepts, and reject if A rejects.

1. A partial certificate corresponds to a set of clues of the same size.
   Proof: The partial certificate only applies to non-deterministic steps, where M writes down a solution. Therefore, a partial certificate tells the mNTM which characters to write at which positions deterministically, which correspond to a set of clues.
2. If a partial certificate has an mNTM with only one accepting branch, then the corresponding set of clues has a unique solution.
   Proof: If the partial certificate has only one accepting branch, then it means that the corresponding set of clues have only a unique way to complete the clues to have a solution, since each branch explores every possible solution which is a superset of the set of clues.

3. If there exists a set of clues with a unique solution, there exists a partial certificate with only one accepting branch.
   Proof: The set of clues specifies a partial certificate. If there is one way to fill the set of clues to have a solution, each solution will correspond to a branch of the computation.
4. The smallest partial certificate with a unique accepting branch in the mNTM is the smallest set of clues with a unique solution.
   Proof: follows from 2 and 3 and the fact that corresponding partial certificates and sets of clues have the same size.

□

We are almost there, we just need to show that the condition that $\Sigma = \{0, 1\}$ is unnecessary. We do this by cleverly choosing how to branch.

Proposition 3. *Any fewest clues problem is in* FCP.

Proof. The only problem with the above proof for arbitrary *finite* Σ is that we simply cannot write down every possible solution in the non-deterministic step in $p(|x|)$ steps. Let $|\Sigma| = \sigma$. If we require the NTM M to have branching factor 2, it will take at least $\log \sigma$ non-deterministic steps to specify a unique character in Σ to write down. This is a problem, since a specific character in the partial certificate might not be a clue, but rather might say, "the ith character of the solution is in the first half of Σ". We would no longer maintain the number of clues equal to the size of the partial certificate, since each clue would correspond to $\log \sigma$ characters in a partial certificate.

Instead of taking $\log \sigma$ steps to decide which character of Σ to write in the solution, we use σ steps. The first non-deterministic step will choose whether to write the first element of Σ. The second will choose whether to write the second. The third step will choose the third element of Σ and so on (see Figure 2.2). Since each branch of the computation must be the same length, once we pick, we continue branching for σ steps, and if we ever take a right branch again, will reject. I claim now that each partial certificate corresponds to a clue of the same size and vice-versa, allowing up to complete the theorem.

1. Each clue corresponds to a segment of σ characters in a certificate.
   Proof: This is obvious, since it takes us σ non-deterministic steps to make a choice of character.
2. There are no 0's in the smallest partial certificate with a unique accepting branch.
   Proof: Suppose there was a 0 in the smallest partial certificate with a unique accepting branch, then that 0 corresponds to a taking some left branch. The left branch was in the process of picking some character, and suppose the unique accepting branch picked $a \in \Sigma$, then replace the 0 in the partial certificate, with the 1 which picked a in that segment of σ branches. This is also a partial certificate of the same size.
3. The 1's in the smallest partial certificate are inside their chunk of σ branches.
   Proof: if there were two 1's inside the chunk, the branch will reject.

4. If the ith branch of the jth chunk of σ branches in the smallest partial certificate corresponds to the clue of picking the ith element of Σ for index j in the solution.
   Proof: follows from the fact that there are only 1's, each in their σ long chunks.
5. Each smallest partial certificate with unique accepting branch corresponds to a set of clues of the same size with a unique solution.
   Proof: this follows from 4 and the fact that we know how to go from clues to partial certificates.

□

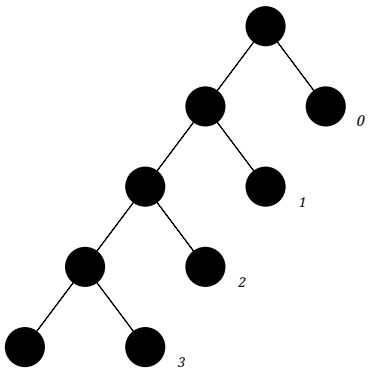

Fig. 2. Branching to pick $\Sigma = \{0, 1, 2, 3\}$

The above proposition justifies our analysis of this class for answering these questions.

3 Some FCP-complete Problems.

In this section, we proceed to show that FCP Sudoku is FCP-complete. In order to do so, we first show that FCP SAT is FCP-complete. We then follow a known chain of reductions that is used to show that the problem of filling in a Sudoku is NP-complete.

3.1 FCP SAT

The following is the fewest clues version of the Boolean satisfiability problem.

– Instance: a Boolean formula ϕ.

– Question: What is the smallest partial assignment of the variables which is uniquely satisfiable?

We will prove that FCP SAT is FCP-complete using the same construction as in the Cook-Levin theorem. We include the theorem here since our proof draws on many of their ideas. The proof is from [**?**], but the authors give it as it appears in Wikipedia.

Theorem 1. *SAT is* NP*-complete.*

Proof. For any NTM M which runs in polynomial time, and given an input x to the M, we will make a Boolean expression ϕ of polynomial size in x such that the formula is satisfiable if and only if M accepts on input x.

Recall that a NTM M is defined by a tuple of states Q, Σ a finite alphabet of tape symbols, $s \in Q$ a starting state, $F \subset Q$ a set of accept states, and $\delta \subset ((Q - F) \times \Sigma) \times (Q \times \Sigma \times \{-1, 1\}))$ is the non-deterministic transition function. Suppose M runs in $p(|x|)$ where p is some polynomial.

For each $q \in Q$, $-p(|x|) \leq i \leq p(|x|)$, $j \in \Sigma$, and $0 \leq k \leq p(|x|)$, we have the variables outlined in the following table:

Variables	Intended Interpretation	How many?
$T_{i,j,k}$	True if the tape cell i contains the symbol j at time k	$O(p(\|x\|)^2)$
$H_{i,k}$	True if the head position is at tape cell i at time k	$O(p(\|x\|)^2)$
$Q_{q,k}$	True if the NTM is at state q at time k	$O(p(\|x\|))$

Now that we've defined polynomially many variables, we will make the clauses of the Boolean formula.

Clause	Conditions	Interpretation	How many?
$T_{i,j,0}$	Cell i contains $j \in \Sigma$ initially	initial content of tape match x	$O(p(\|x\|))$
$Q_{s,0}$		set starting state	1
$H_{0,0}$		starting head position at 0	1
$\neg T_{i,j,k} \vee \neg T_{i,j',k}$	$j \neq j'$	at most one symbol per cell	$O(p(\|x\|)^2)$
$\bigvee_{j \in \Sigma} T_{i,j,k}$		at least one symbol per cell	$O(p(\|x\|)^2)$
$T_{i,j,k} \wedge T_{i,j',k+1} \to H_{i,k}$	$j \neq j'$	tape only changes when written	$O(p(\|x\|)^2)$
$\neg Q_{q,k} \vee \neg Q_{q',k}$	$q \neq q'$	Only one state at a time	$O(p(\|x\|))$
$\neg H_{i,k} \vee \neg H_{i',k}$	$i \neq i'$	Only one head position at a time	$O(p(\|x\|)^2)$
$(H_{i,k} \wedge Q_{q,k} \wedge T_{i,\sigma,k}) \to$ $\bigvee_{(q,\sigma,q',\sigma',d) \in \delta}$ $(H_{i+d,k+1} \wedge Q_{q',k+1} \wedge T_{i,\sigma',k+1})$	$k \leq p(\|x\|)$	possible transitions	$O(p(\|x\|)^2)$
$\bigvee_{k \leq p(\|x\|)} \bigvee_{t \in F} Q_{f,k}$		must have an accepting state	1

If the formula is satisfiable, then the satisfying assignments corresponds to a branch of the computation on M. Likewise, if there is an accepting branch, then the computation through that branch given the values of the variables, since the branch is accepting, all the clauses are satisfied. □

Theorem 2. FCP *SAT is* FCP*-complete.*

Proof. We use the reduction from the Cook-Levin theorem, described in Theomem 1.

1. We assume that we have a NTM M with input x running in polynomial time.
2. We compute the Boolean formula ϕ corresponding to M with input x as in the Cook-Levin theorem.
3. Since we assumed that M has branching factor at most 2, at each step, we add the variables l_k which is True if and only if M takes the left branch of the computation at time k.
4. We transoform each clause $(H_{i,k} \wedge Q_{q,k} \wedge T_{i,\sigma,k}) \rightarrow \bigvee(H_{i+d,k+1} \wedge Q_{q',k+1} \wedge T_{i,\sigma',k+1})$ into
$$(H_{i,k} \wedge Q_{q,k} \wedge T_{i,\sigma,k}) \rightarrow \begin{pmatrix} (H_{i+d,k+1} \wedge Q_{q',k+1} \wedge T_{i,\sigma',k+1} \wedge l_k) \\ \vee \\ (H_{i+d',k+1} \wedge Q_{q'',k+1} \wedge T_{i,\sigma'',k+1} \wedge \neg l_k) \end{pmatrix}$$
Proof: Here, we are encoding in the variable l_k, whether we took the left branch or the right branch. Also, we are using the fact that the non-deterministic transition contains only two possible branches.
5. We add the following clauses to obtain ϕ':
$$(H_{i,k} \wedge Q_{q,k} \wedge T_{i,\sigma,k} \wedge l_k) \leftrightarrow a_{l,i,q,\sigma,k}$$
$$(H_{i,k} \wedge Q_{q,k} \wedge T_{i,\sigma,k} \wedge \neg l_k) \leftrightarrow a_{\neg l,i,q,\sigma,k}$$
Proof: We can do this without increasing ϕ too much since we have $O(p(|x|)^2)$ clauses.
6. ϕ' is satisfiable if and only if ϕ is satisfiable.
Proof: We know whenever ϕ' is satisfiable, ϕ is satisfiable since it contains more clause requirements. Now if ϕ is satisfiable, then we know that for each time k, only one of each head, state, and time position variables are True, in addition, we either took the left branch, or right branch, so that gives us the value of l_k, so we set the corresponding $a_{l,i,q,\sigma,k}$ or $a_{\neg l,i,q,\sigma,k}$ to True and all other $a_{l,i',q',\sigma',k}$ and $a_{\neg l,i',q',\sigma',k}$ to False.
7. Let P be the smallest partial assignment of the variables in ϕ' such that ϕ' is uniquely satisfiable after the assignment.
8. In order to simplify notation, we use l' be the literal l or $\neg l$.
9. Suppose some $a_{l',i,q,\sigma,k}$ in P is assigned to False, then we can replace it with some other $a_{l'',i',q',\sigma',k}$ which is assigned to True.
Proof: In the unique solution, there is some $a_{l',i',q',\sigma',k}$ which is True. We also know that $a_{l'',i',q',\sigma',k} \rightarrow \neg a_{l',i,q,\sigma,k}$ since there can only be one state variable, one head position variable, and one tape cell variable for that head position be True at that time.
10. We can find this is alternate $a_{l'',i',q',\sigma',k}$ in polynomial time.
Proof: There are polynomially many $a_{l'',i',q',\sigma',k}$ so we can write the formula ϕ' which remains after all the $P - \{a_{l',i,q,\sigma,k}\}$ are assigned. Furthermore, we assign $a_{l'',i',q',\sigma',k}$ and check whether the smallest partial assignment of this formula is empty (i.e. there exists a unique solution).

11. Likewise, if there is any other type of variable in P which is assigned to False, we can replace it with a True variable.
    Proof: the same procedure works as in 6 and 7 work.
12. If some variable in P is not a variable of type a, then we can replace it by a variable of type a.
    Proof: Existance holds since at that step, there is an action taken by NTM M, and we can find it by the same procedure as in 7.
13. So we can find a smallest partial assignment P made up of all variables of the form
    $$a_{l',i,q,\sigma,k}$$
    Proof: follows from 7 and 9.
14. The variables $a_{l',i,q,\sigma,k}$ being True, indicate whether at a given time, M took a left branch or a right branch.
    Proof: In fact, we have complete information of the transition NTM M made at the kth step. If $l' = l$, then we took a left branch, if $l' = \neg l$, then we took a right branch.
15. If the partial assignment has a unique satisfying assignment, then the mNTM M has a unique accepting branch.
    Proof: This follows from 14 and the fact that each assignment of the variables corresponds bijectively with a branch of the computation of M.
16. The smallest partial assignment with a unique satisfying assignment is the same size as the smallest partial certificate with a unique accepting branch.
    Proof: For each variable, we have one indication of a branch, and for each transition made by an accepting branch, we have a variable in the partial certificate.

□

From this we can easily show that the fewest clues version of the 3SAT problem is FCP-complete.

Theorem 3. FCP *3SAT is* FCP*-complete.*

Proof. We reduce from FCP SAT.

1. We assume without loss of generality that each SAT formula is a conjuction of clauses, where each are disjunctions.
   Proof: these are the kind of Boolean formulas we get from the Cook-Levin theorem.
2. Suppose C is a clause, we define a function g to handle C. There are four cases:
   - $C = x_1$. Then we let $g(C) = (x_1 \vee x_1 \vee x_1)$.
   - $C = (x_1 \vee x_2)$. Then we let $g(C) = (x_1 \vee x_2 \vee x_1)$.
   - $C = (x_1 \vee x_2 \vee x_3)$, then we just let $g(C) = C$.
   - $C = (x_1 \vee x_2 \vee \cdots \vee x_n)$, then we let
     $$g(C) = (x_1 \vee x_2 \vee z_2) \wedge (\overline{x_2} \vee \overline{z_2}) \wedge (x_2 \vee z_2) \wedge g((\overline{z_2} \vee x_3 \vee \cdots \vee x_n))$$

3. $\phi = \bigvee C_i$ is satisfiable if and only iff $\phi' = \bigvee g(C_i)$ is satisfiable.
   Proof: A satisfying assignment for one gives a satisfying assignment for the other. The only case to check is the last case with big clauses. In this case, we let $z_i = \overline{x_i}$, and this gives a satisfying assignment.
4. If some z_i is in a partial assignment of ϕ', we can replace it with a variable x_i.
   Proof: this is because $z_i \leftrightarrow \overline{x_i}$.
5. There exists a smallest partial assignment of ϕ' which contains only variables of ϕ, and we can find it in polynomial time.
   Proof: we can replace the added variables using the implication.
6. the smallest partial assignment of ϕ' with a unique satifying assignment containing no added variable is a smallest partial assignment of ϕ with a unique satisfying assignment.
   Proof: If ϕ' yields a unique satisfying assignment, then so does ϕ. The only thing to check is that it could not be smaller. This is true because a partial assignment of ϕ is a partial assignment of ϕ' with the same information.

□

3.2 Reductions

By analyzing the previous two reductions (Theorem 2 and Theorem 3), we can generalize the process we used in order to show these problems were FCP-hard. In particular, we started with some FCP-hard problem FCPA, and wanted to show that problem FCPB was FCP-hard.

We summarize our method of reduction in the following lemma.

Lemma 4. *Let* FCPA *be an* FCP*-hard problem. Let* FCPB *be a problem. Denote* C_A *as the clues of problem* A *and* C_B *as the clues for* B*.*

- *Let* $f : A \to B$ *map instances of* A *into instances of* B*.*
- *Let* $C'_B \subset C_B$ *be a particular set of clues for* C_B*, and* $h : C'_B \to C_A$ *be a bijection on the clues.*
- *Let* M *be an algorithm which runs in polynomial time and takes a set of clues for* B *and outputs a set of clues with clues in* C'_B *of the same size.*

Suppose we satisfy the condition that if $C_{f(x)} \in C'_B$ *is a set of clues for* $f(x)$*, then* $\{h(c)|c \in C_{f(x)}\}$ *is a set of clues solution for* x*. Then* FCPB *is* FCP*-hard.*

Proof. Let x be an instance of FCPA. Suppose C is a set of fewest clues of $f(x)$. Then let $M(C) = C_{f(x)}$ is another set of fewest clues with all clues are in C'_B. Now I claim that $C_x = \{h(c)|c \in C_{f(x)}\}$ is a fewest clues set for x. We know that its is a set of clues for x by the hypothesis. Suppose that there existed a smaller one, since h is a bijection, it would give us a smaller set of clues for $f(x)$, which would contradict $C_{f(x)}$.

Suppose that C_x did not yield a unique solution, then there exists $C_x^{(1)}, C_x^{(2)}$ such that these sets are not equal, and $C_x \subset C_x^{(1)} \cap C_x^{(2)}$ which are clues for different solutions. This would imply that $C_{f(x)}$ has distinct supersets which are subsets of two different solutions. □

As an example, in Theorem 3, $f : SAT \to 3SAT$, was the mapping which applied the mapping g to each clause. C'_B was the set of assignments of ϕ' whose variables are x_i, and h was the identity. The algorithm M took each assignment of z_i and replaced it with an assignment of x_i. The algorithm was correct since $z_i \leftrightarrow \overline{x_i}$.

Lemma 5. FCP *1-in-3SAT is* FCP*-hard.*

Proof. We use the following reduction. We map each clause of ϕ, (x_1, x_2, x_3) into five clauses,

$$(\overline{x_1}, a, b)$$
$$(x_2, b, c)$$
$$(\overline{x_3}, c, d)$$
$$(b, \overline{f}, f)$$

Where we added the variables a, b, c, d, e, f for each clause. In this case, we pick our special clues to be the clues with variables corresponding to variables in ϕ.

Note that in a fewest clues assignment, if any of a, c, d, f, g, h is present, it is present with a value of true. If it is in the fewest clues assignment as a value of false, then if we can derive the truth value of the clause, it would imply the false assignment, so we would get rid of it.

One issue is that happens if b and c imply the values of two variables. However, if b or c is true, then we must need another clue for e or f (in b's case), or g or h (in c's case). So if b or c is true, we know that two variables are true, □

3.3 FCP Triangle Partition

This problem is a known NP-complete problem which asks to partition the edges of a graph into triangles [**?**]. We present the fewest clues version of the problem since we will use it to show hardness results for FCP Latin Squares and FCP Sudoku. There is a general description of the problem in [**?**]. A specific instance is addressed in [**?**]. We will use the definition in [**?**]. We first present a definition of a property of tripartite graphs which will simplify the analysis.

Definition 6. *A tripartite graph is* uniform *if each vertex has equally many neighbors in each of the other two partitions.*

- Instance: A uniform tripartite graph, along with a parameter k.
- Question: Does there exist a set of k triangles such that including them implies a unique partition of the remainder of the graph?

Note that if a tripartite graph has a triangle partition, the graph must be uniform. Therefore, we assume without loss of generality that this is the case.

The following proof is a sketch and is under revision.

Theorem 6. FCP *Triangle Partition is* FCP*-complete.*

Proof. We use the reduction from [**?**], applying the slight modifications made in [**?**] so that the graph is tripartite, additionally we apply another modification which makes us reduce the problem from 1-in-3SAT. We refer the reader to [**?**], [**?**] for questions about notation.

We define f to map instances to instances. Do the same reduction as in [**?**], but we make some additional associations. In particular, if variable x appears in clause c as x, then [**?**] associates an F-patch in x to an F-patch in c_1. Additionally, we associate a T-patch in x to a T-patch in c_1. By associating F-patches, we guarantee that one of x and c_1 must be a T-partition. By adding the association between a T-patch in x and a T-patch in c_1, we guarantee that one must be an F-partition. This means that if x is a T-partition, then c_1 is an F-partition, and if x is an F-partition, then c_1 is a T-partition.

We pick the center triangle of an F-patch and a center triangle of a T-patch in each variable gadget. This will comprise the special clues for Triangle Partition. The mapping h will be the following. The center triangle of an F-patch of a variable will map to assigning that variable to true. The center triangle of a T-patch of a variable will map to assign that variable to false.

Now we give the algorithm M. Note that if you know any triangle in the partition, then that triangle necesarily implies a partition in that graph, which will be connected to a partition in a variable, which means we can replace that triangle, with our special triangle in the variable. So algorithm M determines the partition of the clause gadget or the variable gadget and using the fact that an partitions of clauses connect to partitions of variables, we can replace the arbritrary triangles in the clues with the special triangles in the variables.

Since clues for Triangle Partition map to clues of 1-in-3SAT, FCP Triangle Partition is FCP-hard. □

3.4 FCP LatinSquares

In Latin Squares we are given a $n \times n$ grid, some of whose entries have been filled in with numbers from 1 to n. The goal of the game is to fill the rest of the grid with numbers from 1 to n while enforcing that no row or column repeats a number.

It is known that such problem is NP-complete. We present the problem FCP Latin Squares in our framework.

- Instance: A partially filled latin square, along with a parameter k.
- Question: Does there exists set of at most k entries such that pre-setting them implies a unique solution?

Theorem 7. FCP *Latin Squares is* FCP*-complete.*

Proof. We do this via reduction from FCP TrianglePartition. Given a tripartite graph G, we first check whether or not it is uniform. If it is not uniform, then there does not exist a triangle partition. If it is uniform, we can write down a Latin framework $LF(G; 2n, 2n, 2n)$ in polynomial time [**?**]. Since the Latin

framework is constructed so that G is the defect of $LF(G; 2n, 2n, 2n)$, there is a triangle partition of G if and only if we can complete the partial Latin Square $LF(G; 2n, 2n, 2n)$. This completes the proof.

We claim that the above proofs give us an FCP-hardness result for completing a partial Latin Square. To see this, note that the triangle partitions of a defect graph $G(P)$ are in one to one correspondence with the solutions to the Latin Square P. □

3.5 FCP Sudoku

We first introduce the problem which motivated this study. In Sudoku puzzles, we are given a $n^2 \times n^2$ grid consisting of n^2 $n \times n$ bolded blocks, some of whose entries have been filled in with number from 1 to n^2. Often in the literature, n is referred to as the order of the puzzle. The goal of the game is to fill in the rest of the grid with numbers from 1 to n^2 while enforcing that no row, column or bolded square repeats a number.

- Instance: A partially filled Sudoku puzzle, along with a parameter k.
- Question: Does there exists at set of at most k entries such that presetting them yield a unique Sudoku puzzle?

Note that when the instance is an empty Sudoku puzzle of order 3, the question becomes exactly the question of determining the size of the valid Sudoku puzzles. This particular instance of the problem can solve the problem that [**?**] computed.

Theorem 8. FCP *Sudoku is* FCP*-complete*

We will show the proof from [**?**]. For proofs of the following two propositions, refer to the paper.

Proposition 4. *Let S_0 be defined as*

$$S_0(i,j) = ((i \mod n)n + \lfloor i/n \rfloor + j) \mod n^2.$$

Then S_0 represents a solution to an order n Number Place.

Proposition 5. *Let S be a Number Place puzzle of order n such that*

$$S(i,j) = \begin{cases} \perp & : (i,j) \in B \\ S_0(i,j) & : \textit{otherwise} \end{cases}$$

where $B = \{(i,j) | i < n \text{ and } (j \equiv 0) \mod n\}$. Then a square S' obtained by filling in the blanks of S is a solution to S if and only if

- *For any $(i,j) \in B$, $S'(i,j) \equiv 0 \mod n$*
- *A square L defined by $L(i, j/n) = S'(i,j)/n$ for all $(i,j) \in B$ is a Latin Square.*

Proof. We will show an FCP-reduction from FCP Latin Squares and argue that it can be done in polynomial time.

Suppose we are given a Latin Square L of order n. We will construct a Sudoku instance of order n as follows:

$$S(i,j) = \begin{cases} \perp & : (i,j) \in B, L(i,j/n) = \perp \\ L(i,j/n)n & : (i,j) \in B, L(i,j/n) \neq \perp \\ S_0(i,j) & : \text{otherwise} \end{cases}$$

This construction can be done in polynomial time. In addition, from our previous analysis we know that any solution of L has a unique corresponding solution of S. Therefore, we get a polynomial time FCP-reduction.

0		
	1	
		2

Fig. 3. Example of partial Latin Square of order 3.

0	1	2		4	5		7	8
	4	5	3	7	8		1	2
	7	8		1	2	6	4	5
1	2	3	4	5	6	7	8	0
4	5	6	7	8	0	1	2	3
7	8	0	1	2	3	4	5	6
2	3	1	5	6	4	8	0	7
5	6	4	8	0	7	2	3	1
8	0	7	2	3	1	5	6	4

Fig. 4. Example of corresponding partial Number Place on a board of order 3.

□

4 Relationship to Other Complexity Classes.

In this section, we consider how FCP fits in relation to the well-known complexity classes, NP, coNP, $\mathsf{NP}^{\mathsf{NP}}$, PSPACE. Recall that $\mathsf{NP}^{\mathsf{NP}}$ refers to the class of

problems solvable in nondeterministic polynomial time with an NP-oracle. We give a number of different containments, but it remains to be shown if any of these containments are strict.

Proposition 6. $\mathsf{NP} \subset \mathsf{FCP}$

Proof. If a problem is in NP, then there exists an NTM which decides in time at most $p(n)$, where p is a polynomial and n is the size of the input. Thus, we can solve any problem in NP with the following equivalent FCP question: is there a partial (possibly full) certificate of size at most $p(n)$? □

Corollary 9. *If a problem is* FCP*-hard, it is also* NP*-hard.*

Proposition 7. *If a problem is* FCP*-hard, it is also* coNP*-hard.*

Proof. Consider the problem of $\mathsf{UNIQUE-SAT}$, which gives a Boolean formula and asks if there is exactly one solution. This problem is known to be coNP-hard [**?**]. We can solve $\mathsf{UNIQUE-SAT}$ by asking the following FCP question: given a Boolean formula, is there a partial certificate of size at most 0? If and only if such a certificate exists, then by definition of the Boolean formula is uniquely satisfiable. □

We switch our focus now to the problem of FCP SAT, an FCP-complete problem.

Proposition 8. FCP *is contained in* $\mathsf{NP}^{\mathsf{NP}}$.

Proof. We do this by providing an algorithm that solves FCP SAT with an NTM with a SAT oracle.

The algorithm we give will rely on the following observations.

Observation 1: A Boolean formula ϕ has a partial assignment of k variables which yield a unique solution if and only if for every variable outside the partial assignment, one assignment of this variable yields a satisfiable formula, and the other does not. This observation holds due to the fact that there is only one accepting branch.

Observation 2: A Boolean formula ϕ with a partial assignment of k variables is *not* uniquely satisfiable if and only if there exists a variable outside of the partial assignment for which both truth assignments of this variable give a satisfiable Boolean formula. This observation holds because this situation corresponds exactly with cases where we are not on a unique accepting branch.

To complete the proof, we simply give the algorithm:

FCP SAT: Instance: ϕ formula with variables $\{x_i\}$ and variable k.

```
1  for i = 1, ..., k
2      Non-deterministically pick a variable x_i, without repetition
3      Non-deterministically pick an assignment
4  Iterate through all remaining variables
5      if a variable is such that both assignments give a satisfiable formula
6          reject
7      else assign that variable
8  Once you have assigned all variables, accept.
```

□

5 Conclusion

We have formalized the notion of what it means for a problem to be uniquely solvable after giving the fewest possible number of clues. This gave rise to the FCP class. We presented FPC versions of some classical NP problems and showed they were complete for this class. In particular, this means FCP Sudoku is NP-hard, but is unlikely to be PSPACE-hard. However, the instances of FCP Sudoku which make it FCP-complete are very far from empty. In fact, just by looking at the last reduction from FCP Latin Squares, a Sudoku of dimensions $n^2 \times n^2$ which is a hard instance, at most n^2 of the n^4 squares at empty. It would be interesting to see if the empty instances of Sudoku are still complete for FCP. This would be exactly the problem [**?**] solved for $n = 3$.

Further work in the field includes looking at other related results and adapting our framework to a the more general setting. The authors also think that it is important to determine where this class fits in the complexity zoo. In particular, there seem to be connections with the class $\#P$. A stricter notion of a reduction might also be required. Finally, it would be interesting to find problems in P such that their SPC versions are as hard for the class, or NP-hard problems such that their SPC versions are not hard for the class.

References

1. McGuire, G., Tugemann, B., Civario, G.: There is no 16-clue sudoku: Solving the sudoku minimum number of clues problem. arXiv preprint arXiv:1201.0749 (2012)
2. Ghandehari, M., Hatami, H., Mahmoodian, E.: On the size of the minimum critical set of a latin square. Discrete Mathematics **293**(13) (2005) 121 – 127 19th British Combinatorial Conference 19th British Combinatorial Conference.
3. Cooper, J., Kirkpatrick, A.: Critical sets for sudoku and general graph colorings. Discrete Math. **315-316** (February 2014) 112–119
4. Royle, G.: Good at sudoku? heres some youll never complete
5. Garey, M.R., Johnson, D.S.: Computers and Intractability; A Guide to the Theory of NP-Completeness. W. H. Freeman & Co., New York, NY, USA (1990)
6. Holyer, I.: The np-completeness of some edge-partition problems. SIAM Journal on Computing **10**(4) (1981) 713–717
7. Colbourn, C.J.: The complexity of completing partial latin squares. Discrete Applied Mathematics **8**(1) (1984) 25–30
8. Takayuki, Y., Takahiro, S.: Complexity and completeness of finding another solution and its application to puzzles. IEICE transactions on fundamentals of electronics, communications and computer sciences **86**(5) (2003) 1052–1060
9. Blass, A., Gurevich, Y.: On the unique satisfiability problem. Information and Control **55**(1) (1982) 80–88