Extremal Combinatorics, Iterated Pigeonhole Arguments and Generalizations of PPP

Amol Pasarkar
Columbia University, New York, NY, USA

Christos Papadimitriou
Columbia University, New York, NY, USA

Mihalis Yannakakis
Columbia University, New York, NY, USA

Abstract

We study the complexity of computational problems arising from existence theorems in extremal combinatorics. For some of these problems, a solution is guaranteed to exist based on an *iterated* application of the Pigeonhole Principle. This results in the definition of a new complexity class within TFNP, which we call PLC (for "polynomial long choice"). PLC includes all of PPP, as well as numerous previously unclassified total problems, including search problems related to Ramsey's theorem, the Sunflower theorem, the Erdős-Ko-Rado lemma, and König's lemma. Whether the first two of these four problems are PLC-complete is an important open question which we pursue; in contrast, we show that the latter two are PPP-complete. Finally, we reframe PPP as an optimization problem, and define a hierarchy of such problems related to Turàn's theorem.

2012 ACM Subject Classification Theory of computation → Complexity classes

Keywords and phrases Total Complexity, Extremal Combinatorics, Pigeonhole Principle

Digital Object Identifier 10.4230/LIPIcs.ITCS.2023.88

Related Version *Full Version*: `https://arxiv.org/abs/2209.07625`

Funding Research partially supported by NSF Grants CCF-1763970, CCF-2212233.

1 Introduction

The complexity class TFNP [13] captures a wide variety of search problems which are believed to lie between P and NP – in fact, almost all problems in NP not yet known to be in P, or close to it, appear to belong to this class. The "TF" in TFNP indicates that it is a class of total function problems – computational search problems which are mathematically guaranteed to have a solution on all instances – while the letters "NP" in TFNP signify that solutions are polynomially checkable. Two things make TFNP interesting: First, it is a microcosm of many complexity classes, each of which is identified with a non-constructive combinatorial lemma used in the proof of totality [14, 9, 6]. Second, it contains – almost by its definition – many problems that are of great interest in Cryptography. The most obvious and best known example is of course factoring, but many other computational problems of cryptographic interest lie in the subclass **PPP** [8, 15] whose existence lemma is the Pigeonhole Principle: *if there are 2^n pigeons to be placed into $2^n - 1$ pigeonholes, there must exist a pigeonhole with at least two pigeons.*

In this paper we study the complexity of total search problems in the important field of extremal combinatorics [2, 7, 10]. Notice that the Pigeonhole Principle itself can be seen as an argument in extremal combinatorics: *"If a combinatorial object of a certain kind (here, a set mapped to $[N]$) is large enough, it must contain a certain substructure (here a collision)."* This leads one to ask: Are then the computational problems coming from extremal combinatorics in PPP? We point out that the combinatorial lemma underlying many such problems is a counting argument which iteratively uses a form of the Pigeonhole Principle. Accordingly, we introduce a new complexity class, which we call "Polynomial Long Choice" (PLC) capturing the complexity of the iterated Pigeonhole Principle.

To understand the generic problem in this class, consider the following two-player game. Player 1 seeks to construct a long sequence of pigeons and Player 2 tries to make this impossible. We start with 2^n pigeons. At each stage, Player 1 can pick (without replacement) a single pigeon from the remaining available pigeons to add to the long sequence. Once Player 1 has made a move, Player 2 can then partition the remaining pigeons into two groups. Next, Player 1 will pick a pigeon from one of these groups to add to the sequence, and the pigeons from the other group will immediately be removed from the game. Player 1 wins if a sequence of pigeons of length $n+1$ is constructed, otherwise Player 2 wins. It is easy to see that Player 1 has a winning strategy in this game (pick any pigeon from the larger group in every iteration) – and this is the existence lemma defining PLC. To make this into a computational problem, which we call Long Choice, we equip Player 2 with a suite of polynomial-time algorithms, one for each stage of the game (we give the precise definitions below); PLC is the class of all search problems reduced to Long Choice.

We show that Long Choice is PPP-hard, and hence PLC contains PPP. But is this containment strict? And even if not, why is Long Choice, defined by an iterative application of pigeonhole arguments, not contained in $\mathrm{P}^{\mathrm{PPP}}$? The difficulty is this: The winning strategy requires that Player 1 estimates the majority correctly at each stage and also successfully selects a pigeon from this majority. PPP does not seem to support both of these challenges.

It turns out that PLC contains a host of natural problems embodying important theorems in extremal combinatorics, first and foremost Ramsey's, but also the sunflower theorem, the Erdős-Ko-Rado lemma and König's Lemma. The latter two, however, can be shown to be PPP-complete. We also study problems associated with another classical result in extremal combinatorics, namely Mantel's Theorem ("a graph with N nodes and more than $N^2/4$ edges cannot be triangle-free") and Turan's theorem (the generalization to k-clique-free graphs). We identify an infinite hierarchy of problems related to these theorems, each of which is PPP-hard. This generalization of PPP seems substantially different from PLC, in the sense that the source of computational hardness arises from information-theoretic reasons, namely, an inefficient encoding of the search problem.

But iterating the Pigeonhole Principle can take us even higher: consider the dual problem which can be called Short Choice: Suppose that the above game had $2^n - 2$ pigeons, and that Player 1 now wants to terminate the game as soon as possible and Player 2 wants the opposite, to extend it; the game terminates when one of the groups created by Player 2 is empty. It is easy to see that Player 1 cannot be forced to make more than $n-1$ moves, by choosing in every iteration to continue in the smaller of the two groups. Now call this problem Short Choice; it is certainly total, but it does not seem to belong to NP (how does one verify that there is no pigeon left that is consistent with all the previous choices of Player 1?). This is reminiscent of the *empty pigeonhole principle* recently explored in [11] and the class PEPP belonging in $TF\Sigma_2P$ and not believed to be inside TFNP (or NP).

We show that Short Choice is a PEPP-hard problem that defines a new subclass of $TF\Sigma_2P$.

2 Long Choice

We start by recalling the definition of the class PPP. We first define the problem Collision to be the following: we are given a Boolean circuit C with n input bits and n output bits, and we seek either (a) an input x such that $C(x) = 0^n$, or (b) a *collision*, two distinct inputs

$x \neq y$ such that $C(x) = C(y)$. The class PPP is the set of all search problems that reduce to COLLISION. For the *weak* version of PPP, denoted PWPP, the circuit has n inputs and $n-1$ outputs, and a collision is sought. It is known that $n-2$ or fewer outputs, down to n^δ for any $\delta > 0$, yield the same class [12].

Let us next define the search problem RAMSEY, motivated by one of the most influential theorems in all of combinatorics: Given a graph with 2^{2n} nodes, represented by a circuit with $4n$ input bits and one output bit, we seek either a clique with n nodes, or an independent set with n nodes. The nodes are represented by $2n$-bit strings and the circuit specifies the edge relation of the graph. The well known proof of Ramsey's theorem proceeds by constructing a sequence of $2n$ nodes, where: the first node is arbitrary; and the next node is selected from the available nodes to belong in the *majority,* either adjacent or nonadjacent to the last node, whichever group is larger. In addition, the smaller group becomes unavailable. Since we start with 2^{2n} nodes and the minority becomes unavailable at each step, it is clear that a sequence of $2n$ nodes can be selected, and therein we will find either an independent set or a clique with n nodes.

This proof inspires the key definition of this paper:

▶ **Definition 1.** *The* LONG CHOICE *problem is the following: There is a universe U of 2^n objects, represented by the 2^n n-bit strings. We are given a sequence of $n-1$ circuits $P_0, \ldots, P_{n-2}$, each of poly(n) size, such that P_i has $(i+2)n$ input bits and one output bit; circuit P_i represents a* predicate *on $i+2$ objects. We are asked to find a sequence of $n+1$ distinct objects $a_0, \ldots, a_n$, with the following property: for each i in $[0, \ldots, n-2]$, $P_i(a_0, \ldots, a_i, a_j)$ is the same for all $j > i$.*

▶ **Theorem 2.** LONG CHOICE *is a total problem in TFNP.*

Proof. Our construction is inspired by the proof of Ramsey's Theorem as well as the two-player game which we described in the introduction.

First, we pick an arbitrary element a_0 in the universe. Then, we partition the remaining elements a_i into two categories, based on the value of $P_0(a_0, a_i)$. Since there are $2^n - 1$ elements being partitioned, the majority of this partition must have at least 2^{n-1} elements. We select an arbitrary element a_1 from this majority and discard all elements from the minority.

We then continue this procedure: we partition the remaining elements x based on the value of $P_1(a_0, a_1, x)$, and we pick an arbitrary element a_2 from the majority of this new partition.

We can continue partitioning elements in this fashion and picking elements from the majority until we arrive at a complete Long Choice certificate. This proves the totality of the problem. Membership in TFNP follows from the fact that a candidate certificate sequence $a_0, \ldots a_n$ can be checked easily in polynomial time. ◀

Critically, LONG CHOICE is also PPP-hard. To see why, consider the two-player game from the Introduction which characterizes LONG CHOICE. In this game, player 2 can behave (i.e., the predicates P_i can be specified) in a way that guarantees that the only way player 1 wins is by finding a certificate to a PPP-complete problem COLLISION. Consider an instance of COLLISION, given by a circuit, C, which maps n-bit strings to n-bit strings. Player 1 starts the game with 2^n objects (the domain of the circuit). At each round, player 1 picks an element from the remaining set of objects. If at any round, player 1's choices so far contain a certificate to COLLISION (that is, one element is a zero element or a pair of elements collide under C), then player 2 stops partitioning the remaining elements. That is, player 2, for the rest of the game, places all remaining elements into the same side of the partition, guaranteeing a path to victory for player 1.

In each round, player 2 considers the "vacant spots" in the range of C: the nonzero values of the range that do NOT contain the image of player 1's choices. At round 1 of the game, player 1 makes some choice, call it a_0. If $C(a_0) = 0$, then player 1 has found a certificate, and we are done. If $C(a_1)$ is positive, then there are $2^n - 2$ vacant spots left. Player 2 splits the set of vacant spots into two even halves (there are many ways to do this, one way is for player 2 to specify a constant, k and declare that all vacant spots less than or equal to k belong to one half, and the vacant spots greater than k belong to the other).

Regardless of what value player 1 chooses for a_1 in round 2, all subsequent elements must belong to the same subgroup of $C(a_1)$. After $C(a_1)$ is chosen, the number of vacant spots in this subgroup is $2^{n-1} - 2$.

The game continues in this fashion, with player 2 always taking note of the remaining available vacant spots, and splitting this set into 2 even groups. In general, after the i-th round, there will be at most $2^{n-i+1} - 2$ vacant spots left. Therefore, after the n-th round, assuming no certificate has yet been found, there will be $2^1 - 2 = 0$ vacant spots. Therefore, the $(n + 1)$-th choice player 1 makes must provide a collision (or zero element). We provide a complete proof of this result in the appendix.

▶ **Theorem 3.** LONG CHOICE *is PPP-hard.*

Several problems reduce to simplified cases of Long Choice where the predicates have fixed arity. Define UNARY LONG CHOICE to be the version of Long Choice where every predicate P_i depends only on its last argument, i.e., $P_i(a_0, \ldots, a_i, x) = P_i(x)$. Define BINARY LONG CHOICE to be the version of Long Choice where every predicate P_i depends only on two of its arguments, the last argument x and one of the previous a_k, i.e. $P_i(a_0, \ldots, a_i, x) = P_i(a_k, x)$ for some $k \leq i$.

It is easy to see that PWPP reduces to UNARY LONG CHOICE: Given a circuit C for PWPP with n input bits and $n - 1$ output bits, define $P_i(x)$ to be the $(i + 1)$-th bit of $C(x)$. Then in any valid certificate $a_0, \ldots, a_{n-1}, a_n$ for this instance of UNARY LONG CHOICE, we must have $C(a_{n-1}) = C(a_n)$.

▶ **Theorem 4.** *PWPP reduces to* UNARY LONG CHOICE.

In the next section we will see that Ramsey problems reduce to BINARY LONG CHOICE.

In the opposite direction, we can make the Long Choice problem harder by requiring the elements a_i in the certificate to satisfy additional conditions, while still preserving the totality of the problem; for example we can require the a_i to satisfy a given total order. In the proof of the totality of LONG CHOICE, we partition in each step the currently available set and pick an arbitrary element from the majority. We can instead pick a specific element, e.g. the smallest element under the given ordering. We call this generalization LONG CHOICE WITH ORDER; see the appendix for a formal definition.

3 Long Choice and r-Color Ramsey

Ramsey's theorem for multi-colored graphs states that for every number $r \geq 2$ of colors and every integer $n \geq 2$, there is a number $R(r, n)$ such that for every r-coloring of the edges of the complete graph on $R(r, n)$ nodes there is a monochromatic clique with n nodes. The standard Ramsey theorem corresponds to the case of $r = 2$ colors. A simple proof of the multi-colored Ramsey theorem uses the same type of iterative process as the $r = 2$ case, except that in every step we partition the set of available nodes into r groups instead of 2[1]; as before, we pick the largest group to continue the process. The bound on $R(r,n)$ from this simple proof is $R(r,n) \leq r^{rn}$. Obtaining better upper and lower bounds on $R(r,n)$ for $r=2$ and for general r has been (and continues to be) the subject of a long line of intense research effort.

In the computational version of the problem, denoted (r,n)-RAMSEY, we are given the r-coloring of an exponentially large complete graph, which is specified via a poly-size circuit C, and are asked to find a monochromatic clique of size n. The RAMSEY problem of the last section is equivalent to $(2,n)$-RAMSEY. The number r of colors in general need not be fixed, it could be a function of n. Assume for simplicity that r is power of 2 (otherwise, replace $\log r$ in the following by $\lceil \log r \rceil$). Every node of the complete graph is represented by a unique $rn\log r$-bit string, and the given circuit C takes as input two $rn\log r$-bit strings (two nodes u,v) and outputs a $\log r$-bit string (the color of the edge (u,v)).

▶ **Theorem 5.** *(r,n)-RAMSEY is in PLC for all r,n. In particular, (r,n)-RAMSEY reduces to BINARY LONG CHOICE.*

Proof Sketch. We describe first the proof for the case $r=2$, which simply follows the existence proof sketched in the previous section: Given a 2-colored complete graph on 2^{2n} nodes, specified by a given circuit C, define the predicate P_i for each i, to map any sequence $a_0,\dots,a_i,x$ of nodes to the color, 0 or 1, of the edge (a_i,x). Note that P_i depends only on the last two arguments a_i,x. Consider a valid certificate $a_0,\dots,a_{2n}$ of this instance of Binary Long Choice. For each $i=0,\dots,2n-2$, all edges (a_i,a_j) for $j>i$ must have the same color, 0 or 1; assign this color to node a_i. At least n of the $2n-1$ nodes $a_0,\dots a_{2n-2}$ are assigned the same color. These nodes induce a monochromatic clique of size n.

In the case of general r, given an r-colored complete graph on $2^{rn\log r}$ nodes, define each function P_i as follows. Let $k = (\lfloor i/\log r \rfloor)\log r$, i.e, k is the greatest multiple of $\log r$ that is $\leq i$. Set $P_i(a_0,\dots,a_i,x)$ to be the $(i-k+1)$-th bit of the color of the edge (a_k,x). Note that again all these predicates depend only on two arguments, a_k and x.

Consider a valid certificate $a_0,\dots,a_{rn\log r}$ of this instance of Long Choice, and let $b_i = a_{i\log r}$ for each i. From the construction of the Long Choice instance, it is easy to see that, for each $i=0,\dots,r(n-1)$, all edges (b_i,b_j) for $j>i$ must have the same color: note that the t-th bit of the color of edge (b_i,b_j), for all $j>i$, is the value of $P_l(a_0,\dots,a_l,b_j) = P_l(a_0,\dots,a_l,a_{l+1})$ for $l = i\log r + t - 1$. Assign to each node b_i the (common) color of the edges (b_i,b_j), $j>i$. There are $r(n-1)+1$ distinct nodes $b_0,\dots,b_{r(n-1)}$, each assigned one of r colors, therefore at least n of them are assigned the same color. These nodes induce a monochromatic clique of size n. ◀

Finally, consider how Ramsey relates to PWPP. It was shown previously in [12] that there exists a randomized reduction from PWPP to RAMSEY, as well as a reduction from PWPP to the multi-color Ramsey problem. In the full paper, we use properties of metric spaces to provide an alternative deterministic reduction from PWPP to multi-color Ramsey.

4 Sunflowers

An important aspect in extremal combinatorics is the extremal *bound*: how large a system has to be to guarantee that the desired combinatorial structure exists. Often times, the tightest bounds are unknown - improving these bounds is an important research tradition in Combinatorics [1, 3].

[1] Along the same lines, we could extend Long Choice to allow the functions P_i to have a more general range $[r]$ instead of $\{0,1\}$.

However, despite this uncertainty, we can still use weaker extremal bounds to define provably hard TFNP search problems, which we can then relate to the subclasses of TFNP. In this section, we focus on the Sunflower Lemma [10], but this paradigm for reasoning about extremal problems can be applied to many other problems as well.

We begin with some basic definitions.

▶ **Definition 6.** *A k-*set system *is a collection of distinct sets in which every set contains exactly k elements. A collection of distinct sets $S_1, \ldots, S_n$ is a* sunflower *if for all i, j, $S_i \cap S_j$ is the same.*

The Sunflower lemma states that for all positive integers k, s, there is a number $f(k,s)$ such that every k-set system of size $f(k,s)$ contains a sunflower of size s. The lemma was formulated and proved by Erdős and Rado [5] for $f(k,s) = k!(s-1)^{k+1}$, using an inductive proof that applies an iterative pigeonhole argument. The conjecture is that $f(k,s) \leq C^k$ for some constant C that depends only on s. Progress on improving the upper bound on $f(k,s)$ was made recently in [1].

A computational problem based on the Sunflower Lemma was formulated in [12], and shown to be hard on average assuming the existence of collision resistant hash function. Here, we use an even weaker bound to define a problem which we call Naive Sunflower, and relate it to the multi-color Ramsey problem.

▶ **Definition 7** (Naive Sunflower). *We are given a poly(k)-sized circuit*

$$C : \{0,1\}^{k^3 \log k} \mapsto (\{0,1\}^{k^3 \log k})^k$$

which is supposed to specify a family of $2^{k^3 \log k}$ distinct sets of size k over a universe of $2^{k^3 \log k}$ elements (each element is represented by a $k^3 \log k$-bit string). The problem is to find either:

1. *(An error): An index i such that the set (represented by) $C(i)$ contains two identical elements, or find two distinct indices i, j such that the sets $C(i), C(j)$ are equal,or*
2. *A sunflower of size k^2.*

Following the exact same argument presented in [12], this problem is hard on average assuming that Collision Resistant Hash Function families exist.

We will reduce Naive Sunflower to Multi-color Ramsey by using a characterization of large sunflowers as a pairwise equidistant collection of points in a metric space. Given a k-set system F, define the distance $d(A,B)$ between any two sets A, B of F as $d(A,B) = |A \Delta B|/2$, where $A \Delta B = \{A \setminus B\} \cup \{B \setminus A\}$ is their symmetric difference (it has even size since A, B have the same size). The function d is a valid metric. Clearly, any sunflower in F is a set of pairwise equidistant points in this metric.

Conversely, by a result of Deza [4], any collection of at least $k^2 - k + 2$ pairwise equidistant k-sets must form a sunflower. Therefore, in this regime, we can reduce the problem of finding a sunflower of size k^2 to the problem of finding k^2 pairwise equidistant sets. In turn, we can reduce this problem to the multi-color Ramsey problem.

▶ **Theorem 8.** Naive Sunflower *reduces to* $(\sqrt{n}, n)$-Ramsey.

Proof. Consider an instance of Naive Sunflower, given by a circuit C. As above, C is supposed to define a set system consisting of $2^{k^3 \log k}$ sets, where each set contains k elements. The core idea behind our reduction is simple: we construct a graph consisting of $2^{k^3 \log k}$ nodes, where the i-th node u_i corresponds to the set $C(i)$. To color the edges of this graph, we use k colors, given by the numbers $1, 2, \ldots, k$. Every edge (u_i, u_j) is colored as follows.

If $C(i), C(j)$ are distinct k-sets, then assign color $d(C(i), C(j))$ to the edge (u_i, u_j). If one of $C(i), C(j)$ is not a k-set, e.g. contains a duplicate element, or if the sets are equal, then assign color 1 to the edge (u_i, u_j).

Let $n = k^2$. Since the graph has $2^{k^3 \log k} = k^{nk}$ nodes, it contains a monochromatic clique of size $n = k^2$ by Ramsey's Theorem. Let M be any such monochromatic clique of size k^2. If M contains a node u_i such that $C(i)$ is not a k-set, or if it contains two nodes u_i, u_j such that $C(i), C(j)$ are equal sets, then we have a violation for the circuit C. Otherwise, the collection $\{C(i) | u_i \in M\}$ has k^2 pairwise equidistant sets, and thus by Deza's theorem, they form a sunflower. ◀

We have shown earlier that (r, n)-Ramsey is in PLC for any r, n; thus we can conclude:

▶ **Corollary 9.** Naive Sunflower *is in PLC.*

5 Short Choice

As we have seen above, PPP is contained in the class PLC. There is an intuitive reason for this: implicit in any PPP-complete problem is an iterated pigeonhole argument.

The class PEPP, introduced in [11], embodies the dual of the class PPP - an anti-pigeonhole principle: if there are $2^n - 1$ pigeons and 2^n holes, then no matter how the pigeons are placed, there must be an empty hole. PEPP belongs to the class $\mathbf{TF\Sigma_2 P}$, which is believed to lie outside of the class NP.

While the existence proof for Pigeonhole Circuit has a majority argument, the existence proof for Empty has a corresponding "minority" argument. Suppose we are given an instance of the PEPP-complete problem $EMPTY$: we are given a poly(n)-sized circuit $C : [2^n - 1] \mapsto [2^n]$, where the inputs and outputs are all represented concisely using exactly n bits. The challenge here is to find an element j in the range such that there is no i with $C(i) = j$. There must exist a bit c_1 such that the minority of elements in the domain of C map to a n-bit string whose first bit is c_1. This minority has size *at most* $2^{n-1} - 1$. Among the elements in this minority, there must exist a bit c_2 such that the minority of these elements map to a n-bit string whose second element is c_2. This new minority has size *at most* $2^{n-2} - 1$. If we continue this argument n times, we find that there exists a bit string $c_1 \circ c_2 \circ \cdots \circ c_n$ which is not in the image of C.

As discussed earlier, Long Choice is a generalization of PPP that encapsulates the iterated majority arguments. We can define an analogous generalization of the class PEPP that encapsulates the iterated minority argument. For this, we introduce a problem called Short Choice, a problem in $\mathbf{TF\Sigma_2P}$ which is the dual of Long Choice.

▶ **Definition 10** (Subcertificate). *Given a Long Choice problem instance defined by predicate functions $P_0, P_1, \ldots, P_{n-2}$ we call a subcertificate a sequence of distinct elements $a_0, a_1, \ldots, a_k$ ($k \leq n$) which satisfy the* Long Choice *conditions imposed by the predicate functions P_i. That is, for each predicate function $i < k$, we require that $P_i(a_0, \ldots a_i, a_j)$ is the same for all $j > i$.*

▶ **Definition 11** (Problem: Short Choice). *The input is the same as in the* Long Choice *problem, except that the universe U has now $2^n - 2$ objects. As in* Long Choice*, we are given a sequence of $n - 1$ poly(n)-sized circuits, $P_0, \ldots, P_{n-2}$, where each P_i defines a predicate function $P_i : U^{i+2} \mapsto \{0, 1\}$*

The problem is to find a sequence $a_0, a_1, \ldots, a_k$ of at most $n - 1$ distinct objects in U and a bit $c \in \{0, 1\}$ with the property that (1) the sequence $a_0, a_1, \ldots, a_k$ is a subcertificate, and (2) there does not exist any object $a_{k+1} \in U$ that both extends this subcertificate and has $P_k(a_0, a_1, \ldots, a_k, a_{k+1}) = c$.

As in our discussion of LONG CHOICE, we can show that SHORT CHOICE is both total and PEPP-hard. The proof of totality uses a repeated minority argument, in the same way that the proof for Long Choice used a repeated majority argument. And the proof of PEPP-hardness is along similar lines as the PPP-hardness proof for LONG CHOICE. We provide the complete proofs in the appendix.

▶ **Theorem 12.** SHORT CHOICE *is a total problem.*

▶ **Theorem 13.** SHORT CHOICE *is PEPP-hard*

We can define the class PSC (Polynomial Short Choice) to be the class whose complete problem is SHORT CHOICE.

6 König and Erdős-Ko-Rado

In this section we introduce and characterize computational problems associated with two classical theorems in combinatorics.

6.1 König

König's lemma states that in every infinite connected graph with finite degree there is an infinite simple path starting at every node. The lemma is often stated and used for trees: Every infinite (rooted) tree with finite branching has an infinite path (starting at the root). The finite version of the lemma is that every large enough connected graph (or tree) with bounded degree contains a long path. For example, every rooted binary tree with 2^n nodes contains a path of length n. The graph version follows easily from the tree version: Given a connected graph, take a spanning tree of the graph, for example a breadth-first-tree from an arbitrary node.

The standard proof of König's tree lemma (both the infinitary as well as the finitary version) is by the same type of repeated majority argument as in the proof of totality for Long Choice. Starting from the root of the tree, proceed to the child whose subtree contains the largest number of nodes, and repeat the process from there. If the tree is infinite, one of the children must have an infinite subtree (since the degree is finite), thus this process will generate an infinite path. Similarly, in the finite case, every iteration reduces the number of nodes at most by a factor of d (the degree), so the process generates a path of logarithmic length.

Given a (succinctly represented) exponentially large connected graph or tree with bounded degree, e.g. a binary tree, how hard is it to find a long simple path? We formulate this problem below for binary trees, represented through the parent information.

▶ **Definition 14** (Problem: KÖNIG). *We are given a poly(n)-sized circuit, $P : \{0,1\}^n \to \{0,1\}^n \times \{0,1\}$. This circuit is supposed to define a rooted binary tree on 2^n nodes, where each node is encoded by a n-bit string. It does so by defining a parent relation: for a node u, $P(u)$ is an ordered pair (v, b) where v is the (binary encoding of) the parent of u, and b is a bit which indicates whether u is the left or right child of v. If $P(u) = u$, that means that u does not have a parent (it is a root node). In the König problem, we are given P and a root node, r, and are asked to find either a violation (P does not specify a binary tree rooted at r) or find a path of length n. Specifically, return one of the following certificates:*

1. ***Identical children:*** *Return 2 distinct nodes a and b with the property that $P(a) = P(b)$. (That is, they are both left children or right children of the same node).*

2. ***Invalid Root:*** *Return r if $P(r) \neq r$.*
3. ***Non-Unique Root:*** *Return a node $s \neq r$ with the property that $P(s) = s$.*
4. ***Far Away Node:*** *Return a node s with the following property: if we apply the parent operator P to s a total of n times, we do not reach the root r.*
5. ***Long Path:*** *A sequence of $n+1$ nodes $a_0, a_1, \ldots, a_n$ with $a_0 = r$, and the property that $P(a_i) = a_{i-1}$ for all $i \geq 1$. Note that it suffices to provide a_{n-1} as a valid certificate here; the rest of the path can be recovered by applying the parent operator P.*

We refer to the circuit P as the *parent operator.* In cases 1, 2, 3, P does not induce a binary tree. The same is true in case 4, if applying P to s n times produces a repeated node; otherwise, we get a simple path of length n. Case 5 yields a path of length n from the root r.

The proof of König's lemma suggests that the problem should be in PLC. It turns out that KÖNIG is in fact in PPP, and furthermore it is complete; see the appendix for a formal proof.

▶ **Theorem 15.** KÖNIG *is PPP-complete.*

6.2 Erdős-Ko-Rado

The Erdős-Ko-Rado Lemma is one of the foundational results in extremal set theory.

▶ **Theorem 16** (Erdős-Ko-Rado Lemma). *If F is any k-set system over a universe X of size $n > 2k$, and every pair of sets in F has non-empty intersection, then*

$$|F| \leq \binom{n-1}{k-1}$$

Thus, if $|F| > \binom{n-1}{k-1}$ then F must contain two disjoint sets. How hard is it to *find* these two disjoint sets, if F is a succinctly given exponentially large system?

Take the case in which $k = 2$ and the size of the universe is 2^n. In this case, the Erdos-Ko-Rado lemma tell us that the largest possible intersecting set system has size $\binom{2^n-1}{1} = 2^n - 1$. Therefore, given a 2-set system F of size $2^n > 2^n - 1$, then F must contain two disjoint sets.

▶ **Definition 17** (Problem Statement: ERDŐS-KO-RADO). *We are given a poly(n)-sized circuit*

$$F : \{0,1\}^n \mapsto (\{0,1\}^n)^2$$

which is supposed to represent a 2-set system of size 2^n over the universe $X = \{0,1\}^n$. The problem is to find either a violation (F is not a valid encoding) or two disjoint sets of the set system. Specifically, return one of the following certificates:

1. *(Error:) An index i such that $F(i) = (a, a)$ for some $a \in \{0,1\}^n$ (i.e., the set $F(i)$ has two identical elements), or two distinct indices i, j such that the sets (represented by) $F(i), F(j)$ are equal, or*
2. *(Disjoint sets:) Two indices i, j such that the sets (represented by) $F(i), F(j)$ are disjoint.*

By the Erdős-Ko-Rado lemma, one of these conditions must occur, placing the problem in TFNP.

In the Appendix we show the following:

▶ **Theorem 18.** ERDŐS-KO-RADO *is PPP-complete.*

7 Mantel, Turán, and Bad Colorings

In this section, we introduce a new flavor of problems from extremal combinatorics which generalize PPP. This class of problems is related to Mantel's and Turán's theorem and graph colorings. Mantel's theorem states that a triangle-free graph with N nodes has at maximum $\lfloor N^2/4 \rfloor$ edges. The maximum is achieved by a complete bipartite graph with equal or almost equal parts (depending on whether N is even or odd). Turán's theorem answers the generalized question of what is the maximum number of edges in a graph with N nodes that does not contain a $(k+1)$-clique: the maximum is achieved by a complete k-partite graph that has equal or almost equal parts (see [10] for a detailed exposition).

The same quantities answer the easier question of, what is the maximum number of edges of a k-colorable graph on N nodes. A k-colorable graph whose color classes have sizes $x_1, \ldots, x_k$ can have at most $\sum_{i \neq j} x_i x_j$ edges. Since the x_i's are integers that sum to N, it can be shown that the maximum is achieved when they are all equal or almost equal.

These theorems induce corresponding total computational problems: Given (succinctly) an exponential graph with more edges than the above bounds of Mantel or Turán, find a triangle or a $(k+1)$-clique respectively. If we are given in addition a k-coloring of the nodes, find an illegally colored edge. We call these problems respectively MANTEL, k-TURÁN and BAD k-COLORING (MANTEL is just 2-TURÁN). We define below formally the problems as TFNP problems.

▶ **Definition 19** (k-TURÁN). *We are given a poly(n)-sized circuit $E : [\binom{k}{2}(2^n)^2+1] \mapsto [(k2^n)^2]$, which is supposed to represent a graph with $k2^n$ nodes and $\binom{k}{2}(2^n)^2+1$ edges (nodes and indices of edges are encoded by bit-strings of appropriate length as usual); E maps the index of an edge to the two nodes of the edge. The problem is to find either a violation (E is not a valid encoding of the edges of a graph) or a $(k+1)$-clique. Specifically, return one of the following certificates:*

1. *(Error): An index i such that $E(i)$ consists of two identical nodes, or two distinct indices i, j such that $E(i), E(j)$ contain the same two nodes (not necessarily in the same order), or*
2. *($(k+1)$-clique): $\binom{k+1}{2}$ indices which are mapped by E to the edges of a clique on $k+1$ nodes.*

▶ **Definition 20** (BAD k-COLORING). *We are given a poly(n)-sized circuit $E : [\binom{k}{2}(2^n)^2+1] \mapsto [(k2^n)^2]$, which is supposed to represent a graph with $k2^n$ nodes and $\binom{k}{2}(2^n)^2+1$ edges and a poly(n)-sized circuit $C : [k2^n] \mapsto [k]$ which colors the nodes with k colors. The problem is to find either a violation (E is not a valid encoding of the edges of a graph) or an edge whose nodes have the same color. Specifically, return one of the following certificates:*

1. *(Error): An index i such that $E(i)$ consists of two identical nodes, or two distinct indices i, j such that $E(i), E(j)$ contain the same two nodes (not necessarily in the same order), or*
2. *(Bad edge): An index i such that $E(i) = (a, b)$ and $C(a) = C(b)$.*

We show the following relations between PPP and these problems (proofs in the full paper):

▶ **Theorem 21.**

1. *PPP reduces to* BAD 2-COLORING.
2. *For all $k \geq 2$,* BAD k-COLORING *reduces to* k-TURÁN. *In particular,* BAD 2-COLORING *reduces to* MANTEL.
3. *For all $k \geq 2$,* BAD k-COLORING *reduces to* BAD $(k+1)$-COLORING.
4. *For all $k \geq 2$, k* TURÁN *reduces to* $(k+1)$-TURÁN.

Thus, we have a hierarchy of problems on top of PPP. The Bad Coloring problems can be viewed as instances of the pigeonhole problem (there is no iteration here), but the mapping is given indirectly and cannot be easily constructed: We can view the indices of the edges as the pigeons and the potential legal edges, i.e. all the pairs of differently colored nodes, as the holes. There are more pigeons than holes, so either two pigeons are mapped to the same hole ($E(i), E(j)$ are the same edge for some pair of indices i, j)), or some pigeon is not mapped to a hole (for some i, $E(i) = (a, a)$ or $E(i) = (a, b)$ with $C(a) = C(b)$; this corresponds to the special 0 value in the PPP problem). The difference with PPP, is that the set of holes (the range of the mapping) is not given a priori explicitly as a set of bit-strings (or integers) as in PPP, but rather it is implied indirectly by the coloring C. As a consequence, even for $k = 2$, we cannot compute easily in polynomial time for example the number h of available holes (h is the product of the sizes of the two color classes), and we cannot compute efficiently an index function mapping each legal pair of nodes (pair (a, b) with $C(a) \neq C(b)$) to an index in $[h]$. In the Turán problems, there is in addition the complication of optimizing over all partitions (colorings) and of seeking a clique rather than a single edge.

Another example along the same lines is the following BAD k-SET COLORING problem: Given (poly(n)-size circuits specifying) a k-coloring C of a set V of $k2^n$ nodes and a family F of $2^{kn} + 1$ k-sets over V, find a k-set in F that is not panchromatic, i.e. two of its elements have the same color, or find two equal sets in F.

▶ **Theorem 22.**

1. *PPP is equivalent to* BAD 1-SET COLORING.
2. *For all* $k \geq 1$, BAD k-SET COLORING *reduces to* BAD $(k+1)$-SET COLORING

These problems form a hierarchy on top of PPP, where again existence of a certificate is guaranteed by (1) the answer to an optimization problem (what is the maximum number of panchromatic k-sets over all k-colorings), and (2) the pigeonhole principle, where however the mapping is not given explicitly, but is defined indirectly in an inefficient manner.

8 Discussion and Future Work

The generalizations of PPP which we explore in this paper seem to give rise to a remarkably rich set of tantalizing open questions. Some examples:

1. Prove black box separations between the problems and classes studied in this paper. For example, prove a separation between PPP and PLC; between PEPP and PSC; between PPP and the hierarchy of Turán and Bad coloring problems.
2. What other natural problems belong to PLC or are PLC-complete? One problem in TFNP that has long evaded classification is BERTRAND-CHEBYSHEV: given a number n, find a prime number between n and $2n$.
3. What is the complexity of finding monochromatic cliques in smaller graphs, whose existence is guaranteed by a century of fascinating improvements of Ramsey's theorem?
4. What natural problems belong to the class PSC (but not to PEPP)?
5. How does PLC relate to problems in cryptography, and specifically lattices?
6. The Bad Coloring hierarchy suggests a novel source of computational hardness: inefficient encoding of objects. What other interesting natural problems share this type of hardness?
7. More generally, what other problems from extremal combinatorics give rise to search problems in TFNP, and how are these problems classified in TFNP subclasses?

References

1 Ryan Alweiss, Shachar Lovett, Kewen Wu, and Jiapeng Zhang. Improved bounds for the sunflower lemma. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2020, pages 624–630, 2020.

2 Bèla Bollobàs. *Extremal Graph Theory*. Dover, 2013.

3 David Conlon and Asaf Ferber. Lower bounds for multicolor ramsey numbers, 2020. `doi:10.48550/arXiv.2009.10458`.

4 Michel Deza and Peter Frankl. Every large set of equidistant (0, +1, −1)-vectors forms a sunflower. *Combinatorica*, 1:225–231, September 1981.

5 P. Erdös and R. Rado. Intersection theorems for systems of sets. *Journal of the London Mathematical Society*, s1-35(1):85–90, 1960.

6 Paul W. Goldberg and Christos H. Papadimitriou. Towards a unified complexity theory of total functions. *Journal of Computer and System Sciences*, 94:167–192, 2018.

7 Ronald L Graham, Bruce L Rothschild, and Joel H Spencer. *Ramsey theory*, volume 20. 'John Wiley & Sons', 1990.

8 Emil Jeřábek. Integer factoring and modular square roots. *Journal of Computer and System Sciences*, 82(2):380–394, 2016.

9 David S. Johnson, Christos H. Papadimitriou, and Mihalis Yannakakis. How easy is local search? *Journal of Computer and System Sciences*, 37(1):79–100, 1988.

10 Stasys Jukna. *Extremal Combinatorics With Applications in Computer Science*. Springer Berlin, 2013.

11 Robert Kleinberg, Oliver Korten, Daniel Mitropolsky, and Christos H. Papadimitriou. Total functions in the polynomial hierarchy. In *12th Innovations in Theoretical Computer Science Conference, ITCS*, volume 185 of *LIPIcs*, pages 44:1–44:18, 2021.

12 Ilan Komargodski, Moni Naor, and Eylon Yogev. White-box vs. black-box complexity of search problems: Ramsey and graph property testing. *J. ACM*, 66(5), July 2019.

13 Nimrod Megiddo and Christos H. Papadimitriou. On total functions, existence theorems and computational complexity. *Theoretical Computer Science*, 81(2):317–324, 1991.

14 Christos H. Papadimitriou. On the complexity of the parity argument and other inefficient proofs of existence. *Journal of Computer and System Sciences*, 48(3):498–532, 1994.

15 Katerina Sotiraki, Manolis Zampetakis, and Giorgos Zirdelis. Ppp-completeness with connections to cryptography. In *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS*, pages 148–158. IEEE Computer Society, 2018.

A Missing material from Section 2 (Long Choice)

We first remark that in the definition of Long Choice, it is unimportant what the initial element a_0 actually is, and whether it is specified or not. Consider a variant of the problem, Constrained Long Choice, where a specific initial element is required. By the proof of Theorem 2, this constrained variant is also a total search problem. In the full paper we prove:

▶ **Proposition 23.** Long Choice *with no initial element is equivalent to* Constrained Long Choice.

▶ **Theorem 3.** Long Choice *is PPP-hard.*

Proof. Suppose that we are given a circuit C_0 mapping n bits to n bits, an instance of the Collision problem. We can view the inputs and outputs of circuits both as n-bit strings or as the equivalent integers in $[0, \ldots, 2^n - 1]$. Given C_0, we define a new circuit, C which maps n-bit strings to n-bit strings. On input a, we define C as follows:

1. $C(a) = 2^n - 1$ (the all-1 string) if $C_0(a) = 0$
2. $C(a) = C_0(a)$ otherwise

By the pigeonhole principle, since C only maps inputs to nonzero values, it must have collisions. Any such collision will either allow us to recover a collision in C_0 or allow us to recover a zero element of C.

We now reduce the problem instance given by circuit C to a Long Choice problem. We begin by defining our set, U, to be the domain of circuit C: the set of n-bit strings. Note that U has 2^n distinct elements, each of which is represented as a unique n-bit string. It now suffices to define $n-1$ predicate functions $P_0, P_1, \dots, P_{n-2}$.

As in the proof of totality for Long Choice, we can think about constructing a certificate by sequentially making the choices $a_0, a_1, \dots, a_n$. Put simply, our predicate functions will classify the elements of U based on their *images* under C. These functions will enforce the following property: subsequent elements of our Long Choice certificate will have images under C which are closer and closer together. More specifically, consider the first three elements of a Long Choice certificate, a_0, a_1, a_2. While $C(a_0)$ and $C(a_1)$ may be more than $2^{n-1}-1$ units apart, predicate function P_0 will enforce the condition that $C(a_1)$ and $C(a_2)$ are within $2^{n-1}-1$ units of each other. Each predicate function will, in effect, enforce similar "closeness" conditions. Ultimately, this will force any Long Choice certificate to have two distinct elements whose images under C are 0 units apart (a collision under C!), as desired.

In the following discussion, we assume all inputs to predicate functions are distinct, because this is required of any valid certificate.

We now explicitly define the predicate functions:

1. $P_i(a_0, \dots, a_i, x) = 1$ if $C(x)$ is in the interval F_i (defined below)
2. $P_i(a_0, \dots, a_i, x) = 0$ if $C(x)$ is *not* in the interval F_i.

To complete the definition, we define the intervals, F_i, which depend on the elements $a_0, \dots, a_i$. Before we do so, we introduce some basic terminology to make this discussion clearer.

1. **Unfilled set:** For any sequence of elements $a_0, \dots, a_i$ and an interval $[p, q]$, the unfilled set of the interval is defined as $\{\{p, p+1, \dots, q\} \setminus \{C(a_0), C(a_1), \dots, C(a_i)\}\}$ For example, if given a sequence of points a_0, a_1, a_2 with $C(a_0) = 0$, $C(a_1) = 2$, $C(a_2) = 1$, the unfilled set for the interval $[0, 5]$ is $\{3, 4, 5\}$.
2. **Indexing an interval:** (Purely for notational convenience) Given an interval $I = [a, b]$, for $k > 0$, define $I[k]$ as the interval containing the smallest k elements of I, and define $I[-k]$ as the interval containing the largest k elements of I. For example, given $I = [1, 4]$, $I[2] = [1, 2]$, the first two integers in the interval, while $I[-3] = [2, 4]$, the largest 3 integers in the interval.

We now define a sequence of intervals $B_0, \dots, B_i, \dots$ and $F_0, \dots, F_i, \dots$. Critically, each interval F_i is a contained in the corresponding interval B_i. We proceed with an inductive definition.

Base Case Definition. For any single element sequence a_0, the corresponding interval B_0 is $[1, 2^n - 1]$. The unfilled set for B_0 has size $2^n - 2$. Consider the value of k that guarantees that $B_0[k]$ has an unfilled set of size $2^{n-1} - 1$. This can be easily computed: $k = 2^{n-1} - 1$ if $C(a_0) > 2^{n-1} - 1$, else $k = 2^{n-1}$. We define $F_0 = B_0[k]$.

Inductive Definition. Suppose that we have a sequence of elements $a_0, \dots, a_i$, and for all $k < i$, B_k and F_k are defined. We first define B_i using the following rules:

1. First, if B_{i-1} is an interval of size 1, then $B_i = F_i = B_{i-1}$

2. Otherwise, if $C(a_i)$ is in F_{i-1}, then $B_i = F_{i-1}$. If $C(a_i)$ is not in F_{i-1}, then $B_i = B_{i-1} \setminus F_{i-1}$.
3. Finally, we define F_i. Let x denote the size of the unfilled set of B_i. Let k be the smallest integer such that $B_i[k]$ has $\lceil \frac{x}{2} \rceil$ unfilled spots. We let $F_i = B_i[k]$.

This completes the definition of the Long Choice problem instance. It remains to prove that a valid certificate for this problem instance allows us to recover a collision under C. Let $a_0, a_1, \ldots, a_n$ be a valid certificate. If there is a collision among these elements, we are done. So assume there is no collision; we will derive a contradiction.

Consider the sequence of set B_i, F_i generated from the sequence $a_0, a_1, \ldots, a_n$. Note that $B_0 \supseteq B_1 \cdots \supseteq B_i \ldots$ and $F_i \subseteq B_i$ for all i. It is easy to show inductively from the construction that the following two properties hold:
1. $\forall i, \forall j \geq i, C(a_j) \in B_i$.
2. $|B_i \setminus \{C(a_0), \ldots, C(a_i)\}| = 2^{n-i} - 2$ for all $i \leq n-2$.

The basis case ($i = 0$) for both properties is trivial. The induction step for property 1 follows from the fact that $P_{i-1}(a_0, \ldots, a_{i-1}, a_j)$ has the same value for all $j \geq i$, hence either all these $C(a_j)$ are in F_{i-1} or they are all not in F_{i-1} and thus they are in $B_{i-1} \setminus F_{i-1}$ (because they are all in B_{i-1} by the induction hypothesis). It follows from the definition of B_i that they are all in B_i.

For the induction step of property 2 note that the induction hypothesis

$$|B_{i-1} \setminus \{C(a_0), \ldots, C(a_{i-1})\}| = 2^{n-i+1} - 2$$

implies that both F_{i-1} and $B_{i-1} \setminus F_{i-1}$ have $2^{n-i} - 1$ unfilled spots. Since $C(a_i) \in B_i$ and a_i does not collide with any earlier a_j, it follows that $|B_i \setminus \{C(a_0), \ldots, C(a_i)\}| = 2^{n-i} - 2$.

From property 2, B_{n-2} has an unfilled set of size $2^2 - 2 = 2$. The interval F_{n-1} is, by definition, constructed such that F_{n-1} and $B_{n-2} \setminus F_{n-1}$ both have 1 unfilled spot. Critically, based on the definition of P_{n-2}, we know that $C(a_{n-1})$ and $C(a_n)$ must both belong to F_{n-1} or both belong to $B_{n-2} \setminus F_{n-1}$. Therefore, $C(a_{n-1})$ and/or $C(a_n)$ must collide with each other or with another element in the sequence, a contradiction. ◀

Finally, we define Long Choice with Order formally:

▶ **Definition 24** (LONG CHOICE WITH ORDER). *Consider a set U of 2^n objects, each represented by a unique n-bit string. We are given a sequence of $n-1$ predicate functions, $P_0, \ldots, P_{n-2}$ represented by poly(n)-size circuits. Predicate function P_i has arity $i+2$:*

$$P_i : U^{i+2} \mapsto \{0, 1\}$$

We are also given a function F, which purportedly defines a strict total order over the above set U. The function F is also represented by a poly(n)-size circuit:

$$F : U^2 \mapsto \{0, 1\}$$

Given two distinct inputs a_x, a_y, $F(a_x, a_y) = 0$ indicates that $a_x < a_y$ in this total ordering. $F(a_x, a_y) = 1$ indicates that $a_x > a_y$. The problem is to find any of the following:
1. ***Monotone certificate:*** *A monotone increasing sequence of $n+1$ distinct objects $a_0, \ldots, a_n$ in U, with the following property: for each i in $[0, \ldots, n-2]$, $P_i(a_0, \ldots, a_i, a_j)$ is the same for all $j > i$.*
2. ***Order Violation:*** *A set of 3 distinct objects a_x, a_y, a_z which violate the transitivity property of total orders.*

B Missing material from Section 5 (Short Choice)

▶ **Theorem 12.** Short Choice *is a total problem.*

Proof. Let U denote the set of $2^n - 2$ objects in the universe of this Short Choice problem instance. We will construct a certificate, given by a subcertificate $a_0, a_1, \dots, a_k$, $(k \leq n-2)$ and a bit c. In order to construct this sequence, we will also define a sequence of nonempty sets $U_0 \supset U_1 \supset \cdots \supset U_k$ with the following properties:

1. $U_0 = U$
2. $|U_k| \leq 2^{n-k} - 2$ for all k
3. $a_k \in U_k$ for all k *and* whenever $k \geq 1$, $a_0, \dots, a_{k-1} \notin U_k$.
4. For every $j \leq k$, an element x extends the subcertificate $a_0, a_1, \dots, a_j$ if and only if $x \in U_j \setminus \{a_j\}$.
5. For all $x \in U_{j+1}$, $P_j(a_0, \dots, a_j, x)$ is the same.

As a base case, we begin by defining $U_0 := U$, and we pick an arbitrary element $a_0 \in U_0$. Note that the base case of our construction so far satisfies properties 1 and 2: $U_0 = U$, $|U_0| = |U| \leq 2^{n-0} - 2$. Property 3 holds because $a_0 \in U_0$ (the second condition in property 3 is vacuously true here). Property 4 holds trivially, since any element $x \neq a_0$ must belong to $U_0 \setminus a_0$.

We now define U_1 in such a way that our base case satisfies property 5 as well. We partition the elements $x \in \{U_0 \setminus \{a_0\}\}$ in two groups based on the value $P_0(a_0, x)$. We know that $|U_0 \setminus \{a_0\}| = 2^n - 3$, and since this set is being partitioned into two disjoint sets, the minority must have size at most $2^{n-1} - 2$. If the minority is nonempty, we define U_1 to be the minority (which guarantees condition 5 holds). We also define a_1 as an arbitary element of U_1. Otherwise, if the minority is empty, this means that $P_0(a_0, x)$ takes on a constant value (call it b) for all $x \in U \setminus \{a_0\}$. This in turn means that there is no value x with $P_0(a_0, x) = \neg b$ (the opposite of bit b). We can thus return a_0 along with the bit $\neg b$ as a valid certificate to the Short Choice problem. This completes the base case.

Now, suppose for some $0 \leq j \leq k$ we have defined the nonempty sets $U_0 \supset U_1 \supset \cdots \supset U_j$ and the subsequence $a_0, a_1, \dots, a_j$ in such a way that they satisfy the above properties. We can first conclude based on properties 3 and 5 that the sequence $a_0, \dots, a_j$ is a valid subcertificate. Furthermore, an element x extends this subcertificate if and only if it belongs to $U_j \setminus \{a_j\}$, according to property 4.

To continue our inductive construction, we partition the elements $x \in \{U_j \setminus \{a_j\}\}$ based on the value of $P_j(a_0, \dots, a_j, x)$. If one side of this partition is empty, that means that there exists a bit $c \in \{0, 1\}$ such that there are no elements $x \in \{U_j \setminus \{a_j\}\}$ where $a_0, a_1, \dots, x$ is a subcertificate and $P_j(a_0, \dots, a_j, x) = c$. In this case, we are done: the sequence $a_0, \dots, a_j$ along with the bit value c, serves as a certificate to the problem.

Otherwise, both sides of this partition are nonempty. In this case, we can continue the inductive construction: we pick the minority side of the partition of the elements $x \in \{U_j \setminus \{a_j\}\}$. We then define U_{j+1} to be all of the elements on the minority side of the partition, and pick an arbitrary element $a_{j+1} \in U_{j+1}$ to extend the subcertificate. We know by the inductive hypothesis that $|U_j| \leq 2^{n-j} - 2$, and by the same Pigeonhole argument used in the base case, we know that $|U_{j+1}| \leq 2^{n-j} - 2$, as desired. Thus properties 1 and 2 hold. To see why property 3 holds, note that $a_{j+1} \in U_{j+1}$. Furthermore, $a_0, \dots, a_{j-1} \notin U_j \supset U_{j+1}$. Finally, $a_j \notin U_j$, as stated above. To see why property 4 must hold, consider any candidate element a_{j+2} which may extend the Long Choice subcertificate. We know from the inductive hypothesis that a_{j+2} must belong to $U_j \setminus a_j$. Additionally, we must also now have that

$$P_j(a_0, \dots, a_j, a_{j+1}) = P_j(a_0, \dots, a_j, a_{j+2})$$

By the definition of P_j, this means that a_{j+1} and a_{j+2} both belong to U_{j+1}, and since a_{j+1} and a_{j+2} must be distinct in order to be in the same subcertificate, we conclude $a_{j+2} \in U_{j+1} \setminus a_{j+1}$. This argument also shows that property 5 continues to hold as well.

Note that our construction can only proceed until $j = n-2$. To see why, suppose that $j = n-2$. Then, by our inductive hypotheses, $|U_j| \leq 2^{n-(n-2)} - 2 = 2$. Furthermore, since $a_j \in U_j$ by our assumption, we know that $U_j \setminus \{a_j\}$ has at most 1 other element. Denote this element (if it even exists) by a_{n-1}, and consider the value $c_{bad} = P_{n-2}(a_0, \ldots, a_{n-2}, a_{n-1})$. It is clear that if we consider c to be the opposite bit of c_{bad}, there are no elements x which extend this subcertificate with $P_{n-2}(a_0, \ldots, a_{n-2}, x) = c$. Thus, we can return $a_0, \ldots, a_{n-2}$ and c and we are done.

Therefore, our construction is guaranteed to terminate with a subcertificate $a_0, \ldots, a_j$ of the appropriate length, as well as a bit c, which provide us with a solution to the Short Choice problem. ◀

▶ **Theorem 13.** SHORT CHOICE *is PEPP-hard*

Proof. Consider an instance of the PEPP-complete problem, EMPTY, which is given by a poly(n)-sized circuit $C : [2^n - 2] \mapsto [2^n - 1]$, where the challenge is to find an element, $e \in [2^n - 1]$ such that for all $i \in [2^n - 2]$, $C(i) \neq e$.

We now define a SHORT CHOICE instance whose solution allows us to recover a solution to the EMPTY problem.

The universe, U, of this instance consists of the elements in the set $[2^n - 2]$ (the domain of C). It now suffices to define a sequence of predicate functions, $P_0, \ldots, P_{n-2}$.

Each predicate function P_i takes as input the distinct elements $a_0, a_1, \ldots, a_i, x$ and follows a similar procedure to the proof of PPP-hardness of LONG CHOICE,. First, it calculates a set H_i of elements belonging to the range of C. This set H_i has the form $[x_i, y_i] \setminus \{C(a_0), C(a_1), \ldots, C(a_i)\}$. P_i then calculates the **midpoint** of H_i, defined as the smallest value $k \in H_i$ such that half of the elements of H_i are less than or equal to k. P_i returns 1 if $C(x)$ is less than or equal to the midpoint of H_i and 0 otherwise.

It now remains to define the sets H_i. The sets H_i are defined inductively with the following two key properties:

1. $|H_i| \geq 2^{n-i} - 2$
2. If $a_0, a_1, \ldots, a_i$ is a Long Choice subcertificate, then for every $k \in \{0, 1, \ldots, i-1\}$, $C(a_j) \in H_k \bigcup \{C(a_0), C(a_1), C(a_2), \ldots, C(a_k)\}$ whenever $j > k$.

As a base case, given a first input of a_0, P_0 defines $H_0 := [2^n - 1] \setminus \{C(a_0)\}$. This satisfies the first inductive property: $[2^n - 1] \setminus \{C(a_0)\}$ has $2^{n-0} - 2$ elements.

To see why H_0 satisfies the second inductive property, note that $H_0 \bigcup C(a_0)$ is actually the set $[2^n - 1]$ - the range of C! Therefore, for any subcertificate, $a_0, a_1, \ldots, a_i$, we will have, for all $k \in \{1, \ldots, i\}$, that $C(a_k) \in H_0 \bigcup C(a_0)$, as desired.

Now, suppose that the $H_0, \ldots, H_i$ have been defined in such a way that they satisfy the two inductive properties. Suppose P_i takes as input a valid subcertificate $a_0, \ldots, a_i$, along with a final element a_{i+1}. Then, P_i first uses the elements $a_0, \ldots, a_i$ to define H_i. Next, it calculates the midpoint of H_i. Then, $P_i(a_0, a_1, \ldots, a_i, a_{i+1}$ returns 1 if $C(x)$ is less than or equal to the midpoint and 0 otherwise. Accordingly, we define H_{i+1} as follows:

1. If $P_i(a_0, \ldots, a_i, a_{i+1})$ is 1, then we define H_{i+1} to be the elements of $H_i \setminus C(a_{i+1})$ which are less than or equal to the midpoint of H_i
2. Otherwise, we define H_{i+1} to be the elements of $H_i \setminus C(a_{i+1})$ which are greater than the midpoint of H_i

To prove that the first inductive property holds for H_{i+1}, recall that by our inductive assumption, H_i contains at least $2^{n-i} - 2$ elements. By definition, the midpoint will split this set into two sets, A and B, each of size at least $2^{n-i-1} - 1$. H_{i+1} is defined as either $A \setminus C(a_{i+1})$ or $B \setminus C(a_{i+1})$. Thus, in either case, H_i has at least $2^{n-i-i} - 2$ elements, as desired.

To prove that the second property holds, assume without loss of generality that $C(a_i)$ is greater than the midpoint of H_{i-1}. Then, as described above, we define H_i to be the subset of $H_{i-1} \setminus C(a_i)$ containing elements greater than the midpoint of H_{i-1}.

Now, in order for $P_{i-1}(a_0, a_1, \ldots, a_{i-1}, a_j)$ to be constant for all $j \geq i$, we must have that $C(a_j)$ is also greater than the midpoint of H_{i-1}. However, by the second inductive assumption, we know that $C(a_j)$ must belong to $H_{i-1} \bigcup \{C(a_0), \ldots, C(a_{i-1})\}$. If we apply these two facts together, we conclude that $C(a_j)$ must belong to the set $H_i \bigcup \{a_0, \ldots, a_{i-1}, a_i\}$. Letting $j = i + 1$ in our case proves that the second inductive property holds.

Finally, consider any certificate to this instance. It consists of a subcertificate $a_0, \ldots, a_j$ ($j \leq n - 2$) and a bit, c. As we know, this bit c has the following property: there is no object a_{j+1} which both extends the certificate and also has $P_j(a_0, \ldots, a_j, a_{j+1}) = c$.

Consider the set H_j; it has size at least $2^{n-(j-2)} - 2 \geq 2^{n-(n-2)} - 2 = 2$. Thus, the midpoint of H_j splits H_j into two nonempty subsets of size at least 1. Let A denote the subset of H_j containing elements less than or equal to its midpoint, and let B denote the subset of H_j containing elements greater than its midpoint. Suppose that $c = 0$. Then we can conclude that there are no elements $C(a_{j+1}) \in B$. If there were, we could pick such an element to extend the Long Choice sequence. Thus any element of B would be a solution to our problem. Similarly, if $c = 1$; then we can similarly conclude that there are no elements $C(a_{j+1}) \in A$; thus any element of A would be a solution to our problem. Finally, note that we can easily identify which elements belong to the set A and B; as mentioned earlier, they take the form $[x_i, y_i] \setminus \{C(a_0), C(a_1), \ldots, C(a_i)\}$. This completes the proof. ◀

C Missing material from Section 6 (König, Erdős-Ko-Rado)

C.1 König

The proof that König is PPP-complete (Theorem 15) follows from the next two lemmas.

▶ **Lemma 25.** König *is PPP-hard*

Proof. We provide a reduction from Collision. Suppose we are given a circuit C that defines a Collision instance:

$$C : \{0,1\}^n \mapsto \{0,1\}^n$$

We define a König problem instance on 2^{n+1} nodes, by a parent mapping :

$$P : \{0,1\}^{n+1} \mapsto \{0,1\}^{n+1} \times \{0,1\}$$

Note that P implicitly defines a graph on 2^{n+1} nodes. The nodes are supposed to be arranged in a binary tree with root 0. We start by defining the positions of nodes whose binary encodings are in the range $[0, 2^n - 1]$, and we do so recursively. First, 0 is the root. Next, consider any node s in this range. If the binary encoding of s is odd, we define $P(s) = (\lfloor \frac{s-1}{2} \rfloor, 0)$; that is, s is the left child of a node with binary encoding $\lfloor \frac{s-1}{2} \rfloor$. Similarly, if the binary encoding of s is even, we define $P(s) = (\lfloor \frac{s-1}{2} \rfloor, 1)$. (This arrangement is similar to the indexing of a heap.) Note that so far, our definition of P has absolutely no dependence on the circuit C.

To finish the definition of P, it remains to consider the nodes whose binary encodings are in the range $A = [2^n, 2^{n+1} - 1]$. Consider a given node s whose encoding lies in the range given by A. Then $s - 2^n$ lies in the domain of C. Furthermore, for each $s \in A$, $s - 2^n$ corresponds to a unique element in the domain of C. For each $s \in A$, we let $r = C(s - 2^n) + 2^n - 1$ and we define

$$P(s) = (\lfloor \frac{r-1}{2} \rfloor, \text{parity}(C(s - 2^n)))$$

where the parity function returns 0 if the input is even and 1 if it is odd.

Because the circuit C has range $[0, 2^n - 1]$, r must lie in the interval $[2^n - 1, 2^{n+1} - 1]$. Then, based on the definition of $P(s)$, the parent of each $s \in A$ must lie in the interval $B = [2^{n-1} - 1, 2^n - 1]$. There are 2^{n-1} nodes in the interval B, and each of these nodes can have at most 2 children. Thus, a total of 2^n nodes can have parent nodes in the interval B.

On the other hand, every node of A must have a parent in the interval B, and we know there are 2^n nodes in the interval A. In addition to the nodes of A, we know that the node $2^n - 1$, which does not belong to A, also has a parent in B: $P(2^n - 1) = 2^{n-1} - 1$. This implies that a total of $2^n + 1$ elements must have a parent belonging to the interval B.

Since B can only have 2^n children, by the pigeonhole principle, there must exist a node in B with two left children or two right children. By the definition of the KÖNIG problem, these two left or right children form a valid certificate to the KÖNIG problem. It remains to show that these two nodes also provide a certificate for the COLLISION problem.

There are two cases to consider. In the first case, suppose that parent node $2^{n-1} - 1$ has two left children. We know that one of these children is the node $2^n - 1$, and the other child must be a node $s \in A$. In this case, if $P(s) = (\lfloor \frac{r-1}{2} \rfloor = 2^{n-1} - 1, 0)$, we can conclude that $r = 2^n - 1$. However, this would in turn imply that $C(s - 2^n) = 0$, which means $s - 2^n$ is a zero element to our original COLLISION problem!

In the second case, suppose that a parent node $i > 2^{n-1} - 1$ has two left (or right) children. This would imply that there are two distinct nodes $s_1, s_2 \in A$ with $P(s_1) = P(s_2)$. By the definition of P, this in turn implies that $C(s_1 - 2^n) = C(s_2 - 2^n)$. Since $s_1 \neq s_2$, we can conclude that $s_1 - 2^n$ and $s_2 - 2^n$ provide us with a collision in our COLLISION certificate. ◀

▶ **Lemma 26.** KÖNIG *is in PPP*

Proof. We are given an instance of the KÖNIG problem, which is defined by a root node, r, and a circuit $P : \{0,1\}^n \mapsto \{0,1\}^n \times \{0,1\}$ As described above, P implicitly describes a graph on 2^n nodes. We wish to reduce it to an instance of Pigeonhole Circuit in polynomial time. To do this, we define a circuit, C:

$$C : \{0,1\}^n \mapsto \{0,1\}^n$$

where the domain of C will be the nodes of the graph defined by P.

In any binary tree, we can assign every node a unique "index" based on its position relative to the root. This indexing scheme is defined in an inductive fashion. First, the root is given index 0. Next, suppose a given node a_i has index i. Then, we say that its left child has index $2 * i + 1$ and its right child has index $2 * i + 2$;

Consider any node of the graph, g. Suppose that g is neither a Far away node nor a Long path certificate nor a Non-unique root. (Note that all of these conditions are easy to verify). Under these assumptions, we show how to efficiently find the index of g. To do this, we repeatedly apply the parent operator, P, to the node g until we reach the root node. Based on the assumptions we have made, this is always possible. Furthermore, it will require at most $n-1$ applications of the parent operator. Every time we apply the parent operator to a node, we receive two pieces of information: the node's parent and whether the node is a left or right child of its parent. Thus, once we reach the root node in this process, we have a sequence of nodes $a_0, a_1, a_2, \ldots, a_k$ where $a_k = r$ and $a_0 = g$. Furthermore, for each a_i and a_{i+1}, we know whether a_i is the left or right child of a_{i+1}. We therefore have a path from r to g. Using this path, we can use the indexing scheme mentioned above to find the indices of the nodes $r = a_k, a_{k-1}, \ldots, a_0 = g$ in that order to finally arrive at the index of node g.

We are now in a position to fully define the circuit C. On input i, $C(i)$ outputs:

1. 0 if r is an invalid root
2. 0 if $i \neq r$ and i is a root (in this case, there is a non-unique root).
3. 0 if i is a Far Away Node
4. 0 if i provides a certificate for a Long Path.
5. If **none** of the above conditions are met, then let x be the index of node i in the binary tree. $C(i)$ returns $x+1$.

It now remains to show that a certificate to the KÖNIG problem can be recovered from a certificate to the COLLISION problem we have just defined. There are two cases to consider.

In the first case, suppose our certificate to the COLLISION problem is a zero element. In this case, there are four possibilities. Either r is an invalid root, or $i \neq r$ is a root, or i is a Far Away Node or i is a certificate for a Long Path. We can polynomially verify all of these conditions, and they all represent valid certificates to the KÖNIG problem.

In the second case, suppose our certificate to the COLLISION problem is a collision of two elements in which neither element is a zero element: $i \neq j$ with $C(i) = C(j) \neq 0$. Then, we know that i and j are not Long Path certificates and are not Far Away nodes. Furthermore, we know that their indices must be equal to each other, which in turn implies that they are left (or right) children of the same parent node. Thus, i and j are identical children and provide us with a valid certificate. ◀

C.2 Erdős-Ko-Rado

▶ **Theorem 18.** ERDŐS-KO-RADO *is PPP-complete.*

Proof. We first prove that ERDŐS-KO-RADO is PPP-hard. Accordingly, suppose we are given some instance of COLLISION, defined by a circuit $C : \{0,1\}^n \mapsto \{0,1\}^n$, and the challenge is to find an input that maps to 0 or find two distinct inputs which map to the same value.

We construct a circuit, F, which implicitly defines a 2-set system of size 2^n. Our circuit F takes, as input, an n-bit string, and outputs a $(\{0,1\}^n)^2$-bit string: $F(x) = (0^n, C(x))$.

It remains to show that any solution to this instance of the ERDŐS-KO-RADO problem allows us to recover a solution to the original COLLISION problem. There are 3 possible types of certificates that we can find in the ERDŐS-KO-RADO problem instance. First, consider a certificate providing two disjoint sets. This is impossible from the above definition of F, since every set contains the element 0^n. Second, consider a certificate x providing an invalid set $F(x)$. This means that $C(x) = 0^n$, i.e. we find a zero element in the COLLISION problem. Third, consider a certificate which defines a repeat set. That is, we have two indices a and b with $F(a)$ and $F(b)$ defining the same set. As we see in the above definition of F, this implies that $C(a) = C(b)$, which provides us with a collision for our original COLLISION problem. Thus, any certificate for the ERDŐS-KO-RADO problem provides us with a certificate for the COLLISION problem above.

We now reduce Erdős-Ko-Rado to Collision. Consider an Erdős-Ko-Rado problem instance, defined by a circuit F which implicitly provides us with a 2-set system of size 2^n. We construct a circuit $C : \{0,1\}^n \mapsto \{0,1\}^n$ as follows. We first examine two arbitrary sets from the set system, F. Without loss of generality, we examine $F(0)$ and $F(1)$. If these two sets are disjoint, identical, or invalid, we are done. Otherwise, they have an intersection of exactly 1 element. Let $F(0) = (a, b)$ and $F(1) = (b, c)$, such that their intersection is $\{b\}$. For any given index i, define $C(i)$, as follows:

1. If $F(i)$ outputs an invalid set, set $C(i) = 0$
2. If $F(i)$ contains both b and d (where $d \neq b$), then we let $C(i) = d$.
3. If $F(i)$ does not contain b, then set $C(i) = 0$.

There are two cases to consider here. We start by considering the case in which we recover a zero element. That is, we find an element x with $C(x) = 0$. This can only happen in 2 cases, based on the above definition of C. First, $F(x)$ might generate an invalid set. In this case, x is a valid certificate. Second, $F(x)$ might not contain b. In this case, suppose $F(x) = (p, q)$. We note that x is distinct from 1 and 0, because 0 and 1 are not zero elements, based on the above definition. If $F(x)$ is disjoint from either $F(1)$ or $F(0)$, then we are done. If, instead, $F(x)$ does not contain b and has nonempty intersection with $F(1)$ and $F(0)$, then $F(x)$ must contain a and c. Then, consider *any* index y that is distinct from $0, 1, x$. If $F(y)$ is invalid, we are done. If $F(y)$ is identical to one of $F(0), F(1), F(x)$, we have found a repeated set certificate, and we are done. Otherwise, $F(y)$ must be fully disjoint from at least one of $F(0), F(1), F(x)$, and we can recover two disjoint sets. That is, the only way a set $F(y)$ of size 2 intersects each of $\{a, b\}, \{a, c\}, \{b, c\}$ is if $F(y)$ is identical to one of $\{a, b\}, \{a, c\}, \{b, c\}$.

Now, suppose that we recover a collision from C. That is, we find two distinct n-bit strings x and y with $C(x) = C(y)$. First, suppose $C(x) = C(y) \neq 0$. Based on the above definition of C, this can only happen if $F(x)$ and $F(y)$ both contain the element b as well as the element $C(x)$. In this case, $F(x)$ and $F(y)$ both describe the set $\{b, C(x)\}$, which means that we have recovered a "repeated set" certificate. In the second case, suppose $C(x) = C(y) = 0$. That is, suppose we have found two zero certificates. In this case, the argument from above tells us that we can recover the desired certificate using any one of these zero certificates. In this case, $F(x)$ and $F(y)$ are each either invalid sets or they do not contain b. In the case that they are invalid sets, we are done. If these indices both do not contain b, then $F(x)$ and $F(y)$ must each contain both a and c in order to have nonempty intersection with $F(1)$ and $F(2)$. ◀