The complexity of the $K$th largest subset problem and related problems

Christoph Haase$^a,*,1$, Stefan Kiefer$^b,*,2$

$^a$ LSV, CNRS & ENS Cachan, Université Paris-Saclay, France
$^b$ Department of Computer Science, University of Oxford, UK

**Abstract**

We show that the $K$th largest subset problem and the $K$th largest $m$-tuple problem are in PP and hard for PP under polynomial-time Turing reductions. Several problems from the literature were previously shown NP-hard via reductions from those two problems, and by our main result they become PP-hard as well. We also provide complementary PP-upper bounds for some of them.

© 2015 Elsevier B.V. All rights reserved.

1. Introduction

The following two problems are listed in Garey and Johnson’s classical compendium on the theory of NP-completeness [1, p. 225] as problems [SP20] and [SP21], respectively:

**$K$th Largest Subset Problem**

**INSTANCE:** A set $X = \{x_1, \ldots, x_m\} \subseteq \mathbb{N}$ and $K, B \in \mathbb{N}$.

**QUESTION:** Is $\#\{Y \subseteq X : \sum_{x \in Y} x \leq B\} \geq K$?

**$K$th Largest $m$-tuple Problem**

**INSTANCE:** Finite sets $X_1, \ldots, X_m \subseteq \mathbb{N}$ and $K, B \in \mathbb{N}$.

**QUESTION:** Is $\#\{(x_1, \ldots, x_m) \in X_1 \times \cdots \times X_m : \sum_{i=1}^m x_i \geq B\} \geq K$?

The first problem was initially introduced by Johnson and Kashdan [2] and the second by Johnson and Mizoguchi [3]. Garey and Johnson [1] state that both problems are NP-hard under polynomial-time Turing reductions but mention that their membership in NP, and, thus, their NP-completeness, is open. For the NP-hardness proof of the $K$th largest subset problem, see Theorem 6 in the paper of Johnson and Kashdan [2], or the discussion provided by Garey and Johnson [1, p. 115]. It is unknown whether this problem is also NP-hard under polynomial-time many-one reductions [4, p. 148]. The $K$th largest subset problem has its name for historical reasons: note that the problem is about the $K$th smallest subset.

Both problems have repeatedly been used as starting points for proving other problems NP-hard. In this note, we show that the $K$th largest subset and the $K$th largest $m$-tuple problem are complete for the complexity class PP. More precisely, they are in PP, and they are hard for PP under polynomial-time Turing reductions. The complexity class PP [5] can be defined as the class of languages $L \subseteq \Sigma^*$ that have an NP Turing machine $M_L$ such that for all words $w \in \Sigma^*$ one has $w \in L$ if and only if more than
half of the computation paths of \( M_L \) on \( w \) are accepting. The class \( \text{PP} \) includes \( \text{NP} \). Closely related is the function class \( \#\text{P} \), which consists of those functions \( f: \Sigma^* \to \mathbb{N} \) for which there exists an \( \text{NP} \) Turing machine \( M_f \) such that for all \( w \in \Sigma^* \) the function value \( f(w) \) is equal to the number of accepting computation paths of \( M_f \) on \( w \). The class \( \text{PP} \) is remarkably powerful: Toda’s Theorem [6] states that \( \text{P}^{\text{PP}} \) (which equals \( \text{P}^{\text{NP}} \)) contains the polynomial-time hierarchy \( \text{PH} \). This means, in particular, that if the \( k \)-th largest subset (or \( m \)-tuple, resp.) problem were in \( \text{NP} \) then the polynomial-time hierarchy would collapse to \( \text{P}^{\text{NP}} \). This can be taken as evidence (under standard assumptions from complexity theory) that the problems considered in this note are not in \( \text{NP} \) and, hence, are harder than \( \text{NP} \).

By an application of the main result of this note, it follows that several other problems for which reductions from the \( k \)-th largest subset (or \( m \)-tuple) problem have been constructed are not only \( \text{NP} \)-hard under polynomial-time Turing reductions, but in fact \( \text{PP} \)-hard under polynomial-time Turing reductions. We give examples of such problems in Section 3.

2. Main result

The class \( \text{PP} \) can equivalently be characterised as the class of languages \( L \subseteq \Sigma^* \) that have a polynomial-time bounded deterministic Turing machine \( M \) with access to a fair coin such that for all \( w \in \Sigma^* \) we have \( w \in L \) if and only if \( M \) accepts \( w \) with probability at least \( 1/2 \). This follows from Theorem 4.4 of Simon [7], and in particular from the fact that \( \text{P} \) is closed under complement [7], see also the work of Gill [5]. However, in some proofs it will be more convenient if the Turing machine has access to biased rather than fair coins, and if the biases (rational numbers between 0 and 1) can depend on the input. More precisely, a probabilistic polynomial-time bounded Turing machine (\( \text{PP} \)-TM) is a polynomial-time bounded Turing machine that takes an input \( w \) and then operates in two phases:

1. Depending only on \( w \), it deterministically computes finitely many rational numbers \( b_1, \ldots, b_k \) with \( b_i \in (0, 1) \) for all \( i \in \{1, \ldots, k\} \). Each \( b_i \) is represented as a quotient of integers written in binary. We view the \( b_i \) values as biases of biased coins.
2. It acts like a nondeterministic polynomial-time bounded Turing machine, but the nondeterminism is resolved probabilistically using biased coins with biases \( b_1, \ldots, b_k \) that were computed in the first phase. More concretely, besides deterministic instructions, the Turing machine has instructions of the form

\[
\text{if coin}(b_i) \text{ then goto c1 else goto c2}
\]

where \( b_i \) is a bias computed in the first phase. The semantics of this operation is the natural one: with probability \( b_i \) the Turing machine continues at c1, otherwise (i.e., with probability \( 1 - b_i \)) it continues at c2.

We have the following lemma.

**Lemma 1.** Let \( L \subseteq \Sigma^* \) and \( M \) be a \( \text{PP} \)-TM. Furthermore, let \( \tau: \Sigma^* \to (0, 1) \) be a polynomial-time computable function returning a rational number encoded as a quotient of integers in binary such that for all \( w \in \Sigma^* \) we have \( w \in L \) if and only if \( M \) accepts \( w \) with probability at least \( \tau(w) \). Then \( L \in \text{PP} \).

**Proof.** The proof consists of three steps:

1. We show that we can, without loss of generality, take \( \tau(w) = 1/2 \) for all \( w \in \Sigma^* \).
2. We construct from \( M \) an \( \text{NP} \) Turing machine \( M' \) such that for all \( w \in \Sigma^* \) we have \( w \in L \) if and only if at least half of the computation paths of \( M' \) on \( w \) are accepting.
3. We argue that the existence of \( M' \) implies membership of \( L \) in \( \text{PP} \).

In the first step of the proof, we show that we can, without loss of generality, take \( \tau(w) = 1/2 \) for all \( w \in \Sigma^* \).

Indeed, given \( M \) and \( \tau \), construct a \( \text{PP} \)-TM \( M' \) as follows. Given \( w \in \Sigma^* \), the \( \text{PP} \)-TM \( M' \) first computes \( \tau := \tau(w) \).

If \( \tau < 1/2 \), it computes \( p := (1/2 - \tau)/(1 - \tau) \), and accepts immediately with probability \( p \). Otherwise (i.e., with probability \( 1 - p \)) the \( \text{PP} \)-TM \( M' \) simulates \( M \). It follows that

\[
\Pr(M' \text{ accepts } w) = p + (1 - p) \cdot \Pr(M \text{ accepts } w).
\]

A simple calculation shows that \( \Pr(M \text{ accepts } w) \geq \tau(w) \) if and only if \( \Pr(M' \text{ accepts } w) \geq 1/2 \).

Similarly, if \( \tau > 1/2 \), the \( \text{PP} \)-TM \( M' \) computes \( p \) as above, and rejects immediately with probability \( 1 - p \). Otherwise (i.e., with probability \( p \)) the \( \text{PP} \)-TM \( M' \) simulates \( M \). It follows that

\[
\Pr(M' \text{ accepts } w) = p \cdot \Pr(M \text{ accepts } w).
\]

Again, we have \( \Pr(M \text{ accepts } w) \geq \tau(w) \) if and only if \( \Pr(M' \text{ accepts } w) \geq 1/2 \).

Consequently, for the remainder of the proof we can assume that for all \( w \in \Sigma^* \) we have \( w \in L \) if and only if \( M \) accepts \( w \) with probability at least \( 1/2 \). Furthermore, we can ensure that for any \( w \in \Sigma^* \) all computation paths of \( M \) on \( w \) involve the same number, say \( p_w \), of accesses to a coin.

In the second step of the proof, we construct from \( M \) an \( \text{NP} \) Turing machine \( M' \) such that for all \( w \in \Sigma^* \) we have \( w \in L \) if and only if at least half of the computation paths of \( M' \) on \( w \) are accepting. Given a word \( w \in \Sigma^* \), the Turing machine \( M' \) computes, like the \( \text{PP} \)-TM \( M \), the set \( B \) of biases. Then \( M' \) computes a multiple, say \( q_w \), of the denominators of the biases in \( B \). Subsequently, \( M' \) simulates \( M \), except that when it hits an instruction

\[
\text{if coin}(b) \text{ then goto c1 else goto c2}
\]

then \( M' \) takes \( b \cdot q_w \) nondeterministic branches to c1 and \( (1 - b) \cdot q_w \) nondeterministic branches to c2. By the definition of \( q_w \) the numbers \( b \cdot q_w \) and \( (1 - b) \cdot q_w \) are natural numbers. They might be of exponential magnitude, but their representation in binary is of polynomial length. Using this representation, the Turing machine \( M' \) can implement the \( b \cdot q_w \) and \( (1 - b) \cdot q_w \) nondeterministic branches.
using cascades of binary nondeterministic branches. By this construction, we have for any \( w \in \Sigma^* \) that the total number of computation paths of \( M' \) on \( w \) equals \( (q_w)^P \) and, furthermore, if \( M \) accepts \( w \) with probability \( x \) then \( x \cdot (q_w)^P \) computation paths of \( M' \) on \( w \) are accepting. We conclude that we have \( w \in L \) if and only if at least half of the computation paths of \( M' \) on \( w \) are accepting.

For the third and final step of the proof, construct an NP Turing machine \( M' \) from \( M \) by swapping accepting and rejecting states. Then we have \( w \not\in L \) if and only if more than half of the computation paths of \( M' \) on \( w \) are accepting. Hence, using the definition of PP, the complement of \( L \) is in PP. But PP is closed under complement, see [5], thus \( L \in \text{PP} \). \( \square \)

In order to show PP-hardness of the \( K \text{TTH LARGEST SUBSET} \) and the \( K \text{TTH LARGEST m-TUPLE} \) problem, we employ \#P-completeness of a variant of the classical \text{SubsetSum} problem. Given a set \( X = \{ x_1, \ldots, x_m \} \subseteq \mathbb{N} \) and \( T \in \mathbb{N} \), \#\text{SubsetSum} asks for determining the number of subsets \( Y \) of \( X \) whose elements sum up to \( T \), i.e., computing

\[
\#\left\{ Y \subseteq X : \sum_{x \in Y} x = T \right\}.
\]

Building upon the results of Hunt et al. [8], Faliszewski and Hemaspaandra [9, p. 104] derive that \#\text{SubsetSum} is \#P-complete, where \#P-hardness holds under parsimonious many-one reductions. As defined by Simon [7], given functions \( f, g : \Sigma^* \rightarrow \mathbb{N} \) we say that \( f \) parsimoniously reduces to \( g \) if there is a polynomial-time computable function \( h : \Sigma^* \rightarrow \Sigma^* \) such that \( f(w) = g(h(w)) \) for all \( w \in \Sigma^* \). For subsequent reference, let us capture the result of Faliszewski and Hemaspaandra [9] by the following proposition.

**Proposition 2.** \#\text{SubsetSum} is \#P-complete, and in particular \#P-hard under parsimonious many-one reductions.

We are now prepared to prove our main result. In the remainder of this note, whenever we refer to \#P-completeness, \#P-hardness holds under polynomial-time Turing reductions. In the statements of the theorems we make this explicit to avoid a confusion of casual readers.

**Theorem 3.** The \( K \text{TTH LARGEST SUBSET} \) problem and the \( K \text{TTH LARGEST m-TUPLE} \) problem are \#P-complete under polynomial-time Turing reductions.

**Proof.** First, we prove membership in PP for both problems. Given an instance \( X = \{ x_1, \ldots, x_m \} \subseteq \mathbb{N} \) and \( B, K \in \mathbb{N} \) of the \( K \text{TTH LARGEST SUBSET} \) problem, we can construct a PP-TM \( M \) that first chooses every subset \( Y \subseteq X \) uniformly at random by selecting every \( x_i \) with probability \( 1/2 \). Subsequently, \( M \) accepts if and only if \( \sum_{x \in Y} x \leq B \). It follows that the probability that \( M \) accepts is equal to

\[
\#\left\{ Y \subseteq X : \sum_{x \in Y} x \leq B \right\} / 2^m.
\]

Hence, \( \#\{ Y \subseteq X : \sum_{x \in Y} x \leq B \} \geq K \) if and only if \( M \) accepts with probability at least \( K / 2^m \). By Lemma 1, the \( K \text{TTH LARGEST SUBSET} \) problem is in PP.

For the \( K \text{TTH LARGEST m-TUPLE} \) problem, the construction is similar: a PP-TM \( M \) selects an \( m \)-tuple \( (x_1, \ldots, x_m) \in X_1 \times \cdots \times X_m \) uniformly at random, i.e., each \( m \)-tuple with probability \( 1 / \prod_{i=1}^{m} (#X_i) \). By the characterisation of PP in Lemma 1, this can be achieved by independently choosing some \( x_i \in X_i \) with probability \( 1 / #X_i \) for every \( 1 \leq i \leq m \). Then, \( M \) accepts if and only if \( \sum_{i=1}^{m} x_i \leq B \). Hence, \( \#\{ (x_1, \ldots, x_m) \in X_1 \times \cdots \times X_m : \sum_{i=1}^{m} x_i \geq B \} \geq K \) if and only if \( M \) accepts with probability at least \( K / \prod_{i=1}^{m} (#X_i) \). An application of Lemma 1 shows that the \( K \text{TTH LARGEST m-TUPLE} \) problem is in PP as well.

In order to show hardness for PP, we give polynomial-time Turing reductions from the canonical \#P-complete problem \text{MaxSAT} [5,10]. This problem asks, given a Boolean formula \( \psi \) over \( n \) Boolean variables \( b_1, \ldots, b_n \), does a majority (i.e., at least \( 2^{n-1} + 1 \)) of the variable assignments satisfy \( \psi \). Since \#SAT, i.e., the task of computing the number of satisfying assignments of a Boolean formula, is \#P-complete [10], it follows from Proposition 2 that there is a polynomial-time computable function \( h(\psi) \) that outputs an instance \( X = \{ x_1, \ldots, x_m \} \) and \( T \in \mathbb{N} \) of \#\text{SubsetSum} such that

\[
L := \#\left\{ Y \subseteq X : \sum_{x \in Y} x = T \right\} = \#\left\{ (b_1, \ldots, b_n) \in \{0,1\}^n : \psi(b_1, \ldots, b_n) = 1 \right\}.
\]

Using binary search, with at most \( 2 \cdot m \) queries to the \( K \text{TTH LARGEST SUBSET} \) problem we can compute

\[
M := \#\left\{ Y \subseteq X : \sum_{x \in Y} x \leq T \right\} \quad \text{and}
\]

\[
N := \#\left\{ Y \subseteq X : \sum_{x \in Y} x \leq T - 1 \right\}.
\]

We now have that \( L = M - N \), and hence a majority of the variable assignments satisfies \( \psi \) if and only if \( M - N \geq 2^{n-1} + 1 \).

In the same fashion, we can show hardness of the \( K \text{TTH LARGEST m-TUPLE} \) problem. Set \( X_i := \{0, x_i\} \) for every \( 1 \leq i \leq m \). As before, using binary search with at most \( 2 \cdot m \) queries we can compute

\[
M := \#\left\{ (x_1, \ldots, x_m) \in X_1 \times \cdots \times X_m : \sum_{i=1}^{m} x_i \geq T \right\} = \#\left\{ Y \subseteq X : \sum_{x \in Y} x \geq T \right\}
\]

and

\[
N := \#\left\{ (x_1, \ldots, x_m) \in X_1 \times \cdots \times X_m : \sum_{i=1}^{m} x_i \geq T + 1 \right\} = \#\left\{ Y \subseteq X : \sum_{x \in Y} x \geq T + 1 \right\}.
\]
As above, \( L = M - N \) and a majority of the variable assignments satisfies \( \psi \) if and only if \( M - N \geq 2^{n-1} + 1 \), from which PP-hardness of the problem follows. \( \square \)

### 3. Applications

In this section, we discuss several problems whose NP-hardness has been shown in the literature by reductions from the \( K \)th largest subset (or \( m \)-tuple) problem. By Theorem 3, all of these problems become PP-hard. As mentioned in Section 1, Toda’s Theorem implies that if any of these problems were in NP (and hence NP-complete), the polynomial-time hierarchy would collapse to \( \text{P}^{\text{NP}} \).

### 3.1. The \( K \)th Largest-Area Convex Polygon Problem

Chazelle [11] posed the following question: “If given \( n \) points in the Euclidean plane, how hard is it to compute the \( K \)th largest-area convex polygon formed by any subset of the points?” This question was essentially answered by Salowe [12]. In particular, the following decision variant was considered in [12]:

**Theorem 4.** The \( K \)th largest-area convex polygon problem is \( \text{PP} \)-complete under polynomial-time Turing reductions.

**Proof.** PP-hardness follows from the reduction in [12] combined with Theorem 3. Towards membership in \( \text{PP} \), construct a \( \text{PP-TM} \ M \), which, given an instance of the \( K \)th largest-area convex polygon problem, selects a subset \( P' \) of \( P \) uniformly at random, and then checks (in polynomial time) whether \( P' = \text{extr} (\text{conv}(P')) \) and the area of \( \text{conv}(P') \) is at least \( B \).

Here, \( \text{conv}(P') \) refers to the convex hull of the points in \( P' \), and \( \text{extr}(\text{conv}(P')) \) refers to the set of extreme points in that convex hull. Salowe [12] showed that the \( K \)th largest-area convex polygon problem is \( \text{NP} \)-hard, via a reduction from the \( K \)th largest \( m \)-tuple problem. In fact, we have:

**Theorem 5.** The vertex counting problem is \( \text{PP-complete under polynomial-time Turing reductions} \).

**Proof.** PP-hardness follows from the reduction in [13] combined with Theorem 3. To prove membership in \( \text{PP} \), we use the fact that \( \text{PP} \) is closed under complement [5]. So it suffices to prove \( \text{PP} \)-membership of the co-vertex counting problem, which is like the vertex counting problem except that it asks whether the polyhedron has at least \( K \) vertices. Construct a \( \text{PP-TM} \ M \), which, given an instance of the co-vertex counting problem, selects \( n \) bounding hyperplanes, say \( h_1, \ldots, h_n \), uniformly at random, and then checks whether they have a unique intersection. If no, \( M \) rejects. Otherwise \( M \) computes the unique intersection, say \( p \), and checks whether \( p \) is in the polyhedron. If no, \( M \) rejects. Otherwise \( M \) computes the set \( S \) of all bounding hyperplanes on which \( p \) lies. Then \( M \) computes the lexicographically smallest \( n \)-element subset \( S' \) of \( S \) such that \( p \) is the unique intersection of the hyperplanes in \( S' \). That set \( S' \) can be computed in polynomial time, e.g., by greedibly adding hyperplanes with linearly independent normal vectors. If \( S' = \{ h_1, \ldots, h_n \} \), then \( M \) accepts, otherwise \( M \) rejects. By construction, any vertex \( p \) is accepted by exactly one run of \( M \). So the acceptance probability is at least \( K/2^n \) if and only if the given instance of the co-vertex counting problem is a yes-instance. An application of Lemma 1 shows membership in \( \text{PP} \). \( \square \)

### 3.3. Routing Flows with Delay Guarantees

In [14] the problem of routing flows through networks is considered, where the information available for making routing decisions is probabilistic. The aim of the routing algorithm is to guarantee certain quality-of-service requirements. One of the specific problems is to evaluate the end-to-end delay of a given path, as we describe in the following.

Let \( e_1, \ldots, e_n \) be a path in a network. For each \( i \in \{1, \ldots, n\} \), the number \( d_i \in \mathbb{N}_0 \) is a random delay introduced by the link \( e_i \). For all \( i \in \{1, \ldots, n\} \) and \( k \in \mathbb{N}_0 \),
the probability that \( d_i = k \) is given by \( p_{i,k} \in [0,1] \cap \mathbb{Q} \). It is assumed that for each \( i \) there are only finitely many \( k \) with \( p_{i,k} > 0 \), and that the delays \( d_i \) are independent. The \( end-to-end \) delay is \( \sum_{i=1}^{n} d_i \). An \( end-to-end \) delay requirement is given by a maximum delay \( D \in \mathbb{N}_0 \) and a probability threshold \( \tau \in [0,1] \cap \mathbb{Q} \). Guerin and Orda refer to the following problem as “problem \( P(\pi) \)” [14]:

**DELAY GUARANTEE**

**INSTANCE:** A path length \( n \in \mathbb{N} \), a list of all positive \( p_{i,k} \in [0,1] \cap \mathbb{Q} \), the maximum delay \( D \in \mathbb{N}_0 \), a probability threshold \( \tau \in [0,1] \cap \mathbb{Q} \).

**QUESTION:** Is the probability of \( \sum_{i=1}^{n} d_i \leq D \) at least \( \tau \)?

In [14, Lemma IV.1] the **DELAY GUARANTEE** problem was shown NP-hard by a reduction from the **KTH LARGEST SUBSET** problem. In fact, we have:

**Theorem 6.** The **DELAY GUARANTEE** problem is PP-complete under polynomial-time Turing reductions.

**Proof.** PP-hardness follows from the reduction in [14] combined with Theorem 3. Towards membership in PP, construct a PP-TM \( M \), which, given an instance of the **DELAY GUARANTEE** problem, takes independent random values for the \( d_i \) according to the given \( p_{i,k} \) and then checks whether \( \sum_{i=1}^{n} d_i \leq D \). If yes, \( M \) accepts, otherwise \( M \) rejects. So the acceptance probability is at least \( \tau \) if and only if the given instance of the **DELAY GUARANTEE** problem is a yes-instance. An application of Lemma 1 shows membership in PP. □

### 3.4. Further probabilistic problems

The **KTH LARGEST SUBSET** problem has been used by Laroussinie and Sproston [15] to show NP-hardness and coNP-hardness of model-checking “durational probabilistic systems” (a variant of timed automata) against the probabilistic and timed temporal logic PTCTL. Combining this reduction (which is similar to the one for the **DELAY GUARANTEE** problem) with Theorem 3 strengthens the NP-hardness and coNP-hardness results of [15] to PP-hardness.

Finally, the **KTH LARGEST m-TUPLE** problem was used by Cire et al. [16] to show NP-hardness of another probabilistic problem, namely determining whether a so-called “chance-alldifferent constraint” has a solution. This is a weighted extension of stochastic constraint programming. Cire et al. [16] remarked: “Also, we are not aware if the problem of deciding whether there exists a feasible solution to chance-alldifferent is in NP.” But this seems unlikely, as their reduction combined with Theorem 3 proves PP-hardness of this problem.

**References**