The Maximum (k,m)-Subsets Problem is in the Class NEXP

Khalil Challita, Member, IAENG, Jacques Bou Abdo

Abstract—In this paper we define and solve the following new problem: Given a set of \( n \) elements, we are interested in determining the largest number of subsets of size \( k \) that have at most \( m \) elements in common. We prove that in the worst-case scenario, a brute force solution requires a double exponential time with respect to \( n \). Afterwards, we show that our problem is in the class NEXP by reducing it to the succinct K-coloring problem. We use in our proof ordered binary decision diagrams to determine whether there is an edge between two nodes of a graph that corresponds to an instance of our problem. We also show that it is at least EXP-hard.

Index Terms—K-coloring, binary decision diagrams, complexity classes.

I. INTRODUCTION

Given a cluster of \( n \) computers, we wish to separate them in tightly connected sub-clusters of size \( k \) each, such that any couple of sub-clusters have at most \( m \) computers in common. More precisely, we are interested in maximizing the number of sub-clusters that satisfy the abovementioned condition.

Using set theory, we can formulate this problem simply as follows: Let \( S \) be a set of \( n \) elements and \( U_{k,n} \) its subsets of size \( k < n \). Compute the largest set \( S_k \subseteq U_{k,n} \) such that any couple of \( S_k \) share at most \( m \) elements. A formal description of our problem is given in Section III. Since two additional parameters (i.e. \( k \) and \( m \)) are included in the formulation of this problem, we decided to name it the maximum \((k,m)\)-subsets problem. To our knowledge, no one has solved it before, although one can find similar problems in maximizing the number of sub-clusters that satisfy the abovementioned condition.

II. PRELIMINARIES

We recall in this section some previously established results we need for solving our problem. As we already stated in the Introduction, our main proof in Section V consists in reducing the maximum \((k,m)\)-subsets problem to the succinct K-coloring problem. Galperin and Wigderson [13] showed that the former problem is in the class NEXP. Later on, Papadimitriou and Yannakakis [15] proved that it is complete for its class.

Generally speaking, we know from Papadimitriou [17] that all the NP-complete problems become NEXP-complete when the input is exponentially more succinct than the description of the original problem. This result can be extended to include many well known NP-complete problems such as SAT, Hamilton Path, K-Coloring, Knapsack, and Max Cut, where Boolean circuits were chosen to (succinctly) represent graphs.

Shannon [18] was the first to apply Boolean logic [10] to electric circuit design. Since then, his seminal work found applications in a wide variety of different fields such as cryptography, mathematics, and biology [11], [14], [12].

Recall that a Boolean circuit is a directed acyclic graph, where a node falls into one of the following three categories: an input node with no incoming edges, a constant node that is either true or false, and a gate node labeled with one of the following logical connectives \{\( \lor, \land, \neg \}\}. Figure 1 includes two examples of such circuits. The one in (a) computes the exclusive-or function, and the one in (b) represents the function \( f(x_1, x_2) = (x_1 \land x_2) \lor (\neg x_1 \land \neg x_2) \).

Manuscript received January 21, 2020; revised December 20, 2020.

Khalil Challita is an Associate Professor in the Department of Computer Science, Notre Dame University-Louaize, Zouk Mosbeh, Lebanon, (e-mail: kchallita@ndu.edu.lb).

Jacques Bou Abdo is an Assistant Professor in the College of Business and Technology, University of Nebraska at Kearney, USA, (e-mail: boubadog@unk.edu).
Papadimitriou and Yannakakis [15] used a succinct representation of a graph (with exponentially many nodes) in order to prove the NEXP-completeness of the problems they tackled. Formally speaking, given a graph \( G \) with \( n \) nodes (where \( n = 2^k \)), a succinct representation of \( G \) is a Boolean circuit \( C \) with 26 input nodes such that the following holds: for every couple of nodes \( 1 \leq i < j \leq n \), there is an edge between \( i \) and \( j \) if and only if \( C \) accepts the binary representation of the \( i \)-bit integers and \( j \)-bit integers. In our case, we used an ordered binary decision diagram (as described by Bryant [16]) to represent a Boolean function instead of using a Boolean circuit. The reason behind this choice is straightforward. As we shall see in Section V, an ordered binary decision diagram (OBDD) allows us to naturally perform the following task: given a set \( S \) of size \( n \) and two subsets \( L \subseteq S \) and \( L' \subseteq S \) of size \( k < n \), determine the number of elements the subsets \( L \) and \( L' \) have in common.

We next formulate our problem.

III. Problem Formulation

Given a set \( U_n \) that contains \( n \) elements, and given two integers \( k \) and \( m \) that satisfy \( m \leq k < n \), we consider the subsets of \( U_n \) that have size \( k \). We know that there are exactly \( \binom{n}{k} \) such subsets. More precisely, we are interested in solving the problem of determining the maximum number of subsets of size \( k \) that have at most \( m \) elements in common. Formally, the problem can be defined as follows:

**Definition 1:** Maximum \((k,m)\)-subsets problem

Let \( n, k, m \), where \( U_n = \{a_1, \ldots, a_n\} \), and \( 0 \leq m \leq k < n \). We denote by \( U_{k,n} = \{L \subseteq U_n : \text{Card}(L) = k\} \) the set of subsets of \( U_n \) that have exactly \( k \) elements.

We say that \( F_{m,k,n} \in 2^{U_{k,n}} \) is a maximum \((k,m)\)-subset of \( U_n \) if it satisfies the following conditions:

1. \( \forall L, L' \in F_{m,k,n}, \text{Card}(L \cap L') \leq m \).
2. \( \forall F' \in 2^{U_{k,n}} \) that satisfies the above condition, we have \( \text{Card}(F') \leq \text{Card}(F_{m,k,n}) \).

In this case we say that this problem has a solution of size \( K = \text{Card}(F_{m,k,n}) \leq \binom{n}{k} \).

**Example 1:** Let \( U_4 = \{a_1, a_2, a_3, a_4\} \), \( k = 2 \), and \( m = 1 \).

One can easily check that a solution to this problem is simply given by:

\[ F_{2,2,4} = \{\{a_1, a_2\}, \{a_1, a_3\}, \{a_1, a_4\}, \{a_2, a_3\}, \{a_2, a_4\}, \{a_3, a_4\}\} \]

where \( F_{2,2,4} \) happens to be equal to the set of all subsets of \( U_4 \) of size two (i.e. \( U_{2,4} \)).

Indeed, each element of \( F_{2,2,4} \) has a cardinal equal to two, and the intersection of any couple of elements in \( F_{2,2,4} \) is less than or equal to one. To conclude, it is easy to see that the above set is the largest one that satisfies the conditions stated in Definition 1 since \( \text{Card}(F_{2,2,4}) = \binom{2}{2} = 6 \).

**Example 2:** Let \( U_5 = \{a_1, a_2, a_3, a_4, a_5\} \), \( k = 3 \), and \( m = 1 \).

It is straightforward to see that a possible solution for this problem is given by \( F_{3,5,5} = \{\{a_1, a_2, a_3\}, \{a_1, a_4, a_5\}\} \).

Note that this solution is not unique since \( F' = \{\{a_1, a_2, a_3\}, \{a_3, a_4, a_5\}\} \) is another possible one.

Now any attempt to add a set of size \( k = 3 \) to \( F_{3,5,5} \) results in violating the first constraint given in Definition 1.

The trivial case where \( m = 0 \) can be solved in linear time. Indeed, given \( U_n \) and a positive integer \( k \), we consider subsets that contain \( k \) successive elements from \( U_n \). By construction, the intersection of any two subsets in \( F_{k,n}^0 \) is empty. Moreover, \( \text{card}(F_{k,n}^0) = n/k \).

**Example 3:** Let \( U_7 = \{a_1, a_2, a_3, a_4, a_5, a_6, a_7\} \), \( k = 2 \), and \( m = 0 \).

A solution to our problem has size three:

\[ F_{2,7} = \{\{a_1, a_2\}, \{a_3, a_4\}, \{a_5, a_6\}\} \]

and \( \text{card}(F_{2,7}) = 7/2 = 3 \).

It is worth noting that if we modify condition (1) in Definition 1 so as to reason with subsets of \( U_{k,n} \) that have exactly \( m \) elements in common, then solving our problem becomes trivial as shown in Proposition 1.

**Proposition 1:** Using the same notation given in Definition 1, let the first condition be: \( \forall L, L' \in F_{k,n}^m, \text{Card}(L \cap L') = m \).

We certify that \( \text{Card}(F_{m,k,n}) = \binom{n}{k-m} \).

Without loss of generality, consider any subsets of \( U_{k,n} \) that have \( m \) elements in common (e.g. \( \{a_1, a_2, \ldots, a_m\} \)). Once we fix \( m \) elements in a subset of size \( k \geq m \), we choose the remaining \( k-m \) elements out of \( n-m \) elements. Therefore, the total number of subsets of size \( k \) we can build and that have \( m \) elements in common is equal to \( \binom{k-m}{n-m} \).

**Example 4:** Let \( U_{100} \), \( k = 20 \), and \( m = 10 \).

There are at most \( \binom{100-10}{20-10} \times \binom{10}{10} \) subsets of \( U_{100} \) of size 20 that have exactly 10 elements in common.

In the next section we show how to solve our problem using a brute-force approach.

IV. Brute Force

Given a set \( U_n \) of size \( n \), an integer \( k < n \) and a target \( 0 \leq m < k \), a straightforward algorithm for solving our problem consists in enumerating all the subsets of \( U_n \) of size \( k \). Then we process the set of all subsets of \( U_{k,n} \) as follows: for each subset we check whether the intersection of any couple of its elements is less than or equal to \( m \). If this is the case, and if the selected subset is the largest one we considered so far, then we update \( F_{m,k,n} \) accordingly.

**Proposition 2:** The running time of Algorithm 1 is

\[ T(n) = O(n^2 \times (\binom{n}{m})^2 \times 2^{\binom{k}{m}}) \]

**proof** Obviously, lines 1 and 2 require \( \Theta(\binom{n}{m}) = \Theta(2^{\binom{n}{m}}) \) steps. Next we determine the running time of the
Algorithm 1: Maximum (k,m)-subsets problems

Require: \( n, k, m \in \mathbb{N} \), where \( k < n \) and \( 0 \leq m < k \)
1: Compute \( U_{k,n} \)
2: Compute \( 2^{U_{k,n}} \)
3: \( F_{k,n}^m = \emptyset \) \{Initialization to the empty set\}
4: for all \( T_k \in 2^{U_{k,n}} \) do
5: for all \( L, L' \in S_{k,n} \), where \( L \neq L' \) do
6: if \( \text{Card}(L \cap L') > m \) then
7: break
8: end if
9: end for
10: if \( \text{Card}(T_k) > \text{Card}(F_{k,n}^m) \) then
11: \( F_{k,n}^m = T_k \)
12: end if
13: end for
14: return \( F_{k,n}^m \)

Nested two for loops.

Let \( T_k \in 2^{U_{k,n}} \). For each couple of elements in \( T_k \), we test whether they have more than \( m \) elements in common. If this is the case we discard this set. Otherwise (i.e. we have \( \forall L, L' \in T_k; |L \cap L'| \leq m \)), we compute the cardinal of \( T_k \) and, if necessary, update \( F_{k,n}^m \) (lines 10 and 11).

Note that computing the intersection of two sets of size \( k \) requires \( \Theta(k^2) \). Therefore, the test at line 6 requires at most \( O(k^2) \) operations. Moreover, we have to repeat this operation for any couple of elements in \( T_k \) (i.e. inner for loop at line 5), for a total of \( O(k^2) \times \Theta((l+1)/2) = O((kl)^2) \), where \( \text{Card}(T_k) = l \).

Since the for loop at line 4 is repeated \( \Theta(2^{C^k_n}) \) times, we deduce that the overall running time of the algorithm is \( T(n) = O((kl)^2 \times 2^{C^k_n}) \).

It is easy to see that \( S_{T_k} \in 2^{U_{k,n}} \) may contain up to \( C^k_n \) elements, which gives us \( l = O(C^k_n) \).

Before concluding, note that \( k \) may also depend on \( n \) (e.g. \( k = n/2 \)). Therefore the running time of Algorithm 1 is \( T(n) = O(n^2 \times (C^k_n)^2 \times 2^{C^k_n}) \).

Proposition 3: For \( k = \Theta(n) \), we need \( O(2^{n-1}) \) operations to compute \( C^k_n \).

Proof: Let \( k = \frac{n}{2} \). We have:

\[
C^k_n = \frac{n!}{k!(n-k)!} = \frac{n!}{\left(\frac{n}{2}\right)!(\frac{n}{2})!} = \frac{n!}{\frac{n}{2}! \times \frac{n}{2}!}.
\]

Recalling Stirling’s approximation for the factorial function (i.e. \( n! \approx \sqrt{2\pi n} \left(\frac{n}{e}\right)^n\)):

\[
C^k_n \approx \frac{\sqrt{2\pi (2^n)^n}}{\left(\sqrt{2\pi (2^n)^n}\right)^2} \approx \frac{\sqrt{2\pi n} (\frac{n}{e})^n}{\left(2\pi (\frac{n}{e})^n\right)^2} \approx \frac{\sqrt{2\pi n} n^n}{2n^2 \pi n \times 2^n} = \Theta\left(\frac{1}{n} \times 2^{n-1}\right) = O(2^{n-1})
\]

Corollary 1: A set \( U_n \) that contains \( n \) elements has \( O(2^n) \) subsets of size \( k = \Theta(n) \) each.

Corollary 2: In the worst case scenario, Algorithm 1 runs in

\[
T(n) = O(n^2 \times 2^{2n} \times 2^{n-1})
\]

where the leading factor is a double exponential.

Proposition 4: The maximum (k,m)-subsets problem is EXP-hard.

Proof: Let \( k = \Theta(n) \). The number of subsets of size \( k \) is obviously exponential with respect to \( n \).

Let \( m = k - 1 \). Our solution \( F_{k,n}^m \) contains all the subsets of \( U_n \) of size \( k \), and enumerating them requires exponential time.

V. NEXP-COMPLETENESS RESULT

We show in this section that our problem belongs to the class NEXP by reducing it to the succinct K-coloring problem.

Definition 2: We say that an ordered binary decision diagram OBDD represents the set \( U_n \) for an integer \( m < n \) if it is built as described below, where we assume that the levels of the OBDD are numbered from 1 (i.e. the root) to \( n+1 \). For all \( i \) where \( 1 \leq i \leq n \), level \( i \) contains \( 2^{i-1} \) internal nodes \( a_i \).

The last level consists of terminal nodes with values 0 or 1. Given such a node \( x \), we determine its value as follows: We consider the path from the root to \( x \). If the number of times we go to the left is less than or equal to \( m \), then we assign \( x \) the value 1; otherwise we assign it the value 0 (see Figure 2 (a) for an example).

Definition 3: Let an OBDD that represents a set \( U_n \) for an integer \( m \). We say that the OBDD accepts two subsets \( L \in U_n \) and \( L' \in U_n \) if we reach a terminal node with value 1 after following these steps: Starting from the root of the OBDD, and for all \( 1 \leq i \leq n \) we check to see if the processed node \( a_i \) belongs to both \( L \) and \( L' \). If this is the case then we go to the left, otherwise we go to the right.

In other words, an OBDD accepts two subsets \( L \) and \( L' \) if and only if \( \text{card}(L \cap L') \leq m \).

Examples 5: Let \( U_4 = \{a_1, a_2, a_3, a_4\} \) and \( m = 1 \). The OBDD in Figure 2 (a) (or its reduced form in part (b)) allows us to reason about the intersection of subsets of \( U_4 \).

Given \( L = \{a_1, a_3\} \) and \( L' = \{a_2, a_4\} \) for example, we check that: \( a_1, a_2 \) and \( a_4 \) do not belong to \( L \cup L' \), and that...
We indicate in Figure 4 the general form of a reduced ordinary binary decision diagram where the initial set has \( n \) elements. This efficient representation requires only \( O(n) \) space instead of the original \( O(2^n) \) space. We notice that we have only two final states (i.e. 0 and 1), and that the graph has two diagonals: one with the elements \( a_1, a_2, \ldots, a_{n-1} \), and the other one with the elements \( a_2, a_3, \ldots, a_n \).

Below is a pseudo-code that determines whether or not an OBDD accepts two subsets of \( U_n \).

**Algorithm 2** Return true if \( \text{card}(L \cap L') \leq m \), and false otherwise.

```
Require: OBDD representing \( U_n = \{a_1, \ldots, a_n\} \) for \( m = 1 \).
1. Let \( L \subseteq U_n, L' \subseteq U_n \), where \( \text{card}(L) = \text{card}(L') = k \).
2. (Optional) Sort the elements in \( L \) and \( L' \).
3. \( i = 1, \text{counter} = 0 \).
4. while \( (i \leq n) \) do
5. if \( a_i \in L \cap L' \) then
6. go to the left
7. \( i = i + 1, \text{counter} = \text{counter} + 1 \).
8. else
9. go to the right
10. \( i = i + 1 \).
11. end if
12. end while
13. if \( \text{counter} \leq m \) then
14. return 1.
15. else
16. return 0.
17. end if
```

**Proposition 5:** Determining if the intersection of two subsets of size \( k \) contains less than \( m \) elements can be done in \( T(n) = O(n \log n) \).

**Proof** The result can be easily derived from Algorithm 2. Indeed, sorting the subsets requires \( O(k \log k) \). The test on line 5 can be done in constant time since the subsets are sorted. For that we can simply use two counters \( j \) and \( j' \) for the subsets \( L \) and \( L' \), respectively. We compare \( a_i \) to \( a_j \in L \) and to \( a_{j'} \in L' \) in constant time, then increment the counters accordingly. Since the while loop is entered \( O(n) \) times, the overall running time of the algorithm is \( O(n \log k) \). In the worst case we have \( k = O(n) \), therefore \( T(n) = O(n \log n) \).

Note that if the subsets are not sorted, then the test on line 5 would require \( O(k) \) steps. The leading factor for the while loop (and also for the whole algorithm) becomes \( O(kn) = O(n^2) \).

We next prove that our problem is NEXP-complete by reducing to it the succinct K-coloring problem. For this purpose we consider graphs that have \( C_k^k \) nodes and where \( k = \Theta(n) \). We know from Proposition 3 that they contain an exponential number of nodes with respect to \( n \) (and therefore
w.r.t. \( k \)).

Intuitively, each node of the graph corresponds to a subset of \( U_n \) of size \( k \).

**Proposition 6:** Let \( K \) be some positive integer, and \( m, k, n \) such that \( 0 < m \leq k < n \). For any instance \( P_{m,k,n} \) of the maximum \((k,m)\)-subsets problem, there is a graph \( G_{m,k,n} \) such that if \( P_{m,k,n} \) has a solution of size \( K \) then \( G_{m,k,n} \) is \( K \)-colorable.

**proof** Our reduction \( R \) is straightforward and can be done in polynomial time with respect to the size of the input.

Intuitively, a node \( i \) of the graph \( G_{m,k,n} \) is the encoding of a distinct subset of \( U_n \) of size \( k \), that we denoted by \( S_i \) (recall Definition 1). Let \( P_{m,k,n} \) be an instance of the maximum \((k,m)\)-subsets problem. The reduction \( R \) associates to this instance the graph \( G_{m,k,n} \) with \( n \) vertices, where node \( i \) corresponds to the subset \( S_i \). We next consider the OBDD that represents the set \( U_n \) for \( n \) (as described in Definition 3). For any couple of nodes \( i \) and \( j \) of \( G_{m,k,n} \), we check whether the OBDD accepts the subsets \( S_i \) and \( S_j \). If this is the case (i.e. the subsets \( S_i \) and \( S_j \) have less than \( m \) elements in common), we add an edge between \( i \) and \( j \); otherwise we don’t.

Clearly, if \( P_{m,k,n} \) has a solution of size \( K \) then \( G_{m,k,n} \) is \( K \)-colorable.

**VI. CONCLUSION**

Given a set \( S \) of size \( n \), we tackled in this paper the problem of determining the largest number of subsets of \( S \) of size \( k \) that have at most \( m \) elements in common.

We showed that this problem is at least EXP-hard and at most NEXP-complete. Our main proof reduces the maximum \((k,m)\)-subsets problem to the succinct \( K \)-coloring problem, and uses very well-known data structures to represent graphs; namely ordered binary decision diagrams.

Besides showing whether this problem is EXP-complete or NEXP-complete, we intend to determine its tractable instances. More precisely, we would like to investigate whether there is a polynomial-time solution to our problem for some specific values of \( m, k, \) and \( n \).

**REFERENCES**


