

Oracle complexity of matroid intersection

One year ago I participated in a program called REU, where my supervisor was Kristóf Bérczi. In that month i really enjoyed thinking about the problems we faced. That is why when this autumn I had to search for a topic, I asked Kristóf if he has any interesting topic.

1 Introduction to matroids

A matroid is an abstract structure defined by an $(\mathcal{S}, \mathcal{F})$ pair, where \mathcal{S} is a finite set, and \mathcal{F} is a set of subsets of \mathcal{S} , which satisfies some axioms. This concept was first used by Hessler Whitney in 1933. Our goal with this concept is to understand certain properties of groups or rings in a more generic way. In particular, a matroid is somewhat of a generalization of the linear independency.

An other approach describes matroids as structures, on which for any weight function the greedy algorithm gives a good solution. It is well known that if we wanna find a maximal weighted spanning tree in a simple connected graph, with a weight function on it's edges, we would use a greedy algorithm: We choose edges one by one, always the biggest one possible, with one thing in mind: after each step we want to have a forest. It is easy to see that this indeed gives us a maximal weight spanning tree. However, in a bipartite graph, if we want to find a maximal weight pairing, this greedy approach won't work. This gives us the question, what are the needed properties for a greedy algorithm to work good. To discuss this, we need to define what exactly is a greedy algorithm, but we will not dive into that now, we will define matroids in the original way, suggested by Whitney.

1.1 Independency axioms and the rank

Definition 1.1. Let \mathcal{S} is a finite set and \mathcal{F} is the set of the so-called *independent* subsets of \mathcal{S} . We call the pair $\mathcal{M} = (\mathcal{S}, \mathcal{F})$ *matroid*, if \mathcal{F} satisfies the following three axioms:

1. $\emptyset \in \mathcal{F}$.
2. If $X \subseteq Y$ and $Y \in \mathcal{F}$, then $X \in \mathcal{F}$
3. For every $X \subset \mathcal{S}$ subset, all $K \in \mathcal{F}$, for which $K \subset X$ and K is the maximal in X , have the same number of elements

By the third independency axiom it is clear that for any $X \subset \mathcal{S}$ the cardinality of it's maximal independent subsets is equal. This motivates the following definition.

Definition 1.2. We call rank function of a matroid the following r function:

$$r(X) := \max(|Y| : Y \subseteq X \text{ and } Y \in \mathcal{F}) \tag{1}$$

The previous though makes it clear that r is well-defined.

Claim 1.3. *Let $K, N \in \mathcal{F}$ and $|K| < |N|$. Then there exists $x \in N - K$, for which $K + x \in \mathcal{F}$. This claim is equal to the third independency axiom.*

There are many other ways to define the matroids by its independent subsets by swapping the third axiom for something else, while the triple defines the same structure. These become useful, when working in different matroids, since in different structures it might be easier to check another axiom, then the other. However, we will not dive into this deeper.

1.2 Circle axioms

We call a set $X \subseteq \mathcal{S}$ a circle if every real subset of X is independent. We can define matroids also with it's circles.

Let \mathcal{C} be a set of subsets of \mathcal{S} .

1. $\emptyset \notin \mathcal{C}$
2. If $C_1, C_2 \in \mathcal{C}$ then $C_1 \not\subseteq C_2$
3. If $C_1, C_2 \in \mathcal{C}$ and $C_1 \neq C_2$ and $e \in C_1 \cap C_2$ then there exists $C \in \mathcal{C}$ such, that $C \subset C_1 \cup C_2 - e$

Claim 1.4. *If a family of sets \mathcal{C} satisfies the three previous axioms, then, the family of sets:*

$$\mathcal{F} = \{F : \nexists C \in \mathcal{C}, C \subset F\} \tag{2}$$

defines a matroid, where \mathcal{C} will be the circles of the matroid.

1.3 Basis axioms

We call a maximal independent set basis. The matroids rank will be the cardinality of it's basis. It is easy to see, that a set will be independent if and only if, it is a subset of a basis. Furthermore, this also means, that a matroid is determined by it's bases.

Definition 1.5. We call the following two properties basis axioms:

1. \mathcal{B} is non-empty
2. If $B_1, B_2 \in \mathcal{B}$ then for every $x \in B_1 - B_2$ there exists an $y \in B_2 - B_1$, for which $B_1 - x + y \in \mathcal{B}$

Claim 1.6. *A matroids bases satisfies the basis axioms. If a family of sets \mathcal{B} satisfies the basis axioms, then*

$$\mathcal{F} = \{F : \exists B \in \mathcal{B} F \subseteq B\} \tag{3}$$

satisfies the independency axioms.

1.4 Oracle complexity

We already saw, that a matroid's independent subsets are from the subsets of it's ground set. However, there are exponentially many subsets, so if we would want to store the information which subsets are independent, this might require an exponentially big space. This idea makes the next approach more understandable.

Let's say we have a ground set and we know there is a matroid over it, but we know nothing about it. However, there is an oracle, which knows everything about this matroid, as it knows it's independent subsets. We can ask the oracle a set to our liking, and the oracle answers this by saying if it is independent or not. Our goal is to find out some information about the matroid, and we want to do this with the fewest number of question possible.

However, there are other types of oracles. The previously defined oracle is called independency oracle. There is another oracle, with answers with yes and no, the basis oracle. To ask the oracle, we again ask a subset to our liking, and then the oracle's answer depends, whether the set is a basis or not.

There is also an oracle, which is not a 2-bit oracle. This one is called the rank oracle. We again ask a subset, but this time the oracle answers, with the rank of the set.

It is easy to see that the rank oracle is stronger than the independency oracle, since $X \in \mathcal{F}$ if and only if $r(X) = |X|$. However, there is even more connection between the rank and independency oracle.

Claim 1.7. *The rank and independence oracles are polynomially equal.*

What is the connection between the basis and independency oracles?

Claim 1.8. *We can make a basis oracle, from the independency oracle.*

Claim 1.9. *From the basis oracle, we can not produce a polynomial independency oracle.*

2 Matroid intersection problem

The matroid intersection problem is one of the most fundamental problems of combinatorial optimization. Given two matroids $\mathcal{M}_1 = (\mathcal{S}, \mathcal{F}_1)$ and $\mathcal{M}_2 = (\mathcal{S}, \mathcal{F}_2)$, where $|\mathcal{S}| = n$. The goal is to find an $X \in \mathcal{F}_1 \cap \mathcal{F}_2$ with the highest possible cardinality, denoted by r . This problem generalizes some important combinatorial optimization problems, such as bipartite matching, packing spanning trees, or arborecenses in directed graphs. Furthermore, it also has application in electrical engineering.

Starting with the works of Edmunds, many algorithms with polynomial query complexity have been studied. Edmunds developed the first polynomial-query algorithm, with $O(n^4)$ independence oracle queries. After him, Lawler gave an algorithm which requires $O(nr^2)$ queries. In 1986, Cunningham represented an algorithm, reducing the number of required queries to $O(nr^{\frac{3}{2}})$.

We quickly bumped into hard questions, therefore our goal this semester was to understand the problem on some easier cases. For example by supposing that $r = 2$, for which we got the results written in the next section.

3 Oracle complexity of the Matroid intersection, for $r \leq 2$

3.1 Upper bound

Theorem 3.1. *Given two matroids $\mathcal{M}_1 = (\mathcal{S}, \mathcal{F}_1)$, $\mathcal{M}_2 = (\mathcal{S}, \mathcal{F}_2)$. We can find a maximum common independence set with $3n - 1$ independence oracle queries, if $r(\mathcal{M}_1) = r(\mathcal{M}_2) = 2$.*

Proof. In the following we will give an algorithm, which will find a maximum common independence set using at most $3n - 1$ oracle queries.

The algorithm has the following steps:

1. Find a common independent element a .
2. Find an element b , for which $r_1(\{a, b\}) = 2$ and $r_2(\{b\}) = 1$.
3. Find an element c , for which $r_2(\{a, c\}) = 2$ and $r_1(\{c\}) = 1$.

First let's declare a few sets we will use during the algorithm. At the beginning let $T = S = \emptyset$ and $D = \mathcal{F}$. At a given moment the elements of T will be those, which we don't wanna ask about later on, and they are not special elements either. S will be the set of element, which we asked about, but we might ask about later on too. D is the set of elements, which we didn't asked about yet.

Now lets take a closer look at each step.

Step 1: We take an element of D , and ask the following: $\{a\} \in \mathcal{F}_1$.

If $\{a\} \notin \mathcal{F}_1$, we know that a cannot be part of any common independence set, so we won't ask any question on this element again and update $D := D - \{a\}$, $T := T + \{a\}$. Then we move to a new element of D and ask the same question.

If $\{a\} \in \mathcal{F}_1$, then we ask $\{a\} \in \mathcal{F}_2$. If $\{a\} \notin \mathcal{F}_2$, then similarly to the previous case, we update $D := D - \{a\}$, $T := T + \{a\}$, and continue the algorithm with a new element of D .

If $\{a\} \in \mathcal{F}_1$ and $\{a\} \in \mathcal{F}_2$, we found a common independent element and we can move to the next step, and update $D := D - \{a\}$

In this step we ask every element at most twice.

We need to consider, what happens there is no such a , for which $\{a\} \in \mathcal{F}_1$ and $\{a\} \in \mathcal{F}_2$. In this case the only common independence set is the empty set, and with the previous method we will find this out at most $2n$ queries.

Step 2: We take an element $b \in D$, and ask the following: $\{a, b\} \in \mathcal{F}_1$.

If $\{a, b\} \notin \mathcal{F}_1$, then we update $D := D - \{b\}$, $S := S + \{b\}$, and move to a new element of D .

If $\{a, b\} \in \mathcal{F}_1$, then we ask $\{b\} \in \mathcal{F}_2$. If $\{b\} \notin \mathcal{F}_2$, then we update $D := D - \{b\}$, $T := T + \{b\}$, and move to a new element of D .

If $\{b\} \in \mathcal{F}_2$, we just found a wanted b , and $D := D - \{b\}$.

What happens if there is no such b . If $\{a, b\} \notin \mathcal{F}_1$, for every b , then \mathcal{M}_1 , has a rank of 1, since a is not part of any bigger independent set. Therefore a common independent set have a rank of at most 1, and we already found such set: $\{a\}$.

In the other case we found out, that there are some elements for which $\{a, b\} \in \mathcal{F}_1$, but $\{b\} \notin \mathcal{F}_2$. Since $r(\mathcal{M}_1) \leq 2$, \mathcal{M}_1 must have the following structure:

$S = A_1 \dot{\cup} A_2 \dot{\cup} L$, for which:

1. $l \in L : \{l\} \notin \mathcal{F}_1$
2. $x \notin L : \{x\} \in \mathcal{F}_1$.
3. $x \in A_i, y \in A_j$ if $i = j : \{x, y\} \notin \mathcal{F}_1$, if $i \neq j : \{x, y\} \in \mathcal{F}_1$

Since A_1 and A_2 are symmetric for each other, we can suppose that $a \in A_1$.

From the partition of S it is easy to see, that $\forall S \subset \mathcal{S} : |S| = 2 : |S \cap A_2| = 1$. However $\forall b \in A_2 : \{b\} \notin \mathcal{F}_2$. This means, that \mathcal{M}_1 and \mathcal{M}_2 cannot have a common, two-element, independent set. Since we already found a common independent element, and we would be ready at this point.

Step 3: We take an element $c \in D$ and ask if $\{a, c\} \in \mathcal{F}_2$.

If $\{a, c\} \notin \mathcal{F}_2$, we update $T := T + \{c\}$, $D := D - \{c\}$, and move to a new element of D .

If $\{a, c\} \in \mathcal{F}_2$, we ask $\{c\} \in \mathcal{F}_1$. If $\{c\} \notin \mathcal{F}_1$, we update $T := T + \{c\}$, $D := D - \{c\}$, and move to a new c .

If $\{c\} \in \mathcal{F}_1$, we found the needed c , and $D := D - \{c\}$.

However if $D = \emptyset$ before we would find such c , we repeat Step 3, but this time using the elements of S instead of D .

If we do not find such c , neither by repeating this step over the elements of S , we would get to the same position, as in Step 1, resulting there would be no common, two-element, independent set.

Let's have a look what structure do we have after Step 3. We have three elements a, b, c , from which neither is a loop in any of the two matroids. Also we know that:

Case 1: $\{a, b\} \in \mathcal{F}_1$ and $\{a, c\} \in \mathcal{F}_2$. With two more queries we can find a common independent set with two elements. Let's ask $\{a, b\} \in \mathcal{F}_2$ and $\{a, c\} \in \mathcal{F}_1$. If we get a 'yes' answer on any of these questions, we just found a common, two-element, independent set.

However we can get a 'no' answer on both of them. In this case we use the matroid axiom, which says if there is a K, N independent sets, for which $|K| < |N|$, then $\exists x \in N - K$ for which $K + x \in \mathcal{F}$. In \mathcal{M}_1 let $K = \{c\}$ and $N = \{a, b\}$. Using the axiom one of $\{c, a\}$ and $\{c, b\}$ must be independent. But $\{a, c\} \notin \mathcal{F}_2$, therefore $\{b, c\}$ must be independent. Similarly we get that $\{b, c\} \in \mathcal{F}_2$. Therefore we found a common, two-element independent set.

Case 2: Compared to Case 1 we also know that $\{a, c\} \notin \mathcal{F}_1$, which means we need one less question this time, compared to Case 1.

Number of queries

We will show that the number of queries is at most $3n - 1$ and also there exist such $\mathcal{M}_1, \mathcal{M}_2$ for which this algorithm might take exactly $3n - 1$ queries.

By finding a we used 2 queries on every used element. After that, for the right b we used also two queries. On the other elements we used at most three queries: two if we moved that element into T in Step 2, and two or three if it was moved to T in Step 3, or either it is c itself.

If the number of questions we asked about c to find it is 2, then we used $3n - 3$ queries at most at this point. Since for the three special elements we used two questions, and for any other element at most three. However in the final we used 2 more queries, which gives us a $3n - 1$ upper bound in this case.

If the number of questions we asked about c to find it is three, then we used $3n - 2$ queries. But in this case we only needed one extra query in the end, which gives the same upper bound of $3n - 1$.

Let $x \in \mathcal{S}$, such that $\{x\} \in \mathcal{F}_1$ and $\{x\} \in \mathcal{F}_2$. We start the algorithm by this x , so we just found the right a we were looking for in 2 steps.

After this, we get that $\{a, b\} \notin \mathcal{F}_1$, for every other element, until the last one, and we also get that that last element is not a loop in \mathcal{M}_2 . So in this step we used $(n - 1) + 1$ queries.

In step three then we get that $\{a, c\} \in \mathcal{F}_2$, but $\{c\} \notin \mathcal{F}_1$, until the last one, for which we get that $\{a, c\} \in \mathcal{F}_2$, and $\{c\} \in \mathcal{F}_1$. In this step we then used $2(n - 2)$ queries. Then in the final we ask $\{a, b\} \in \mathcal{F}_2$, and after this we will know the maximum, common, independent set.

In total we used $2 + (n - 1) + 1 + 2(n - 2) + 1 = 3n - 1$ queries.

□

3.2 Lower bound

Theorem 3.2. *There exists a bad entity, which can prevent us from finding a common independent set in less than $2n$ queries.*

Proof. Let the bad entity give us the information that $r(\mathcal{M}_1) = r(\mathcal{M}_2) = 1$, before we begin asking. By this information, it is useless to ask about any set which has at least two elements, since we know it is not independent. This means we have $2n$ useful questions: asking any element on its own, in one of the two matroids.

The bad entity's strategy will be the following. If we ask about an element for the first time, it will say that it is independent. However, when we ask about the element for the second time, it will say that it is not independent.

With this strategy, we need to ask on every element twice, since every twice-asked element cannot be a common independent set. However, on every other element we cannot decide it yet.

□

3.3 Other results

During the semester, we faced with some other fundamental questions, such as what is the required number of queries, to find a basis of a matroid, using rank oracles.

Claim 3.3. *Given a matroid with a rank of 1. We can find a basis, which in this case is an independent element, in $\lceil \log(n) \rceil$ queries. However, this upper bound is also sharp.*

Theorem 3.4. *Given a matroid with a rank of r . There exists an algorithm, which requires $r \lceil \log_r(n) \rceil$ query moves at most to find a basis.*