Oracle complexity of matroid intersection

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 a generalization of linear independency.

Another approach describes matroids as structures for which, for any weight function, the greedy algorithm gives a good solution. It is well known that if we want to find a maximum-weight spanning tree in a simple connected graph, with a weight function on its 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 maximum-weight spanning tree. However, in a bipartite graph, if we want to find a maximum-weight pairing, this greedy approach will not work. This gives us the question, what are the needed properties for a greedy algorithm to work well. 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 axoims and the rank

Definition 1.1. Let $\mathcal{S}$ be 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 axoims:

1. $\emptyset \in \mathcal{F}$.
2. If $X \subseteq Y$ and $Y \in \mathcal{F}$, then $Y \in \mathcal{F}$
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 its maximal independent subsets is equal. This motivates the following definition.

Definition 1.2. We call the 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 definition makes it clear that r is well defined.

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.

1.2 Oracle complexity

We already saw that a matroids independent subsets are from the subsets of its ground set. However, there are exponentially many subsets, so if we want to store the information about which subsets are independent, this might require an exponentially large space. This idea makes the next approach more understandable.

Let us say that we have a ground set and we know there is a matroid on it, but we know nothing about it. However, there is an oracle that knows everything about this matroid, as well as its 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 the independency oracle. There is an 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 on wether 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.

2 Matroid intersection problem

The matroid intersection problem is one of the most fundamental problems of combinatorial optimalization. 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 arborecsenses in directed graphs. Furthermore, it also has applications in electrical engineering.

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

Theorem 2.1. *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 $r(\mathcal{M}_1) = r(\mathcal{M}_2) = 2$.*

Theorem 2.2. *There exists a bad entity, which can prevent us from finding a common independent set in less than $2n$ queries.*

Finding a maximal independent set of the intersection turned out to be really difficult even with the rank restrictions. Therefore, in this semester, we shifted our focus to finding a basis in a matroid.

3 Finding a basis

3.1 Independence Oracle

Theorem 3.1. *We can find a basis with n queries using the independence oracle.*

This is a well-known fundamental result of matroid theory. It comes from the theorem that the greedy algorithm works well on matroids. This gives us an upper bound for the number of oracle calls needed, but what can we say about the lower bound?

Claim 3.2. *There is no algorithm that uses at most $n - 1$ queries.*

Proof. Let $M_1 = (\mathcal{F}, S)$, where $\mathcal{F} = \{\emptyset\}$, and let M_2 be a matroid with exactly one independent element.

We will prove that we cannot distinguish these two matroids from each other in at most $n - 1$ queries.

Before starting the algorithm, we can assume that an entity helps us by telling us that every set with cardinality at least 2 is dependent. Therefore, we only ask about sets that contain exactly one element.

However, it can happen that for the first $n-1$ questions we get the answer *no*. Therefore, we found at most $n-1$ many dependent elements, but there is still at least one element, which can still be independent or dependent.

Since we cannot distinguish M_1 and M_2 in at most $n-1$ queries, there is no algorithm that finds a basis using at most $n-1$ queries. □

Remark 3.3. Note that in the previous theorem we did not make any conditions on the matroid itself.

For example, if we know that the rank of the matroid is 1, we can find a basis with $n-1$ queries. To do this, we ask about the elements one by one. After $n-1$ queries, we either got a *yes* answer, in this case we just found a basis, or all answers were *no*. However, since the rank of the matroid is 1, therefore, the only remaining element, which we did not ask about yet, must be independent, so that is a basis.

Note that this idea works for any assumption, where we know the rank of the matroid. We just run the greedy algorithm, and after $n-1$ we either already have an independent set of size r or we have an independent set of size $r-1$. However, in the latter case, we know that by adding the last remaining element to the set of size $r-1$ we must have a basis.

In a later result, we will also see that for some r, where we know that the rank of the matroid is r, there exist even faster algorithms, to find a basis.

3.2 Rank oracle

Claim 3.4. *If there is an algorithm that uses k independence oracle calls to answer a problem, then there exists an algorithm that uses k rank oracle calls to answer the same question.*

This claim comes from the fact that the rank oracle is stronger than the indepence oracle. What does *stronger* mean in this case?

An $X \subset S$ is independent if and only if $r(X) = |X|$. Therefore, with every rank oracle call, we can also decide if the given set is independent or not.

Corollary 3.5. *We can find a basis using n rank oracle calls. If we know the rank of the matroid, we can also do it in $n-1$ queries.*

However, this result can be strengthened since the rank oracle is stronger than the independence oracle.

Claim 3.6. *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.*

Proof. We will repeat the following step:

If we want to find an independent element in the set R we ask about the set T, where $T \subset R$ and $|T| = \lceil |R|/2 \rceil$.

If $r(T) = 1$, then we repeat this step on the set of T instead of R.

If $r(T) = 0$, then we repeat this step on the set $R - T$.

In this algorithm, in each step we halve the number of elements which we are interested in, so it takes $\lceil n/2 \rceil$ steps to finish.

However, there is no algorithm that can find an independent element in less than $\lceil n/2 \rceil$ queries.

We can illustrate the running of the algorithm in a decision tree. In this case, since the rank of the matroid is 1, every set has a rank of 0 or 1, therefore, every edge has 2 outgoing edges at most.

Now, let us estimate how many leaves the tree must have. There are exactly n many matroids that have exactly one independent element. However, each of these matroids must go by a different path in the tree when we run the algorithm, otherwise we could not distinguish two such matroids. Therefore, the number of leafs must be at least n.

However, since each edge has at most two outgoing edges, the depth of the decision tree must be at least $\lceil log_2(n) \rceil$. □

Following the same idea, we can make lower and upper bounds on the number of queries needed to find a basis in matroids of rank r, using rank oracle.

Claim 3.7. *Given a matroid with a rank of r. We can find a basis in $r \cdot \lceil log_2(n) \rceil$ queries. There is no algorithm that finds a basis in less than $\lceil log_{r+1}(\binom{n}{r}) \rceil$.*

Proof. First, we will prove the lower bound. We follow the same idea, when we had a matroid with rank of 1.

This time in the decision tree each tree has at most $r+1$ outgoing edges. The number of leafs must be at least $\binom{n}{r}$, since there are $\binom{n}{r}$ many matroids of rank r with exactly one basis, and each of these matroids must have a different path in the decision tree.

Therefore, the depth of the decision tree must be at least $\lceil log_{r+1}(\binom{n}{r}) \rceil$.

Now we will show an algorithm that uses $r \cdot \lceil log_2(n) \rceil$ many rank oracle queries to find a basis.

In the first step, we will find an independent element. We do this on by same principle as in 3.6. This time we halve T, if $r(T) \geq 1$ everything else works in the same way.

With this we found an independent element e_1 and let $I_1 = \{e_1\}$. Now we want to find an element $e_2 \in S - I_1$, for which $I_1 + e_2$ is independent.

We do this with the same process with a slight difference. The $(i+1)$th step works the following way:

If we want to find the next element in the set R we ask about the set $T + I_i$, where $T \subset R$ and $|T| = \lceil |R|/2 \rceil$.

If $r(T + I_i) \geq i + 1$, then we repeat this step on the set of T instead of R.

If $r(T + I_i) = i$, then we repeat this step on the set $R - T$.

This will give us an element e_{i+1}, for which $I_i + e_{i+1} := I_{i+1}$ is independent.

Repetition of this step r times yields an independent set with size r, which is a basis.

The running time of this algorithm is $r \cdot \lceil log_2(n) \rceil$, since in each step we made $log_2(n)$ oracle calls.

□

Fortunately, this algorithm can be improved. So far, we really did not use the rank of the matroid. We just found the elements one by one, which is the fastest if the rank is 1, but not when the rank is greater.

Claim 3.8. *Given a matroid of rank r. We can find a basis with at most $r \cdot (log\lceil \frac{n}{r} \rceil + 1)$ queries.*

Proof. First, we divide the ground set S into r disjoint sets of the same size. Let us call these sets $S_1, S_2, ..., S_r$.

The ith step of the algorithm works the following way:

First, we have an independent set I_{i-1}, which we want to expand. At $i = 0$, $I_{i-1} = \{\emptyset\}$

First we ask for the rank of $I_{i-1} + S_i$.

If $r(I_{i-1} + S_i) \geq r(I_{i-1})$, then we find a set $R_i \subset S_i$ with a size of $r(I_{i-1} + S_i) - r(I_{i-1})$, such that $I_{i-1} + R_i$ is independent. Then $I_i = I_{i-1} + R_i$. And we go to the $i+1$th step.

We can do this by the same principle that we used in 3.7, with $(r(I_{i-1} + S_i) - r(I_{i-1})) \cdot log_2(|S_i|)$ many oracle calls.

If $r(I_{i-1} + S_i) = r(I_{i-1})$, this means that I_{i-1} is already a maximal independent set in $I_{i-1} + S_i$, so we can continue with the $i+1$th step, where $I_i = I_{i-1}$.

After n steps we will find a basis I_r. How many oracle calls did we make?

In the second case, we made 1 oracle call. In the first case, we made $(r(I_{i-1}+S_i)-r(I_{i-1})) \cdot log_2(|S_i|)+1$ oracle calls.

However, $log_2(|S_i|) \leq log\lceil \frac{n}{r} \rceil$, by construction. So, the number of oracle calls is not greater than $\sum_{i=1}^{r}(r(I_{i-1} + S_i) - r(I_{i-1})) \cdot log\lceil \frac{n}{r} \rceil + 1$.

Note that $\sum_{i=1}^{r}(r(I_{i-1} + S_i) - r(I_{i-1})) = r(I_r) - r(\emptyset) = r$.

Therefore, the number of oracle calls is at most $r \cdot (log\lceil \frac{n}{r} \rceil + 1)$. □

Remark 3.9. Note that $r \cdot (log\lceil \frac{n}{r} \rceil + 1) \leq n$ only if $r \leq n/2$. Therefore, this previous algortihm is slower than the greedy algorithm, for $r > n/2$.

Claim 3.10. *Given a matroid of rank r. We can find a basis with at most $(n-r) \cdot (log\lceil \frac{n}{n-r} \rceil + 1)$ queries.*

Proof. The algorithm for this result works in a similar way as in 3.2. In the previous proof, we always wanted to expand an independent set one by, until it had a size of the basis. In this case, we want to eliminate elements one by one in such a way that the size of the largest independent set of the remaining set does not decrease.

Divide the ground set S into $(n-r)$ many sets which have the same cardinality.

Let $I_0 = \emptyset$.

In the ith step we, do the following:

We ask $r(I_{i-1} + S_i)$. If $r(I_{i-1} + S_i) = |I_{i-1} + S_i|$, we move to the $i+1$th step, otherwise, we know that we can eliminate some elements in S_i. Then let $T \subset S_i$, where $|T| = \lceil |S|/2 \rceil$. Next, we ask $r(I_{i-1} + T)$.

If $r(I_{i-1}+T) < |I_{i-1}+T|$, then we continue with halving T now instead of S_i and ask the same question.

If $r(I_{i-1}+T) = |I_{i-1}+T|$, then we halve $S-T$ into R, but also in the next question we ask $r(I_{i-1}+T+R)$, instead of asking the rank of $I_{i-1} + R$.

In this way using $\log(|S_i|)$ we can eliminate one element of S_i, such that in the remaining set the size of the largest independent set does not decrease.

Therefore, with $(n-r)(\log(\frac{n}{n-r}) + 1)$ rank queries we can reduce the size of the remaining set to r, so it will be independent. □

3.2 together with 3.2 ensures that we can find a basis with at most n many rank oracle queries, and if $r \neq n/2$ then this is strictly faster.

Note that in 3.2 we can replace the rank oracle with the independence oracle. This gives us the following result.

Claim 3.11. *Given a matroid of rank r. We can find a basis using $(n-r)(\log(\frac{n}{n-r}) + 1)$ independence oracle calls.*

Remark 3.12. Note that 3.11 uses less oracle calls than the greedy algorithm.

4 Different oracles, and their connection

We mainly focused on the independent and rank oracles. However, different oracles might become useful in different problems, so it is important to understand the fundamentals of the connection with some of the fundamental oracles.

We examine some fundamental results from [2].

Definition 4.1. Let S be a ground set and $\mathcal{M}$ be a matroid on S. I list some concepts without a precise definition:

circuit = minimal dependent set

spanning set = superset of a basis

girth = $girth(F) = min\{|C| : C \subset F, C \text{ dependent}\}$ if it is meaningful, ∞ otherwise

closure = $closure(F) = \{e \in S : r(S+e) = r(S)\}$

flat = set which equals its closure

hyperplane = maximal flat different from S

Definition 4.2. The independence, basis, circuit, spanning, flat, hyperplane oracles are a mapping of 2^S into $\{YES, NO\}$.

The rank and girth oracles are mappings into $\{0, 1, ..., |S|, \infty\}$.

The closure oracle is a mapping into 2^S.

Theorem 4.3. *The girth oracle is stronger than the independent oracle. The rank, independent, spanning and closure oracles are polinomially equivalent. The basis, flat, hyperplane, circuit oracles are weaker than the independent oracle.*

4.1 Port oracle

In [2] the port oracle was not studied, therefore, in the following part we look at some results about the port oracle from [1].

Definition 4.4. Let $\mathcal{M}$ be a matroid with a distinguished element e. The *port* of $\mathcal{M}$ with respect to e is the set of circuits of $\mathcal{M}$ containing e.

A *port oracle* for $\mathcal{M}$ with respect to e reports whether or not a given subset contains a circuit containing e.

In [1] they present two port oracle algorithms for two problems. First, they show how to compute an e-based ear decomposition of a matroid.

Definition 4.5. A *partial ear decomposition* of a matroid $\mathcal{M}$ is a nonompty sequence $C_1, C_2, ..., C_k$ of circuits of $\mathcal{M}$ such that for every $i \in \{2, ..., k\}$ the following thhree properties hold:

1. $C \cap (C_1 \cup ... \cup C_{i-1}) \neq \emptyset$
2. $C - (C_1 \cup ... \cup C_{i-1}) \neq \emptyset$
3. no circuit C_i' of $\mathcal{M}$ setisfying the first two properties also has $C_i' - (C_1 \cup ... \cup C_{i-1})$ properly contained in $C_i - (C_1 \cup ... \cup C_{i-1})$

An *ear decomposition of a* $\mathcal{M}$ is a partial ear decomposition in which $C_1 \cup ... \cup C_k = S$

Claim 4.6. *Let $\mathcal{M}$ be a matroid and e an element of the ground set S. There exists an algorithm that finds an e-based ear decomposition with $O(|S|^2)$ port oracle calls.*

Using that ear decomposing algorithm as a subroutine, we can also simulate an independence oracle.

Claim 4.7. *Let $\mathcal{M}$ be a matroid on the ground set S. There exists an algorithm that can answer the question: Is F independent?, where $F \subset S$, and the algorithm uses $O(|S|^4)$ many port oracle calls.*

These results show us that the port oracle is polinomially equivalent to the independence oracle. However, since it takes $O(|S|^4)$ port oracle calls to simulate an independence oracle call, this oracle is not really useful when it comes to finding a basis.

References

[1] Collette R Coullard and Lisa Hellerstein. Independence and port oracles for matroids, with an application to computational learning theory. *Combinatorica*, 16(2):189–208, 1996.

[2] Dirk Hausmann and Bernhard Korte. Algorithmic versus axiomatic definitions of matroids. In *Mathematical Programming at Oberwolfach*, pages 98–111. Springer, 2009.