Oracle complexivity of matroid intersection

Viktor Csaplár
Supervisor: Kristóf Bérczi

Introduction to matroids

Definition

Let $\mathcal{S}$ be a finite set, and let $\mathcal{F}$ be 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:

- $\emptyset \in \mathcal{F}$
- If $X \subseteq Y$ and $Y \in \mathcal{F}$, then $X \in \mathcal{F}$
- For every $X \subseteq \mathcal{S}$ subset, all $K \in \mathcal{F}$, for which $K \subseteq X$ and K is maximal in X, have the same cardinality

Definition

We call r the rank function of the matroid, if
$r(X) := max(|Y| : Y \subseteq X \; and \; Y \in \mathcal{F})$

Oracle

- There are more then $poly(|\mathcal{S}|)$ different matroids
- An oracle knows the matroid
- In a subroutine we can give the oracle any subset, and it returns yes or no, depending on if the subset is independent or not
- Rank oracle...

Matroid intersection problem

- Given two matroids $\mathcal{M}_1 = (\mathcal{S}, \mathcal{F}_1)$ and $\mathcal{M}_2 = (\mathcal{S}, \mathcal{F}_2)$, over the same ground set
- Goal: find an $X \in F_1 \cap \mathcal{F}_2$ with the highest possible cardinality
- Known results:
  - Upper bound: $\mathcal{O}(nr^{\frac{3}{2}})$
  - Lower bound: $(log_2 3)n - o(n)$ for $r = n/2$

Results from the previous semester

Theorem ($r \leq 1$)

Given two matroids $\mathcal{M}_1 = (\mathcal{S}, \mathcal{F}_1)$ and $\mathcal{M}_2 = (\mathcal{S}, \mathcal{F}_2)$ over the same ground set, with rank at most 1.
No algorithm that performs fewer than $2n$ queries can solve the problem.
However, there exists an algorithm, using $2n$ queries.

Theorem ($r \leq 2$)

Given two matroids $\mathcal{M}_1 = (\mathcal{S}, \mathcal{F}_1)$, $\mathcal{M}_2 = (\mathcal{S}, \mathcal{F}_2)$. There is an algorithm to find a maximum common independence set, using $3n - 1$ independence oracle queries, if $r(\mathcal{M}_1) \leq 2$ and $r(\mathcal{M}_2) \leq 2$.

Independence oracle to find basis

Theorem

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

Theorem

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

Rank oracle to find a basis

Theorem

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

Theorem

Given a matroid with rank of r*. There exists an algorithm, that requires* $r \lceil log(n/r) + 1 \rceil$ *queries, at most to find a basis.*

Theorem

Given a matroid with rank of r*. There exists an algorithm, that requires* $(n - r) \lceil log(n/n - r) + 1 \rceil$ *queries, at most to find a basis.*

Independence oracle to find basis II.

Theorem

Given a matroid with rank of r. There exists an algorithm, that requires $(n-r)\lceil log(n/n-r)+1\rceil$ queries, at most to find a basis.
Note, that for $r \geq \frac{n}{2}$ this algorithm is faster than the greedy algorithm.

Thank you for you attention