CS 2429 - Foundations of Communication Complexity

Lecturer: Sandro Feuz

1 Dynamic Problems

In the setting of dynamic problems we are given some initial state of the data and have to support different operations on it. Operations are either updates to or queries on the data and we are interested in giving lower bounds for the operations, assuming we had some time to pre-process the initial data. Some examples of dynamic problems are:

- **Dynamic Shortest Path**
  - Given a directed graph G.
  - Updates consist of insertions and deletions of edges.
  - Queries ask for the shortest path between two given nodes u and v.
- **Maximum Matrix Element**
  - Given an integer matrix M.
  - Updates increment either a whole row or a whole column of M by one.
  - Queries ask for the maximum element of M.

Attempts for lower bounds often looked at the *cell-probe model*, where we assume computation is free and are only interested in the number of memory cells accessed. Clearly every lower bound in the cell-probe model also holds in the *Word RAM-model*.

Let N be the size of the inputs to the problems (i.e. the size of the graph or the size of the matrix, in the problems above), then we do not have solutions with pre-processing time $\mathcal{O}(N \cdot \text{poly}\log(N))$ and time per operation of $\mathcal{O}(\text{poly}\log(N))$. Neither do we have matching lower bounds and especially in the *cell-probe model*, the best we can do is $\Omega(\log(N))$ per operation.

Since the search for direct lower bounds of dynamic problems was rather slow, it is worth exploring other paths. We define a dynamic version of the set-disjointness problem, and will use it as a proxy and show that lower bounds for it would translate to many dynamic problems. While we cannot actually prove the necessary lower bounds for this new problem either, we have reason to believe that they hold since we can show that they are implied by a well accepted conjecture.

1.1 The Multiphase Problem

Definition [Pătraşcu] The multiphase problem consists of three phases and is a dynamic version of set-disjointness. In each phase we have access to all the memory written in the previous phases.

Phase I : Given k sets $S_1, \ldots, S_k \subseteq [n]$ and time $\mathcal{O}(nk \cdot \tau)$ to process them.

Phase II : Given another set $T \subseteq [n]$ and time $\mathcal{O}(n \cdot \tau)$ to process it.

Phase III : Given $i \in [k]$, decide whether $S_i \cap T = \emptyset$ in time $\mathcal{O}(\tau)$.

Note that τ serves as a proportionality factor w.r.t the input size of each phase. Clearly the only interesting cases are $\tau < n$, otherwise the computation in phase III can just compare every element in S_i and T. The following conjecture about the hardness of the multiphase problem is currently open.

Conjecture 1 (Pătraşcu) *There exist $\gamma > 1, \delta > 0$ such that for $k = \Omega(n^\gamma)$ any solution to the multiphase problem has $\tau = \Omega(n^\delta)$ (in the Word RAM Model).*

The conjecture says that for k big enough but still polynomial in n, we cannot solve the multiphase problem in $\text{poly}\log(n)$ time.

1.2 Reductions to dynamic problems

Theorem 2 (Pătraşcu) *The dynamic shortest path problem does not have a $\mathcal{O}(N\text{poly}\log(N))$ pre-processing time and $\mathcal{O}(\text{poly}\log(N))$ time per operation solution under the multiphase hardness conjecture (Conj. 1), where N denotes the size of the input graph.*

Proof We assume for contradiction that we have a solution for the dynamic shortest path problem with pre-processing time $\mathcal{O}(N\text{poly}\log(N))$ and time per operation $\mathcal{O}(\text{poly}\log(N))$ and show how to solve the multiphase problem efficiently ($\tau = \mathcal{O}(\text{poly}\log(n))$.

Phase I : Given $S_1, \ldots, S_k \subseteq [n]$, create the graph in Fig. 1 in $\mathcal{O}(nk \cdot \text{poly}\log(nk))$ time ($N = nk$).

Phase II : Given set $T \subseteq [n]$, add the edges depicted in Fig. 2. Taking time $\mathcal{O}(n \cdot \text{poly}\log(nk))$.

Phase III : Given $i \in [k]$, test if the shortest path from S_i to T is 2 and answer yes. Otherwise answer no. This takes time $\mathcal{O}(\text{poly}\log(nk))$.

In the resulting graph there is a path from S_i to T iff $S_i \cap T \neq \emptyset$. We get for $k = \Omega(n^\gamma)$ a value of $\tau = \mathcal{O}(\text{poly}\log(nk)) = \mathcal{O}(\text{poly}\log(n))$, contradicting $\tau = \Omega(n^\delta)$.

Theorem 3 (Pătraşcu) *The maximum matrix element problem does not have a $\mathcal{O}(N\text{poly}\log(N))$ pre-processing time and $\mathcal{O}(\text{poly}\log(N))$ time per operation solution (N being the size of the input matrix) under the multiphase hardness conjecture (Conj. 1).*

Proof We again assume that we have an efficient solution for the maximum matrix element problem with pre-processing time $\mathcal{O}(N\text{poly}\log(N))$ and time per operation $\mathcal{O}(\text{poly}\log(N))$, and construct an efficient solution for the multiphase problem.

Phase I : Given $S_1, \ldots, S_k \subseteq [n]$, construct a matrix $M \in \{0,1\}^{k \times n}$ (note $N = nk$) where $M[i][j] = 1$ iff $j \in S_i$. This takes time $\mathcal{O}(nk \cdot \text{poly}\log(nk))$.

Phase II : Given set $T \subseteq [n]$, increment the columns in T. Taking time $\mathcal{O}(n \cdot \text{poly}\log(nk))$.

Phase III : Given $i \in [k]$, increment the i-th row and then test if the maximum element is 3 and answer yes if it is. Answer no otherwise. Taking time $\mathcal{O}(\text{poly}\log(nk))$.

The resulting maximum element is either 2, if $S_i \cap T = \emptyset$ or 3, if $S_i \cap T \neq \emptyset$. We again get $\tau = \mathcal{O}(\text{poly}\log(n))$ for $k = \Omega(n^\gamma)$.

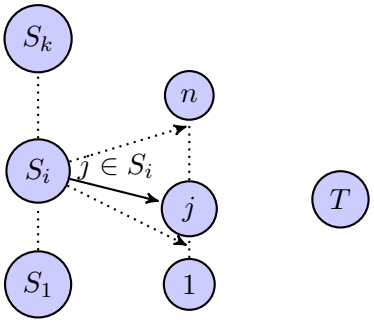

Figure 1: Graph $G = (V, E)$ after Phase I
$V = \{S_1, \ldots, S_k, 1, \ldots, n, T\}$,
$E = \{(S_i, j) | j \in S_i\}$. All edges weight 1.

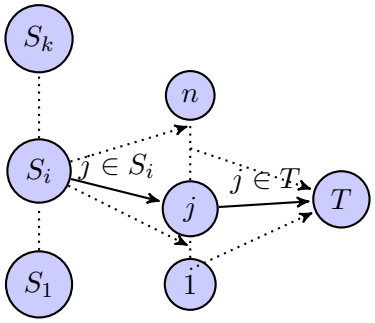

Figure 2: Graph after Phase II.
Add edges (j, T) for each $j \in T$.
All edges weight 1.

1.3 Supporting the Multiphase Hardness Conjecture

One reason we believe the multiphase hardness conjecture is because we can show that it holds, if the well known 3-SUM problem has a $\Omega(n^2)$ lower bound.

Definition [Erickson'95] 3-SUM problem

- Input: A set of numbers S of size $n = |S|$.
- Output: A triple $s_i, s_j, s_k \in S$, such that $s_i + s_j + s_k = 0$.

Theorem 4 *3-SUM is solvable in $\mathcal{O}(n^2)$.*

Proof The following algorithm solves 3-SUM in time $\mathcal{O}(n^2)$.

```
3-SUM(S)
 1  Sort s_1 ≤ s_2 ≤ ··· ≤ s_n in increasing order.
 2  for i = 1, ..., n
        // Looking for triples s_i + s_j + s_k = 0, with i ≤ j ≤ k.
 3      j = i.
 4      k = n.
 5      while k ≥ j
 6          if s_i + s_j + s_k = 0
 7              return 1.
 8          if s_i + s_j + s_k > 0
 9              k−−.
10          else
11              j++.
12  return 0.
```

The algorithm clearly runs in time $\mathcal{O}(n^2)$ ($\mathcal{O}(n \log(n))$ for sorting and $\mathcal{O}(n)$ for each i). Moreover before each step of the inner **while** loop, the invariant holds that there are no solutions including elements s_i and s_l for $l \notin \{j, \ldots, k\}$.

Conjecture 5 (Gajentaan,Anka'95) *3-SUM has a $\Omega(n^2)$ lower bound.*

Theorem 6 (Pătraşcu) *Under the 3-SUM lower bound conjecture (Conj. 5) the multiphase problem with $k = \Omega(n^{2.5})$ requires $\tau \geq n^{0.5-o(1)}$ in the Word RAM model.*

Proof [Outline] The proof first transforms a lower bound for 3-SUM to a lower bound for a version of 3-SUM called Convolution-3-SUM. From there it transforms the lower bound to the triangle reporting problem in a graph. Finally it shows how triangle reporting can be implemented using the multiphase problem, proving $\tau \geq n^{0.5-o(1)}$ as long as $k = \Omega(n^{2.5})$.

2 The Advice Model

While the multiphase problem is a dynamic version of set-disjointness, we can also formulate set-disjointness in a special version of the three player number on forehead model, which we will call *advice model*.

Definition [Pătraşcu] The advice model:
Given a function $f : \mathcal{X} \times \mathcal{Y} \to \{0,1\}$ we define $\mathrm{SEL}_f^{k\times 1} : [k] \times \mathcal{X}^k \times \mathcal{Y} \to \{0,1\}$ as

$$\mathrm{SEL}_f^{k\times 1}(i, x_1, \ldots, x_k, y) = f(x_i, y).$$

In the advice model, we have three players *Alice*, *Bob* and *Carmen*, who get the following information:

- Alice: $x_1, \ldots, x_k, y$.
- Bob: i, y
- Carmen: $i, x_1, \ldots, x_k$.

A protocol in the advice model then describes how Alice first sends some advice **privately** to Bob. After that Alice remains silent and Bob (knowing Alice's advice) and Carmen engage in a two-player protocol until they have computed the answer. The last bit send between Bob and Carmen is assumed to be the result $\mathrm{SEL}_f^{k\times 1}(i, x_1, \ldots, x_k, y) = f(x_i, y)$.

We will denote for a protocol π by $\mathcal{CC}_{A,B}(\pi)$ and $\mathcal{CC}_{B,C}(\pi)$ the total amount of advice given by Alice to Bob and the number of bits exchanged in the two player protocol between Bob and Carmen, respectively. Any non-trivial protocol π will adhere $\mathcal{CC}_{A,B} < k$, as otherwise Alice could just send a bit-string of length k with the answers for every possible i. We therefore will usually assume $\mathcal{CC}_{A,B} = o(k)$

The advice model is visualized in Fig. 3.

2.1 Connection to the multiphase problem

We will now look at the set-disjointness function DISJ in the advice model. The function $\mathrm{DISJ}(x, y)$ takes 0 if $x \in \{0,1\}^n$ and $y \in \{0,1\}^n$ (when viewed as subsets of $[n]$) are disjoint and 1 otherwise. Formally

$$\mathrm{DISJ}(x, y) = \begin{cases} 1, & \text{if } \exists j : x_j = y_j = 1 \\ 0, & \text{otherwise} \end{cases}$$

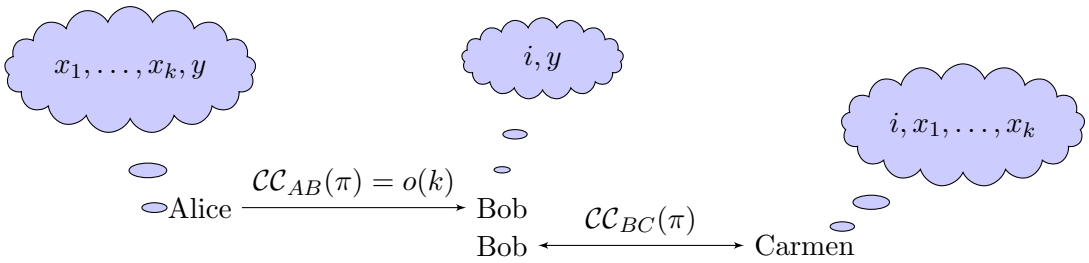

Figure 3: Visualization of a protocol π in the advice model. Alice first sends her advice privately to Bob, who then uses the advice and initiates a two-player communication protocol with Carmen to determine the answer $f(x_i, y)$.

Comparing $\mathrm{SEL}^{k\times 1}_{\mathrm{DISJ}}$ to the multiphase problem (that is comparing set-disjointness in the advice model and as a dynamic problem), one notices a lot of similarities. In the multiphase problem we are given sets $x_1, \ldots, x_k$ and y in the first two phases, respectively. This is exactly the input, which Alice gets in $\mathrm{SEL}^{k\times 1}_{\mathrm{DISJ}}$. Then in Phase III the query i is revealed, corresponding to Bob and Carmen knowing i.

We make this notion precise by conjecturing a lower bound on $\mathrm{SEL}^{k\times 1}_{\mathrm{DISJ}}$ and showing that it implies the hardness conjecture of the multiphase problem.

Conjecture 7 (Pătraşcu) *Let π be a protocol in the advice-model solving $\mathrm{SEL}^{k\times 1}_{\mathrm{DISJ}}$ and having complexity*

- $\mathcal{CC}_{AB}(\pi) = n \cdot M$
- $\mathcal{CC}_{BC}(\pi) = M$

for some M. Then there are constants $\gamma > 1, \delta > 0$ such that for $k = \Theta(n^\gamma)$ it holds that $M = \Omega(n^\delta)$.

Theorem 8 (Pătraşcu) *The $\mathrm{SEL}^{k\times 1}_{\mathrm{DISJ}}$ hardness conjecture (Conj. 7) implies the multiphase problem hardness conjecture (Conj. 1).*

Proof The proof works in the Word RAM, as well as the stronger cell-probe model. Let in the following $\omega = O(\log(n))$ be the size of a cell. We describe a protocol π in the advice model solving DISJ and using a solution to the multiphase problem for some τ.

Alice gets $x_1, \ldots, x_k, y$ and can therefore simulate phases I and II of the solution for the multiphase problem. Alice sends a list of all the cells she changed during phase II to Bob, together with their new value. Since phase II can only take time $\mathcal{O}(n \cdot \tau)$, the message Alice sent as advice is $\mathcal{O}(\omega \cdot n \cdot \tau)$.

Bob after receiving the advice from Alice and knowing i, starts executing the calculations of phase III. Whenever the calculation accesses a cell, Bob checks if he already knows the value of the cell from the message he got from Alice (that happens whenever the cell was accessed during phase II). Otherwise, Bob asks Carmen for help by sending her the address of the cell he wants to know (ω bits). Carmen knows $x_1, \ldots, x_k$ and can therefore simulate the whole first phase and thus answer all of Bob's questions about cells after the first phase.

The communication between Alice and Bob was $\mathcal{CC}_{AB}(\pi) = \mathcal{O}(\omega \cdot n \cdot \tau) = \mathcal{O}(n^{1+\varepsilon} \cdot \tau)$ (for any ε). Phase III takes $\mathcal{O}(\tau)$ time and even if Bob has to ask Carmen for every single cell, the communication between Bob and Carmen will not exceed $\mathcal{O}(\tau \cdot \omega) = \mathcal{O}(n^{\varepsilon} \cdot \tau)$. If we then have a lower bound for $M = n^{\varepsilon} \cdot \tau = \Omega(n^{\delta})$ for $\text{SEL}^{k\times 1}_{\text{DISJ}}$ then we directly get a lower bound of $\tau = \Omega(n^{\delta-\varepsilon}) = \Omega(n^{\delta'})$ for the multiphase problem.

At this point we can summarize the relationship between the problems discussed so far. The newly defined multiphase problem allows easy reductions to many dynamic problems, such as the dynamic shortest path or maximum matrix element problem. There is an open conjecture for the hardness of the multiphase problem and while we can neither prove nor refute it, there is some evidence that it holds since it is implied by two other hardness conjectures: the hardness of 3-SUM and the lower bound conjectures for set-disjointness in the advice model. We refer to Fig. 4 for an overview.

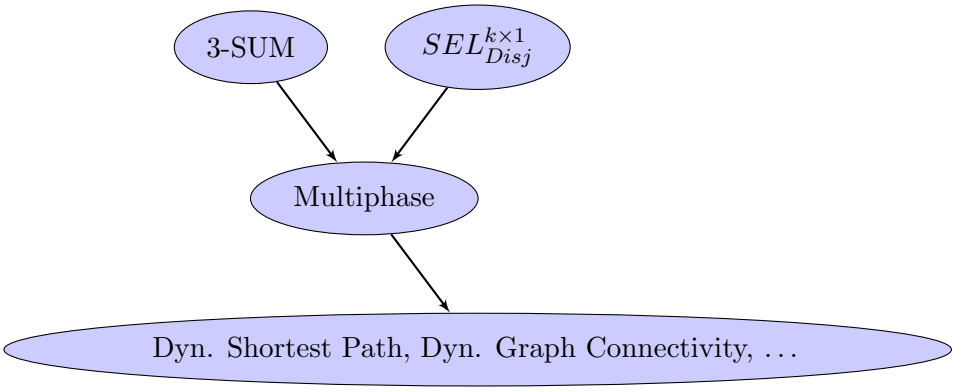

Figure 4: Relationship between 3-SUM, dynamic problems, the multiphase problem and set-disjointness in the advice model. An edge from A to B indicates that the hardness of A, as conjectured, implies the hardness of B.

2.2 Connections between the advice model and two-player communication complexity

Generalizing the argument from the hardness conjecture of $\text{SEL}^{k\times 1}_{\text{DISJ}}$ to other functions, we might develop the intuition that, whenever Alice only gives advice $o(k)$, then for a large portion of the input-cases $x_1, \ldots, x_k$ she does not give any advice at all. Bob and Carmen then basically end up with no advice and have to execute a normal two-player protocol from scratch.

More formally, we might think that for every function f, for every protocol π solving $\text{SEL}^{k\times 1}_{f}$ in the advice model, it holds that

$$\mathcal{CC}_{A,B}(\pi) = o(k) \Rightarrow \mathcal{CC}_{B,C}(\pi) = \Omega\left(\mathcal{CC}(f)\right)$$

where $\mathcal{CC}(f)$ is the two-player communication complexity of f.

Surprisingly, this intuition does not hold. Neither for deterministic nor for randomized protocols.

2.2.1 EQ in the deterministic case

A counter example to the above intuition in the deterministic case is equality. That is, the problem EQ, defined as

$$\mathrm{EQ}(x,y) = \begin{cases} 1, & \text{if } x = y \\ 0, & \text{otherwise} \end{cases}$$

we know that the two-player communication complexity is $\Omega(n)$ in the deterministic setting. In the following we describe an easy deterministic advice model protocol for $\mathrm{SEL}_{\mathrm{EQ}}^{k\times 1}$ with only $\mathcal{O}(\log(k))$ communication cost.

Alice gets $x_1, \ldots, x_k, y$ and determines the smallest integer j, such that $x_j = y$. If there is no such j, set $j = 0$. Alice then sends j privately to Bob.

Bob passes the advice he gets from Alice on to Carmen.

Carmen receives the advice j. If it is $j = 0$, then she can answer 0 directly as none of the x_i are equal to y. Otherwise, she looks up the corresponding x_j and now knows $y = x_j$. Carmen can then test if $x_i = x_j$ holds. If yes she answers 1 and 0 otherwise.

The protocol is deterministic and the number of bits sent is $\mathcal{CC}_{A,B} = \mathcal{CC}_{B,C} = \log(k)$ and therefore $\mathcal{CC}_{B,C} \ll \mathcal{CC}$ for $k = O(\mathrm{poly}(n))$.

2.2.2 Derandomization

A more general attack on our intuition is provided by the following derandomization theorem.

Theorem 9 (Chattopadhyay, Edmonds, Ellen, Pitassi) *If there is a randomized protocol π^R with error ε in the the advice model, then we can construct a deterministic advice model protocol π^D calculating the same function with*

$$\mathcal{CC}_{A,B}(\pi^D) = \mathcal{O}\left(\frac{\log(k)}{\varepsilon^2} \cdot \left(\mathcal{CC}_{A,B}(\pi^R) + \log(kn)\right)\right)$$
$$\mathcal{CC}_{B,C}(\pi^D) = \mathcal{O}\left(\frac{\log(k)}{\varepsilon^2} \cdot \left(\mathcal{CC}_{B,C}(\pi^R) + \log(kn)\right)\right)$$

Proof [Outline] The proof works in two steps. It first shows that the number of random-bits used by π^R cannot be too large ($\mathcal{O}(\log(nk))$) and then shows that there is a particular set of random strings, such that the majority answer over them is correct for all inputs.

The theorem says that for any constant ε and values of k not too big ($k = \mathcal{O}(\mathrm{poly}(n)$, for example) we can derandomize at the cost of a factor of $\log(n)$. This directly implies, that all problems which are easy in the two-player randomized setting will also be easy in the deterministic advice model.

2.2.3 Hard randomized functions

The question remains whether there are also functions which are hard in the two-player randomized setting but easy in the advice model. It turns out that there are such functions, as stated by the following theorem.

Theorem 10 (Chattopadhyay, Edmonds, Ellen, Pitassi) *There exists a boolean function f with two-player randomized communication complexity $\Omega(n)$ such that $\mathrm{SEL}_f^{k\times 1}$ has a deterministic advice model protocol π with*

$$\mathcal{CC}_{A,B}(\pi) = \mathcal{CC}_{B,C}(\pi) = \mathcal{O}(log^2(n))$$

for $k = \mathcal{O}(\mathrm{poly}(n))$.

Proof [Outline] In the proof we first build a randomized protocol in the advice model and then derandomize it. The randomized protocol implements a learning algorithm. That is, Alice tries to teach y to Bob by sending him a set of samples $(j, f(x_j, y))$. Results from learning theory relate the number of samples needed to the *Vapnik-Chervonenkis* dimension of the function-matrix of f and it is known that there are functions f with high randomized two-player communication complexity but constant *Vapnik-Chervonenkis* dimension.

2.3 Upper bounds for set-disjointness in the advice model

The following advice model protocol π upper bounds the complexity of $\mathrm{SEL}_{\mathrm{DISJ}}^{k\times 1}$ by

$$\mathcal{CC}_{A,B}(\pi) = \mathcal{CC}_{A,B}(\pi) = \mathcal{O}\left(\sqrt{n}\log(k)\right)$$

Note that for $k = \mathcal{O}(\mathrm{poly}(n))$, the complexity is almost $\mathcal{O}(\sqrt{n})$. That is for $k = \mathcal{O}(\mathrm{poly}(n))$ we get $\mathcal{CC}_{A,B}(\pi) = \mathcal{CC}_{A,B}(\pi) = \mathcal{O}\left(n^{0.501}\right)$.

Alice gets $x_1, \ldots, x_k, y$ and executes the following algorithm

ALICE$(x_1, \ldots, x_k, y)$
1 $S = \emptyset$.
2 **for** $j = 1, \ldots, k$
  // **Loop-Invariant:** $S \cap y = \emptyset$.
3 **if** $x_j \cap y = \emptyset \wedge |([n] \setminus S) \cap x_j| \geq \sqrt{n}$
4 Send index j to Bob.
5 $S = S \cup x_j$

Bob forwards the advice he gets from Alice to Carmen.

Carmen gets the advice from Bob and knowing $x_1, \ldots, x_k$ she can reconstruct the set S for any step j. Let $M \subseteq [k]$ be all the indices sent as an advice by Alice. Carmen then proceeds with the following algorithm

```
CARMEN(x_1, ..., x_k, i, M)
 1  if i ∈ M
        // x_i ∩ y = ∅ must hold as i was added to M.
 2      return 0
    else
        // Determine the set S when i was processed.
 3      S = ⋃_{j∈M∩[i]} x_j.
 4      if |([n] \ S) ∩ x_i| < √n
            // It holds that S ∩ y = ∅, thus x_i ∩ y = (([n] \ S) ∩ x_i) ∩ y.
            // Brute force over the elements in ([n] \ S) ∩ x_i.
 5          for j ∈ ([n] \ S) ∩ x_i
 6              Ask Bob whether j ∈ y.
 7              if Yes
 8                  return 1.
 9          return 0.
10      else
            // It must hold that x_i ∩ y ≠ ∅, since |([n] \ S_i) ∩ x_i| ≥ √n yet i is not in M.
11          return 1.
```

Analyzing the amount of advice given by Alice, we note that each index j sent by Alice is of length $\log(k)$ and whenever it is sent, then the size of S increases by at least $\sqrt{n}$. The total number of times lines 4 and 5 are executed in ALICE is therefore bounded by $\frac{n}{\sqrt{n}} = \sqrt{n}$ and the total advice given by Alice amounts to $\mathcal{CC}_{A,B} = \mathcal{O}(\sqrt{n} \cdot \log(k))$.

The interaction between Bob and Carmen consists of Bob forwarding the advice from Alice and Carmen asking Bob for help in line 6 of CARMEN. Each such query involves Carmen sending the index j of size $\log(k)$ and Bob answering with one bit. The total amount of queries is bounded by $|([n] \setminus S) \cap x_i| \leq \sqrt{n}$ and we get $\mathcal{CC}_{B,C} = \mathcal{O}(\sqrt{n} \cdot \log(k))$.

3 Restricted Advice Models

Looking at the above protocol for $\text{SEL}^{k\times 1}_{\text{DISJ}}$ one sees that the protocol does not actually use the full power of the advice model. On one hand, the advice sent by Alice is less than the $o(k)$ she is entitled to. On the other hand, the interaction between Bob and Carmen is quite limited. That is, Carmen could just have combined all her $\sqrt{n}$ queries and sent them in one message of size $\mathcal{O}(\sqrt{n}\log(n))$ to Bob. Bob would not have had to actually answer the queries but could have computed the answer himself and terminated the protocol.

Those two attributes of the protocol give rise to study two different restricted versions of the advice model.

3.1 Lower bounds via Direct Product Theorems

We first explore the restricted setting when limiting the advice Alice can give.

Theorem 11 (Klauck) *Set-disjointness has a strong direct product theorem. That is there exist constants $0 < \beta < 1$ and $\alpha > 0$ such that every randomized protocol computing* DISJ^k *with*

$\mathrm{DISJ}^k(x_1, \ldots, x_k, y_1, \ldots, y_k) = (\mathrm{DISJ}(x_1, y_1), \ldots, \mathrm{DISJ}(x_k, y_k))$ *using at most* βkn *bits of communication has worst case success probability less than* $2^{-\alpha k}$.

The theorem carries over to the advice model:

Theorem 12 (Chattopadhyay, Edmonds, Ellen, Pitassi) *There exist constants* $0 < \beta < 1$ *and* $\alpha > 0$ *such that for every deterministic protocol* π *computing* $\mathrm{SEL}_{\mathrm{DISJ}}^{\sqrt{n}\times 1}$ *it holds that*

$$\mathcal{CC}_{A,B} \leq \alpha\sqrt{n} \Rightarrow \mathcal{CC}_{B,C} \geq \beta\sqrt{n}$$

Proof [Outline] The proof works by contradiction, assuming that there is such an efficient, deterministic protocol for $\mathrm{SEL}_{\mathrm{DISJ}}^{\sqrt{n}\times 1}$. From that, one can build a deterministic protocol for DISJ^k for every distribution μ over the inputs. Applying Yao's min-max principle then gives a contradiction to the direct product theorem for set-disjointness.

It is worth noting that the upper bound for $\mathrm{SEL}_{\mathrm{DISJ}}^{k\times 1}$ described in section 2.3 with $k = \Theta(\sqrt{n})$ matches this lower bound up to a factor of $\log(n)$.

3.2 Limited Rounds

For the second variant of restricting the advice model, we define a 1.5 round protocols.

Definition [Chattopadhyay, Edmonds, Ellen, Pitassi] A 1.5 round (m, l, q)-protocol π is a protocol where

Alice sends a message A of size $\mathcal{CC}_{A,B}(\pi) = m = o(k)$ privately to Bob.

Bob, without looking at i, sends a message $B = B(A, y)$ of length l to Carmen.

Carmen sends a message C of length q back to Bob.

Bob knowing i, y, A, C calculates the result.

Since Bob does not look at i when sending his message to Carmen, we say that he only participates in half a round. Note that Alice could calculate the message B as well and we can think of the message B as just being a substring of A of length l.

We note that the protocol realizing the upper bound for $\mathrm{SEL}_{\mathrm{DISJ}}^{k\times 1}$ from section 2.3 is a 1.5 round $(\mathcal{O}(n^{0.501}, n^{0.501}, n^{0.501})$-protocol (for $k = \mathcal{O}(\mathrm{poly}(n))$).

The following two theorems describe lower bounds for the set-disjointness and inner product functions in the 1.5 round advice mode. Both proves work by giving a compression argument, lower bounding the communication complexity between Bob and Carmen.

Theorem 13 (Chattopadhyay, Edmonds, Ellen, Pitassi) *For every 1.5 round* (m, l, q)*-protocol computing* $\mathrm{SEL}_{\mathrm{DISJ}}^{k\times 1}$*, we have* $l \cdot q \geq \frac{n}{35}$ *provided* $k \geq 300n(n + m)$.

Theorem 14 (Chattopadhyay, Edmonds, Ellen, Pitassi) *Let* $\alpha > 2$ *be some constant. For every 1.5 round* (m, l, q)*-protocol computing* $\mathrm{SEL}_{\mathrm{IP}}^{k\times 1}$*, we have* $(l + q) \geq \frac{n(\alpha-2)}{\alpha}$ *provided* $k \geq \alpha(n + m)$.