Improved Direct Product Theorems for Randomized Query Complexity

Andrew Drucker

Sept. 13, 2010

Big picture

- Usually, computer users have not one goal, but many.
- When can multiple computations be combined to make them easier?

Big picture

- Usually, computer users have not one goal, but many.
- When can multiple computations be combined to make them easier?

Separate inputs

Suppose each of the outputs we want to compute depends on a separate input.

For example:

$$\boldsymbol{X^1} \longrightarrow \boldsymbol{F(X^1)}$$

$$\boldsymbol{X^2} \longrightarrow \boldsymbol{F(X^2)}$$

$$\boldsymbol{X^3} \longrightarrow \boldsymbol{F(X^3)}$$

Direct Product Theorems

- Intuition: the different outputs are 'unrelated', so computing them together shouldn't make the task easier.
- **Direct Product Theorems (DPTs)** are results that make this intuition rigorous (when it's correct!).
- DPTs have been studied for many years, in many computational models.
- Our focus: **randomized query algorithms**, with

$$\text{cost} = \text{number of queries to the input.}$$

Direct Product Theorems

- Intuition: the different outputs are 'unrelated', so computing them together shouldn't make the task easier.
- **Direct Product Theorems (DPTs)** are results that make this intuition rigorous (when it's correct!).
- DPTs have been studied for many years, in many computational models.
- Our focus: **randomized query algorithms**, with

$$\text{cost} = \text{number of queries to the input.}$$

Direct Product Theorems

- Intuition: the different outputs are 'unrelated', so computing them together shouldn't make the task easier.
- **Direct Product Theorems (DPTs)** are results that make this intuition rigorous (when it's correct!).
- DPTs have been studied for many years, in many computational models.
- Our focus: **randomized query algorithms**, with

$$\text{cost} = \text{number of queries to the input.}$$

Direct Product Theorems

- Intuition: the different outputs are 'unrelated', so computing them together shouldn't make the task easier.
- **Direct Product Theorems (DPTs)** are results that make this intuition rigorous (when it's correct!).
- DPTs have been studied for many years, in many computational models.
- Our focus: **randomized query algorithms**, with

$$\text{cost} = \text{number of queries to the input.}$$

Direct Product Theorems

- Intuition: the different outputs are 'unrelated', so computing them together shouldn't make the task easier.
- **Direct Product Theorems (DPTs)** are results that make this intuition rigorous (when it's correct!).
- DPTs have been studied for many years, in many computational models.
- Our focus: **randomized query algorithms**, with

$$\text{cost} = \text{number of queries to the input.}$$

Direct Product Theorems

- Intuition: the different outputs are 'unrelated', so computing them together shouldn't make the task easier.
- **Direct Product Theorems (DPTs)** are results that make this intuition rigorous (when it's correct!).
- DPTs have been studied for many years, in many computational models.
- Our focus: **randomized query algorithms**, with

$$\text{cost} = \text{number of queries to the input.}$$

Direct products

- Given
$$F : \{0,1\}^n \to \Sigma, \quad \text{and } k > 1,$$
define
$$F^{\otimes k}(x^1 \ldots, x^k) \triangleq \left(F(x^1), \ldots, F(x^k)\right),$$
a function of k different n-bit inputs $x^1, \ldots, x^k$.
- $F^{\otimes k}$ = 'k**-fold direct product**' of F.

Direct products

- Given
$$F : \{0,1\}^n \to \Sigma, \quad \text{and } k > 1,$$
define
$$F^{\otimes k}(x^1 \ldots, x^k) \triangleq \left(F(x^1), \ldots, F(x^k)\right),$$
a function of k different n-bit inputs $x^1, \ldots, x^k$.
- $F^{\otimes k}$ = '**k-fold direct product**' of F.

Average-case complexity

- For a function F, a query bound $T > 0$, and a distribution μ over inputs to F, define

$$\mathsf{Suc}_{T,\mu}(F)$$

as the **maximum success probability** of any T-query algorithm $\mathcal{R}$ in computing $F(\mathbf{y})$ on input $\mathbf{y} \sim \mu$.

- (probability over randomness in $\mathbf{y}$ and in $\mathcal{R}$)

Average-case complexity

- For a function F, a query bound $T > 0$, and a distribution μ over inputs to F, define

$$\mathrm{Suc}_{T,\mu}(F)$$

  as the **maximum success probability** of any T-query algorithm $\mathcal{R}$ in computing $F(\mathbf{y})$ on input $\mathbf{y} \sim \mu$.
- (probability over randomness in $\mathbf{y}$ and in $\mathcal{R}$)

The form of a DPT

- Let $\mu^{\otimes k}$ denote k independent samples from μ.
- A **Direct Product Theorem** is of the form:

$$\forall F, \quad \mathrm{Suc}_{T,\mu}(F) \leq p \quad \Longrightarrow \quad \mathrm{Suc}_{T',\mu^{\otimes k}}(F^{\otimes k}) \leq p',$$

  where T', p' depend on T, p, and k.
- We hope to have $p' \ll p$ and $T' \gg T$.
- *"F is hard $\Rightarrow F^{\otimes k}$ is harder."*

The form of a DPT

- Let $\mu^{\otimes k}$ denote k independent samples from μ.
- A **Direct Product Theorem** is of the form:

$$\forall F, \quad \mathrm{Suc}_{T,\mu}(F) \leq p \quad \Longrightarrow \quad \mathrm{Suc}_{T',\mu^{\otimes k}}(F^{\otimes k}) \leq p',$$

where T', p' depend on T, p, and k.

- We hope to have $p' \ll p$ and $T' \gg T$.
- *"F is hard $\Rightarrow F^{\otimes k}$ is harder."*

The form of a DPT

- Let $\mu^{\otimes k}$ denote k independent samples from μ.
- A **Direct Product Theorem** is of the form:

$$\forall F, \quad \mathrm{Suc}_{T,\mu}(F) \leq p \quad \Longrightarrow \quad \mathrm{Suc}_{T',\mu^{\otimes k}}(F^{\otimes k}) \leq p',$$

  where T', p' depend on T, p, and k.
- We hope to have $p' \ll p$ and $T' \gg T$.
- *"F is hard $\Rightarrow$ $F^{\otimes k}$ is harder."*

An 'ideal' DPT?

- The strongest DPT we could hope for would say:

$$\forall F, \quad \mathsf{Suc}_{T,\mu}(F) \le 1-\varepsilon \quad \Longrightarrow \quad \mathsf{Suc}_{Tk,\mu^{\otimes k}}(F^{\otimes k}) \le (1-\varepsilon)^k.$$

- $(1-\varepsilon)^k$ is the success prob. we'd get if we run the optimal T-query algorithm on each of the k inputs.
- True for restricted classes of algorithms **[NRS94]**, **[Sha03]**.
- Shaltiel **[Sha03]** defined **fair** Tk-query algorithms for $F^{\otimes k}$ as ones which make exactly T queries to each of the k inputs. He proved an 'ideal' DPT for these algorithms.

An 'ideal' DPT?

- The strongest DPT we could hope for would say:

$$\forall F, \quad \mathrm{Suc}_{T,\mu}(F) \leq 1-\varepsilon \quad \Longrightarrow \quad \mathrm{Suc}_{Tk,\mu^{\otimes k}}(F^{\otimes k}) \leq (1-\varepsilon)^k.$$

- $(1-\varepsilon)^k$ is the success prob. we'd get if we run the optimal T-query algorithm on each of the k inputs.
- True for restricted classes of algorithms **[NRS94]**, **[Sha03]**.
- Shaltiel **[Sha03]** defined **fair** Tk-query algorithms for $F^{\otimes k}$ as ones which make exactly T queries to each of the k inputs. He proved an 'ideal' DPT for these algorithms.

An 'ideal' DPT?

- The strongest DPT we could hope for would say:

$$\forall F, \quad \mathrm{Suc}_{T,\mu}(F) \leq 1-\varepsilon \quad \Longrightarrow \quad \mathrm{Suc}_{Tk,\mu^{\otimes k}}(F^{\otimes k}) \leq (1-\varepsilon)^k.$$

- $(1-\varepsilon)^k$ is the success prob. we'd get if we run the optimal T-query algorithm on each of the k inputs.
- True for restricted classes of algorithms **[NRS94]**, **[Sha03]**.
- Shaltiel **[Sha03]** defined **fair** Tk-query algorithms for $F^{\otimes k}$ as ones which make exactly T queries to each of the k inputs. He proved an 'ideal' DPT for these algorithms.

An 'ideal' DPT?

- The strongest DPT we could hope for would say:

$$\forall F, \quad \mathrm{Suc}_{T,\mu}(F) \leq 1-\varepsilon \quad \Longrightarrow \quad \mathrm{Suc}_{Tk,\mu^{\otimes k}}(F^{\otimes k}) \leq (1-\varepsilon)^k.$$

- $(1-\varepsilon)^k$ is the success prob. we'd get if we run the optimal T-query algorithm on each of the k inputs.
- True for restricted classes of algorithms **[NRS94]**, **[Sha03]**.
- Shaltiel **[Sha03]** defined **fair** Tk-query algorithms for $F^{\otimes k}$ as ones which make exactly T queries to each of the k inputs. He proved an 'ideal' DPT for these algorithms.

An ‘ideal’ DPT?

- But, Shaltiel also showed the ideal DPT is **false** in general!
- The message: we can sometimes solve $F^{\otimes k}$ more effectively by adaptive reallocation of queries.
- Counterexamples of **[Sha03]** apply to most computational models.

An 'ideal' DPT?

- But, Shaltiel also showed the ideal DPT is **false** in general!
- The message: we can sometimes solve $F^{\otimes k}$ more effectively by adaptive reallocation of queries.
- Counterexamples of **[Sha03]** apply to most computational models.

An ‘ideal’ DPT?

- But, Shaltiel also showed the ideal DPT is **false** in general!
- The message: we can sometimes solve $F^{\otimes k}$ more effectively by adaptive reallocation of queries.
- Counterexamples of **[Sha03]** apply to most computational models.

Our new DPT

We modify Shaltiel's techniques for fair algorithms, to show a new DPT for unrestricted query algorithms.

Our new DPT

Theorem

For any Boolean function F and $\alpha > 0$,

$$\mathsf{Suc}_{T,\mu}(F) \le 1 - \varepsilon \quad \Rightarrow \quad \mathsf{Suc}_{\alpha\varepsilon Tk,\mu^{\otimes k}}(F^{\otimes k}) \le (2^{\alpha\varepsilon}(1-\varepsilon))^k.$$

- Success probability drops exponentially in k, if (number of queries) $\approx \varepsilon Tk$.
  For $\alpha \le 1$ we have $2^{\alpha\varepsilon}(1-\varepsilon) \le 1 - \varepsilon + \alpha\varepsilon$.
- Varying α gives a tradeoff between the query bound and the success probability.
- Shaltiel's examples tell us this is a nearly optimal tradeoff (for most parameter settings).

Our new DPT

Theorem

For any Boolean function F and $\alpha > 0$,

$$\mathrm{Suc}_{T,\mu}(F) \leq 1 - \varepsilon \quad \Rightarrow \quad \mathrm{Suc}_{\alpha\varepsilon Tk,\mu^{\otimes k}}(F^{\otimes k}) \leq (2^{\alpha\varepsilon}(1-\varepsilon))^k.$$

- Success probability drops exponentially in k, if (number of queries) $\approx \varepsilon Tk$.
  For $\alpha \leq 1$ we have $2^{\alpha\varepsilon}(1-\varepsilon) \leq 1 - \varepsilon + \alpha\varepsilon$.
- Varying α gives a tradeoff between the query bound and the success probability.
- Shaltiel's examples tell us this is a nearly optimal tradeoff (for most parameter settings).

Our new DPT

Theorem

For any Boolean function F and $\alpha > 0$,

$$\mathrm{Suc}_{T,\mu}(F) \leq 1 - \varepsilon \quad \Rightarrow \quad \mathrm{Suc}_{\alpha\varepsilon Tk,\mu^{\otimes k}}(F^{\otimes k}) \leq (2^{\alpha\varepsilon}(1-\varepsilon))^k.$$

- Success probability drops exponentially in k, if (number of queries) $\approx \varepsilon Tk$.
  For $\alpha \leq 1$ we have $2^{\alpha\varepsilon}(1-\varepsilon) \leq 1 - \varepsilon + \alpha\varepsilon$.
- Varying α gives a tradeoff between the query bound and the success probability.
- Shaltiel's examples tell us this is a nearly optimal tradeoff (for most parameter settings).

Our new DPT

Theorem

For any Boolean function F and $\alpha > 0$,

$$\mathrm{Suc}_{T,\mu}(F) \leq 1-\varepsilon \quad \Rightarrow \quad \mathrm{Suc}_{\alpha\varepsilon Tk,\mu^{\otimes k}}(F^{\otimes k}) \leq (2^{\alpha\varepsilon}(1-\varepsilon))^k.$$

- Success probability drops exponentially in k, if (number of queries) $\approx \varepsilon Tk$.
  For $\alpha \leq 1$ we have $2^{\alpha\varepsilon}(1-\varepsilon) \leq 1-\varepsilon+\alpha\varepsilon$.
- Varying α gives a tradeoff between the query bound and the success probability.
- Shaltiel's examples tell us this is a nearly optimal tradeoff (for most parameter settings).

Proof sketch

- First, some definitions about a single, n-bit input $\mathbf{y} \sim \mu$ to F.
- For $v \in \{0, 1, *\}^n$, let $\mu[v]$ denote $\mathbf{y} \sim \mu$ conditioned on the event

$$[\mathbf{y}_i = v_i, \text{ for each } i \text{ such that } v_i \in \{0, 1\}].$$

- E.g., if μ is uniform on 3 bits, then $\mu[00*]$ is uniform on $\{000, 001\}$.
- (We can assume μ has full support.)
- Let $|v|$ = number of 0/1 entries in v.

Proof sketch

- First, some definitions about a single, n-bit input $\mathbf{y} \sim \mu$ to F.
- For $v \in \{0, 1, *\}^n$, let $\mu[v]$ denote $\mathbf{y} \sim \mu$ conditioned on the event
$$[\mathbf{y}_i = v_i, \text{ for each } i \text{ such that } v_i \in \{0, 1\}].$$
- E.g., if μ is uniform on 3 bits, then $\mu[00*]$ is uniform on $\{000, 001\}$.
- (We can assume μ has full support.)
- Let $|v|$ = number of 0/1 entries in v.

Proof sketch

- First, some definitions about a single, n-bit input $\mathbf{y} \sim \mu$ to F.
- For $v \in \{0, 1, *\}^n$, let $\mu[v]$ denote $\mathbf{y} \sim \mu$ conditioned on the event
$$[\mathbf{y}_i = v_i, \text{ for each } i \text{ such that } v_i \in \{0, 1\}].$$
- E.g., if μ is uniform on 3 bits, then $\mu[00*]$ is uniform on $\{000, 001\}$.
- (We can assume μ has full support.)
- Let $|v|$ = number of 0/1 entries in v.

Proof sketch

- First, some definitions about a single, n-bit input $\mathbf{y} \sim \mu$ to F.
- For $v \in \{0, 1, *\}^n$, let $\mu[v]$ denote $\mathbf{y} \sim \mu$ conditioned on the event
$$[\mathbf{y}_i = v_i, \text{ for each } i \text{ such that } v_i \in \{0, 1\}].$$
- E.g., if μ is uniform on 3 bits, then $\mu[00*]$ is uniform on $\{000, 001\}$.
- (We can assume μ has full support.)
- Let $|v|$ = number of 0/1 entries in v.

Proof sketch

- First, some definitions about a single, n-bit input $\mathbf{y} \sim \mu$ to F.
- For $v \in \{0, 1, *\}^n$, let $\mu[v]$ denote $\mathbf{y} \sim \mu$ conditioned on the event
$$[\mathbf{y}_i = v_i, \text{ for each } i \text{ such that } v_i \in \{0, 1\}].$$
- E.g., if μ is uniform on 3 bits, then $\mu[00*]$ is uniform on $\{000, 001\}$.
- (We can assume μ has full support.)
- Let $|v|$ = number of $0/1$ entries in v.

The k-fold setting

- Say the algorithm $\mathcal{R}$ receives inputs $\mathbf{x}^1, \ldots, \mathbf{x}^k \sim \mu^{\otimes k}$ and makes $M = \lfloor \alpha \varepsilon T k \rfloor$ queries.
- For $j \in \{1, \ldots, k\}$ and $t \geq 0$, let the random string

$$v_t^j \in \{0, 1, *\}^n$$

describe bits seen of the j-th input $\mathbf{x}^j$, after $\mathcal{R}$ has made t queries overall (to the entire collection).

Claim

Conditioned on $v_t^1, \ldots, v_t^k$, the k inputs remain independent, with

$$\mathbf{x}^j \sim \mu[v_t^j].$$

Proof is a simple calculation.

The k-fold setting

- Say the algorithm $\mathcal{R}$ receives inputs $\mathbf{x}^1, \ldots, \mathbf{x}^k \sim \mu^{\otimes k}$ and makes $M = \lfloor \alpha \varepsilon T k \rfloor$ queries.
- For $j \in \{1, \ldots, k\}$ and $t \geq 0$, let the random string

$$v_t^j \in \{0, 1, *\}^n$$

describe bits seen of the j-th input $\mathbf{x}^j$, after $\mathcal{R}$ has made t queries overall (to the entire collection).

Claim

Conditioned on $v_t^1, \ldots, v_t^k$, the k inputs remain independent, with

$$\mathbf{x}^j \sim \mu[v_t^j].$$

Proof is a simple calculation.

The k-fold setting

- Say the algorithm $\mathcal{R}$ receives inputs $\mathbf{x}^1, \ldots, \mathbf{x}^k \sim \mu^{\otimes k}$ and makes $M = \lfloor \alpha \varepsilon T k \rfloor$ queries.
- For $j \in \{1, \ldots, k\}$ and $t \geq 0$, let the random string

$$v_t^j \in \{0, 1, *\}^n$$

describe bits seen of the j-th input $\mathbf{x}^j$, after $\mathcal{R}$ has made t queries overall (to the entire collection).

Claim

Conditioned on $v_t^1, \ldots, v_t^k$, the k inputs remain independent, with

$$\mathbf{x}^j \sim \mu[v_t^j].$$

Proof is a simple calculation.

The k-fold setting

- Say the algorithm $\mathcal{R}$ receives inputs $\mathbf{x}^1, \dots, \mathbf{x}^k \sim \mu^{\otimes k}$ and makes $M = \lfloor \alpha \varepsilon T k \rfloor$ queries.
- For $j \in \{1, \dots, k\}$ and $t \geq 0$, let the random string

$$v_t^j \in \{0, 1, *\}^n$$

describe bits seen of the j-th input $\mathbf{x}^j$, after $\mathcal{R}$ has made t queries overall (to the entire collection).

Claim

Conditioned on $v_t^1, \dots, v_t^k$, the k inputs remain independent, with

$$\mathbf{x}^j \sim \mu[v_t^j].$$

Proof is a simple calculation.

k inputs, k 'fortunes'

- For each input $\mathbf{x}^j$ and each step $t \geq 0$, define a random variable $X(j,t) \in [0,1]$.
- Think of the algorithm $\mathcal{R}$ as a gambler gambling at k tables, and consider $X(j,t)$ his fortune at the j-th table after t steps (i.e., queries).

k inputs, k 'fortunes'

- Recall: $v_t^j \in \{0, 1, *\}^n$ describes the queries made to $\mathbf{x}^j$ so far.
- If $|v_t^j| \leq T$, say that input j is **under-budget** (after t steps), otherwise j is **over-budget**.
- If j is under-budget, define $X(j, t)$ as the **maximum success probability** of computing $F(\mathbf{x}^j)$ correctly of any algorithm making $\leq T - |v_t^j|$ queries to input $\mathbf{x}^j$, under distribution

$$\mathbf{x}^j \sim \mu[v_t^j].$$

- Meaning: $X(j, t) =$ best possible 'winning prospects' of computing $F(\mathbf{x}^j)$, if we stay under-budget.
- Observe: $X(j, t) \geq 1/2$ in this case.

k inputs, k 'fortunes'

- Recall: $v_t^j \in \{0, 1, *\}^n$ describes the queries made to $\mathbf{x}^j$ so far.
- If $|v_t^j| \leq T$, say that input j is **under-budget** (after t steps), otherwise j is **over-budget**.
- If j is under-budget, define $X(j, t)$ as the **maximum success probability** of computing $F(\mathbf{x}^j)$ correctly of any algorithm making $\leq T - |v_t^j|$ queries to input $\mathbf{x}^j$, under distribution

$$\mathbf{x}^j \sim \mu[v_t^j].$$

- Meaning: $X(j, t) =$ best possible 'winning prospects' of computing $F(\mathbf{x}^j)$, if we stay under-budget.
- Observe: $X(j, t) \geq 1/2$ in this case.

k inputs, k 'fortunes'

- Recall: $v_t^j \in \{0, 1, *\}^n$ describes the queries made to $\mathbf{x}^j$ so far.
- If $|v_t^j| \leq T$, say that input j is **under-budget** (after t steps), otherwise j is **over-budget**.
- If j is under-budget, define $X(j, t)$ as the **maximum success probability** of computing $F(\mathbf{x}^j)$ correctly of any algorithm making $\leq T - |v_t^j|$ queries to input $\mathbf{x}^j$, under distribution

$$\mathbf{x}^j \sim \mu[v_t^j].$$

- Meaning: $X(j, t) =$ best possible 'winning prospects' of computing $F(\mathbf{x}^j)$, if we stay under-budget.
- Observe: $X(j, t) \geq 1/2$ in this case.

k inputs, k 'fortunes'

- Recall: $v_t^j \in \{0, 1, *\}^n$ describes the queries made to $\mathbf{x}^j$ so far.
- If $|v_t^j| \le T$, say that input j is **under-budget** (after t steps), otherwise j is **over-budget**.
- If j is under-budget, define $X(j, t)$ as the **maximum success probability** of computing $F(\mathbf{x}^j)$ correctly of any algorithm making $\le T - |v_t^j|$ queries to input $\mathbf{x}^j$, under distribution

$$\mathbf{x}^j \sim \mu[v_t^j].$$

- Meaning: $X(j, t) =$ best possible 'winning prospects' of computing $F(\mathbf{x}^j)$, if we stay under-budget.
- Observe: $X(j, t) \ge 1/2$ in this case.

k inputs, k 'fortunes'

- Recall: $v_t^j \in \{0, 1, *\}^n$ describes the queries made to $\mathbf{x}^j$ so far.
- If $|v_t^j| \le T$, say that input j is **under-budget** (after t steps), otherwise j is **over-budget**.
- If j is under-budget, define $X(j, t)$ as the **maximum success probability** of computing $F(\mathbf{x}^j)$ correctly of any algorithm making $\le T - |v_t^j|$ queries to input $\mathbf{x}^j$, under distribution

$$\mathbf{x}^j \sim \mu[v_t^j].$$

- Meaning: $X(j, t) =$ best possible 'winning prospects' of computing $F(\mathbf{x}^j)$, if we stay under-budget.
- Observe: $X(j, t) \ge 1/2$ in this case.

k inputs, k ‘fortunes’

- If j is over-budget, set

$$X(j, t) = 1/2.$$

- Note: going over-budget can’t increase our fortune!

k inputs, k 'fortunes'

- If j is over-budget, set

$$X(j, t) = 1/2.$$

- Note: going over-budget can't increase our fortune!

Unfavorable gambles

Two important properties:

1. For all j,
$$X(j, 0) \leq 1 - \varepsilon$$
(follows from our initial assumption that $\mathrm{Suc}_{\mathcal{T},\mu}(F) \leq 1 - \varepsilon$).
2. If $\mathcal{R}$ makes its next query at table j, then
$$\mathbb{E}[X(j, t+1) | v_t^1, \ldots, v_t^k] \leq X(j, t), \text{ and}$$
$$X(j', t+1) = X(j', t) \quad \forall j' \neq j.$$
(Follows from definition of $X(j, t)$ and the fact that the inputs remain independent.)

So, choosing input j to query next is like making an unfavorable gamble at the j-th table!

Unfavorable gambles

Two important properties:

1. For all j,

$$X(j,0) \leq 1-\varepsilon$$

   (follows from our initial assumption that $\mathrm{Suc}_{T,\mu}(F) \leq 1-\varepsilon$).

2. If $\mathcal{R}$ makes its next query at table j, then

$$\mathbb{E}[X(j,t+1)|v_t^1,\ldots,v_t^k] \leq X(j,t), \text{ and}$$

$$X(j',t+1) = X(j',t) \quad \forall j' \neq j.$$

   (Follows from definition of $X(j,t)$ and the fact that the inputs remain independent.)

So, choosing input j to query next is like making an unfavorable gamble at the j-th table!

Unfavorable gambles

Two important properties:

1. For all j,
$$X(j,0) \leq 1-\varepsilon$$
(follows from our initial assumption that $\mathrm{Suc}_{T,\mu}(F) \leq 1-\varepsilon$).
2. If $\mathcal{R}$ makes its next query at table j, then
$$\mathbb{E}[X(j,t+1)|v_t^1,\ldots,v_t^k] \leq X(j,t), \text{ and}$$
$$X(j',t+1) = X(j',t) \quad \forall j' \neq j.$$
(Follows from definition of $X(j,t)$ and the fact that the inputs remain independent.)

So, choosing input j to query next is like making an unfavorable gamble at the j-th table!

Unfavorable gambles

Two important properties:

1. For all j,

   $$X(j, 0) \leq 1 - \varepsilon$$

   (follows from our initial assumption that $\mathrm{Suc}_{T,\mu}(F) \leq 1 - \varepsilon$).

2. If $\mathcal{R}$ makes its next query at table j, then

   $$\mathbb{E}[X(j, t+1) | v_t^1, \dots, v_t^k] \leq X(j, t), \text{ and}$$

   $$X(j', t+1) = X(j', t) \quad \forall j' \neq j.$$

   (Follows from definition of $X(j, t)$ and the fact that the inputs remain independent.)

So, choosing input j to query next is like making an <u>unfavorable gamble</u> at the j-th table!

Bounding expectations

- It follows that

$$\mathbb{E}\left[\prod_j X(j, t+1) \middle| v_t^1, \ldots, v_t^k\right] \leq \prod_j X(j, t),$$

so

$$\mathbb{E}\left[\prod_j X(j, t)\right] \leq \prod_j X(j, 0) \leq (1-\varepsilon)^k$$

for all $0 \leq t \leq M$.

Bounding expectations

- It follows that

$$\mathbb{E}\left[\prod_j X(j, t+1) \middle| v_t^1, \ldots, v_t^k\right] \le \prod_j X(j, t),$$

so

$$\mathbb{E}\left[\prod_j X(j, t)\right] \le \prod_j X(j, 0) \le (1 - \varepsilon)^k$$

for all $0 \le t \le M$.

Success probability

- What do the final fortunes $X(j, M)$ tell us?
- If input j is under-budget after M queries, then for any guess $y \in \{0, 1\}$,

$$\Pr\left[y = f(\mathbf{x}^j) \middle| v_M^1, \ldots, v_M^k\right] \leq X(j, M).$$

- If j is over-budget, then (trivially) for any y,

$$\Pr\left[y = f(\mathbf{x}^j) \middle| v_M^1, \ldots, v_M^k\right] \leq 1 = 2 \cdot (1/2) = 2X(j, M).$$

- Also, these k events are independent, after we condition on the guesses $(y_1, \ldots, y_k)$ produced by $\mathcal{R}$.

Success probability

- What do the final fortunes $X(j, M)$ tell us?
- If input j is under-budget after M queries, then for any guess $y \in \{0, 1\}$,

$$\Pr\left[y = f(\mathbf{x}^j) \middle| v_M^1, \ldots, v_M^k\right] \leq X(j, M).$$

- If j is over-budget, then (trivially) for any y,

$$\Pr\left[y = f(\mathbf{x}^j) \middle| v_M^1, \ldots, v_M^k\right] \leq 1 = 2 \cdot (1/2) = 2X(j, M).$$

- Also, these k events are independent, after we condition on the guesses $(y_1, \ldots, y_k)$ produced by $\mathcal{R}$.

Success probability

- What do the final fortunes $X(j, M)$ tell us?
- If input j is under-budget after M queries, then for any guess $y \in \{0, 1\}$,

$$\Pr\left[y = f(\mathbf{x}^j) \middle| v_M^1, \ldots, v_M^k\right] \leq X(j, M).$$

- If j is over-budget, then (trivially) for any y,

$$\Pr\left[y = f(\mathbf{x}^j) \middle| v_M^1, \ldots, v_M^k\right] \leq 1 = 2 \cdot (1/2) = 2X(j, M).$$

- Also, these k events are independent, after we condition on the guesses $(y_1, \ldots, y_k)$ produced by $\mathcal{R}$.

Success probability

- What do the final fortunes $X(j, M)$ tell us?
- If input j is under-budget after M queries, then for any guess $y \in \{0, 1\}$,

$$\Pr\left[y = f(\mathbf{x}^j) \middle| v_M^1, \ldots, v_M^k\right] \leq X(j, M).$$

- If j is over-budget, then (trivially) for any y,

$$\Pr\left[y = f(\mathbf{x}^j) \middle| v_M^1, \ldots, v_M^k\right] \leq 1 = 2 \cdot (1/2) = 2X(j, M).$$

- Also, these k events are independent, after we condition on the guesses $(y_1, \ldots, y_k)$ produced by $\mathcal{R}$.

Success probability

- Thus,

$$\Pr\left[\mathcal{R} \text{ computes } F^{\otimes k} \middle| v_M^1, \ldots, v_M^k\right] \leq 2^{|B|} \prod_j X(j, M),$$

where

$$B \triangleq \{j : \text{ input } j \text{ is over-budget after } M \text{ steps}\}.$$

- Counting queries, we have

$$|B| < M/T \leq (\alpha\varepsilon T k)/T = \alpha\varepsilon k.$$

- Thus

$$\Pr\left[\mathcal{R} \text{ computes } F^{\otimes k}\right] \leq 2^{\alpha\varepsilon k} \mathbb{E}\left[\prod_j X(j, M)\right]$$
$$\leq 2^{\alpha\varepsilon k}(1-\varepsilon)^k.$$

QED

Success probability

- Thus,

$$\Pr\left[\mathcal{R} \text{ computes } F^{\otimes k} \middle| v_M^1, \ldots, v_M^k\right] \leq 2^{|B|} \prod_j X(j, M),$$

where

$$B \triangleq \{j : \text{ input } j \text{ is over-budget after } M \text{ steps}\}.$$

- Counting queries, we have

$$|B| < M/T \leq (\alpha\varepsilon Tk)/T = \alpha\varepsilon k.$$

- Thus

$$\Pr\left[\mathcal{R} \text{ computes } F^{\otimes k}\right] \leq 2^{\alpha\varepsilon k} \mathbb{E}\left[\prod_j X(j, M)\right] \leq 2^{\alpha\varepsilon k}(1-\varepsilon)^k.$$

QED

Success probability

- Thus,

$$\Pr\left[\mathcal{R} \text{ computes } F^{\otimes k} \middle| v_M^1, \ldots, v_M^k\right] \le 2^{|B|} \prod_j X(j, M),$$

where

$$B \triangleq \{j : \text{ input } j \text{ is over-budget after } M \text{ steps}\}.$$

- Counting queries, we have

$$|B| < M/T \le (\alpha\varepsilon Tk)/T = \alpha\varepsilon k.$$

- Thus

$$\Pr\left[\mathcal{R} \text{ computes } F^{\otimes k}\right] \le 2^{\alpha\varepsilon k} \mathbb{E}\left[\prod_j X(j, M)\right]$$
$$\le 2^{\alpha\varepsilon k}(1-\varepsilon)^k.$$

QED

Success probability

- Thus,

$$\Pr\left[\mathcal{R} \text{ computes } F^{\otimes k} \middle| v_M^1, \ldots, v_M^k\right] \le 2^{|B|} \prod_j X(j, M),$$

where

$$B \triangleq \{j : \text{ input } j \text{ is over-budget after } M \text{ steps}\}.$$

- Counting queries, we have

$$|B| < M/T \le (\alpha\varepsilon T k)/T = \alpha\varepsilon k.$$

- Thus

$$\Pr\left[\mathcal{R} \text{ computes } F^{\otimes k}\right] \le 2^{\alpha\varepsilon k} \mathbb{E}\left[\prod_j X(j, M)\right]$$
$$\le 2^{\alpha\varepsilon k}(1-\varepsilon)^k.$$

QED

Seeking generalizations

- Many other DPT variants we'd like to prove. But our previous technique was rather specific.
- We used the fact $X(j, t) \geq 1/2$, which followed since F was Boolean. Result weakens as output alphabet grows.
- Bounding $\mathbb{E}\left[\prod_{j,M} X(j, M)\right]$ helped us upper-bound

  $$\Pr[\mathcal{R} \text{ correct on } \underline{\text{all}} \text{ inputs}],$$

  but we'd like to even bound

  $$\Pr[\mathcal{R} \text{ correct on } \underline{\text{most}} \text{ inputs}].$$

- Next: an approach to address both these issues.

Seeking generalizations

- Many other DPT variants we'd like to prove. But our previous technique was rather specific.
- We used the fact $X(j, t) \geq 1/2$, which followed since F was Boolean. Result weakens as output alphabet grows.
- Bounding $\mathbb{E}\left[\prod_{j,M} X(j, M)\right]$ helped us upper-bound

$$\Pr[\mathcal{R} \text{ correct on } \underline{\text{all}} \text{ inputs}],$$

but we'd like to even bound

$$\Pr[\mathcal{R} \text{ correct on } \underline{\text{most}} \text{ inputs}].$$

- Next: an approach to address both these issues.

Seeking generalizations

- Many other DPT variants we'd like to prove. But our previous technique was rather specific.
- We used the fact $X(j,t) \geq 1/2$, which followed since F was Boolean. Result weakens as output alphabet grows.
- Bounding $\mathbb{E}\left[\prod_{j,M} X(j,M)\right]$ helped us upper-bound

$$\Pr[\mathcal{R} \text{ correct on } \underline{\text{all}} \text{ inputs}],$$

but we'd like to even bound

$$\Pr[\mathcal{R} \text{ correct on } \underline{\text{most}} \text{ inputs}].$$

- Next: an approach to address both these issues.

Seeking generalizations

- Many other DPT variants we'd like to prove. But our previous technique was rather specific.
- We used the fact $X(j, t) \geq 1/2$, which followed since F was Boolean. Result weakens as output alphabet grows.
- Bounding $\mathbb{E}\left[\prod_{j,M} X(j, M)\right]$ helped us upper-bound

  $$\Pr[\mathcal{R} \text{ correct on } \underline{\text{all}} \text{ inputs}],$$

  but we'd like to even bound

  $$\Pr[\mathcal{R} \text{ correct on } \underline{\text{most}} \text{ inputs}].$$

- Next: an approach to address both these issues.

Seeking generalizations

Consider a more general setting than ours, in which a gambler plays games at k tables. Assume:

1. Gambler has an initial endowment of $(1 - \varepsilon)$ at every table.
2. Cannot transfer funds between tables, or go into debt at a table.
3. All games 'favor the house' (in expectation).
4. Gambler can choose which game to play next, at which table.

Seeking generalizations

Consider a more general setting than ours, in which a gambler plays games at k tables. Assume:

1. Gambler has an initial endowment of $(1 - \varepsilon)$ at every table.
2. Cannot transfer funds between tables, or go into debt at a table.
3. All games 'favor the house' (in expectation).
4. Gambler can choose which game to play next, at which table.

Seeking generalizations

Consider a more general setting than ours, in which a gambler plays games at k tables. Assume:

1. Gambler has an initial endowment of $(1 - \varepsilon)$ at every table.
2. Cannot transfer funds between tables, or go into debt at a table.
3. All games 'favor the house' (in expectation).
4. Gambler can choose which game to play next, at which table.

Seeking generalizations

Consider a more general setting than ours, in which a gambler plays games at k tables. Assume:

1. Gambler has an initial endowment of $(1 - \varepsilon)$ at every table.
2. Cannot transfer funds between tables, or go into debt at a table.
3. All games 'favor the house' (in expectation).
4. Gambler can choose which game to play next, at which table.

Seeking generalizations

Consider a more general setting than ours, in which a gambler plays games at k tables. Assume:

1. Gambler has an initial endowment of $(1 - \varepsilon)$ at every table.
2. Cannot transfer funds between tables, or go into debt at a table.
3. All games 'favor the house' (in expectation).
4. Gambler can choose which game to play next, at which table.

Seeking generalizations

- Suppose the gambler wishes to reach a fortune of 1 at every table.
- Reasoning similar to before gives

$$\Pr[\text{success}] \leq (1-\varepsilon)^k.$$

$=$ winning odds if gambler plays independent 'all or nothing' bets at each table!

Seeking generalizations

- Suppose the gambler wishes to reach a fortune of 1 at every table.
- Reasoning similar to before gives

$$\Pr[\text{success}] \leq (1 - \varepsilon)^k.$$

$=$ winning odds if gambler plays independent 'all or nothing' bets at each table!

Seeking generalizations

- Suppose the gambler wishes to reach a fortune of 1 at every table.
- Reasoning similar to before gives

$$\Pr[\text{success}] \leq (1 - \varepsilon)^k.$$

$=$ winning odds if gambler plays independent 'all or nothing' bets at each table!

Seeking generalizations

- Now suppose the gambler's goal is just to reach a fortune of 1 at 'many' tables.
- Here 'many' is specified by some monotone collection $\mathcal{C}$ of subsets of $\{1, \ldots, k\}$.
  That is, $(A \in \mathcal{C} \wedge B \supseteq A) \Rightarrow B \in \mathcal{C}$.
- It's natural to ask: does the 'all or nothing' strategy remain optimal?

Lemma ('Gambling lemma'—informal)

YES! Under assumptions 1-4 above, independent all-or-nothing bets are an optimal strategy.

- Proof is a simple induction.

Seeking generalizations

- Now suppose the gambler's goal is just to reach a fortune of 1 at 'many' tables.
- Here 'many' is specified by some monotone collection $\mathcal{C}$ of subsets of $\{1, \ldots, k\}$.
  That is, $(A \in \mathcal{C} \wedge B \supseteq A) \Rightarrow B \in \mathcal{C}$.
- It's natural to ask: does the 'all or nothing' strategy remain optimal?

Lemma ('Gambling lemma'—informal)

YES! Under assumptions 1-4 above, independent all-or-nothing bets are an optimal strategy.

- Proof is a simple induction.

Seeking generalizations

- Now suppose the gambler's goal is just to reach a fortune of 1 at 'many' tables.
- Here 'many' is specified by some monotone collection $\mathcal{C}$ of subsets of $\{1, \ldots, k\}$.
  That is, $(A \in \mathcal{C} \wedge B \supseteq A) \Rightarrow B \in \mathcal{C}$.
- It's natural to ask: does the 'all or nothing' strategy remain optimal?

Lemma ('Gambling lemma'—informal)

YES! Under assumptions 1-4 above, independent all-or-nothing bets are an optimal strategy.

- Proof is a simple induction.

Seeking generalizations

- Now suppose the gambler's goal is just to reach a fortune of 1 at 'many' tables.
- Here 'many' is specified by some monotone collection $\mathcal{C}$ of subsets of $\{1, \ldots, k\}$.
  That is, $(A \in \mathcal{C} \wedge B \supseteq A) \Rightarrow B \in \mathcal{C}$.
- It's natural to ask: does the 'all or nothing' strategy remain optimal?

Lemma ('Gambling lemma'—informal)

YES! Under assumptions 1-4 above, independent all-or-nothing bets are an optimal strategy.

- Proof is a simple induction.

Seeking generalizations

- Now suppose the gambler's goal is just to reach a fortune of 1 at 'many' tables.
- Here 'many' is specified by some monotone collection $\mathcal{C}$ of subsets of $\{1, \ldots, k\}$.
  That is, $(A \in \mathcal{C} \wedge B \supseteq A) \Rightarrow B \in \mathcal{C}$.
- It's natural to ask: does the 'all or nothing' strategy remain optimal?

Lemma ('Gambling lemma'—informal)

YES! Under assumptions 1-4 above, independent all-or-nothing bets are an optimal strategy.

- Proof is a simple induction.

Further results

With this Gambling Lemma, we can derive a variety of new direct product-type theorems for query complexity:

- **threshold** DPTs;
- an **XOR lemma**;
- DPTs for **worst-case error**;

Further results

With this Gambling Lemma, we can derive a variety of new direct product-type theorems for query complexity:

- **threshold** DPTs;
- an **XOR lemma**;
- DPTs for **worst-case error**;

Even more DPTs...

- DPTs for **search problems** and **errorless heuristics**;
- DPTs for **decision tree size** (greatly improving on earlier ones **[IRW94]**);
- DPTs for **interactive puzzles**, in which the algorithm talks with dynamic entities rather than querying static strings.

What's next?

- Our proofs crucially used the conditional independence property of k independent inputs queried by an algorithm.
- A simple analogue of this property is missing in richer computational models (including the quantum query model), which holds us back.
- But perhaps the ideas in our work can be helpful beyond the randomized query model.

What's next?

- Our proofs crucially used the conditional independence property of k independent inputs queried by an algorithm.
- A simple analogue of this property is missing in richer computational models (including the quantum query model), which holds us back.
- But perhaps the ideas in our work can be helpful beyond the randomized query model.

What's next?

- Our proofs crucially used the conditional independence property of k independent inputs queried by an algorithm.
- A simple analogue of this property is missing in richer computational models (including the quantum query model), which holds us back.
- But perhaps the ideas in our work can be helpful beyond the randomized query model.

Thanks!