Information Processing Letters 133 (2018) 16–20

Contents lists available at ScienceDirect

Information Processing Letters

www.elsevier.com/locate/ipl

The choice and agreement problems of a random function

Or Meir [a,*,1], Avishay Tal [b,2]

[a] Department of Computer Science, Haifa University, Haifa 31905, Israel
[b] Department of Computer Science, Stanford University, Stanford, CA 94305, United States

ARTICLE INFO

Article history:
Received 9 October 2017
Received in revised form 19 December 2017
Accepted 22 December 2017
Available online 8 January 2018
Communicated by Marcin Pilipczuk

Keywords:
Computational complexity
Agree
Choose
Direct sum
KRW composition conjecture

ABSTRACT

The direct-sum question is a classical question that asks whether performing a task on m independent inputs is m times harder than performing it on a single input. In order to study this question, Beimel et al. [3] introduced the following related problems:

- **The choice problem:** Given m distinct instances, choose one of them and solve it.
- **The agreement problem:** Given m distinct instances, output a solution that is correct for at least one of them.

It is easy to see that these problems are no harder than performing the original task on a single instance, and it is natural to ask *whether it is strictly easier or not*. In particular, proving that the choice problem is not easier is necessary for proving a direct-sum theorem, and is also related to the KRW composition conjecture [12].
In this note, we observe that in a variety of computational models, if f is a random function then with high probability its corresponding choice and agreement problem are not much easier than computing f on a single instance (as long as m is noticeably smaller than 2^n).

1. Introduction

The direct-sum question is a classical question that asks whether performing a task on m independent inputs is m times harder than performing it on a single input. More generally, one may ask whether performing multiple independent tasks in parallel is as hard as performing each of them separately. It will be convenient to use the following notation.

Definition 1. Let $T_1, \ldots, T_m$ be computational tasks. The direct-sum problem $\mathsf{sum}_{T_1,\ldots,T_m}$ is the following task: given m inputs $x_1, \ldots, x_m$ for $T_1, \ldots, T_m$ respectively, output a vector $(y_1, \ldots, y_m)$ such that y_i is a correct solution for T_i on x_i for every i.

The direct-sum question asks whether the complexity of $\mathsf{sum}_{T_1,\ldots,T_m}$ is the sum of the individual complexities of $T_1, \ldots, T_m$. This natural question was studied in a variety of computational models (see, e.g., [2,4,5,8,9,11,14–16]), and the answer turns out to be positive in some models and negative in others. In order to study this question, Beimel et al. [3] (following Ambainis et al. [1]) considered the following related problems.

Definition 2 *([3]).* Let $T_1, \ldots, T_m$ be computational tasks.

* Corresponding author.
E-mail addresses: ormeir@cs.haifa.ac.il (O. Meir), avishay.tal@gmail.com (A. Tal).

[1] Partially supported by the Israel Science Foundation (grant No. 1445/16).
[2] Partially supported by a Motwani Postdoctoral Fellowship and by NSF grant CCF-1749750.

https://doi.org/10.1016/j.ipl.2017.12.007
0020-0190/

- The *choice problem* $\mathsf{choose}_{T_1,\ldots,T_m}$ is the following task: Given m inputs $x_1,\ldots,x_m$ for $T_1,\ldots,T_m$ respectively, output a pair (i,y) such that y is a correct solution for T_i on x_i.
- The *agreement problem* $\mathsf{agree}_{T_1,\ldots,T_m}$ is the following task: Given m inputs $x_1,\ldots,x_m$ for $T_1,\ldots,T_m$ respectively, output a value y such that y is a correct solution for T_i on x_i for some $i\in[m]$.

It is easy to see that the agreement task is not harder than the choice task, and that both tasks are not harder than performing the easiest task T_i on a single input. [3] asked whether we can prove that the choice and agreement problem are *not strictly easier than the easiest task* T_i. In addition to being interesting in its own right, this question has the following motivations:

- Proving that the choice problem $\mathsf{choose}_{T_1,\ldots,T_m}$ is *not* strictly easier than the easiest task T_i is *necessary for proving a direct-sum theorem*: To see it, observe that if the choice problem $\mathsf{choose}_{T_1,\ldots,T_m}$ is strictly easier than the easiest task T_i, then the complexity of $\mathsf{sum}_{T_1,\ldots,T_m}$ is strictly less than the sum of the individual complexities of $T_1,\ldots,T_m$ (since one can solve one of the tasks using the best algorithm for $\mathsf{choose}_{T_1,\ldots,T_m}$, and then solve each of the remaining tasks individually).
- Since the agreement problem $\mathsf{agree}_{T_1,\ldots,T_m}$ is not harder than the choice problem $\mathsf{choose}_{T_1,\ldots,T_m}$, one can prove lower bounds for $\mathsf{choose}_{T_1,\ldots,T_m}$ by proving lower bounds for $\mathsf{agree}_{T_1,\ldots,T_m}$.

In this note, we consider the special case where all the tasks $T_1,\ldots,T_m$ are the same task T. Observe that in this case special case, it is trivial to prove that the choice and agreement problems are not strictly easier than solving T for the following reason: Solving T on an input x reduces to solving $\mathsf{choose}_{T,\ldots,T}$ and $\mathsf{agree}_{T,\ldots,T}$ on m copies of x. In order to avoid this trivial case, we require the m inputs to be distinct, resulting in the following definition.

Definition 3. Let T be a computational task.

- The *m-fold choice problem* choose_T^m is the following task: Given m distinct inputs $x_1,\ldots,x_m$ for T, output a pair (i,y) such that y is a correct solution for T on x_i.
- The *m-fold agreement problem* agree_T^m is the following task: Given m distinct inputs $x_1,\ldots,x_m$ for T, output a value y such that y is a correct solution for T on x_i for some $i\in[m]$.

It is again natural to ask whether choose_T^m and agree_T^m are strictly easier than T on its own. In particular, note that the foregoing motivation still holds: proving that choose_T^m is not easier than T is necessary for proving a direct-sum theorem for T (i.e., that the complexity of $\mathsf{sum}_{T,\ldots,T}$ is m times the complexity of T).

In this note, we observe that in a variety of computational models, the answer to this question is negative when T is the task of computing a *random function* f, with high probability over f. Intuitively, this result holds in every model in which the hardness of a random function can be proved using a counting argument, including Boolean circuits, formulas, decision trees, etc.

In order to make this intuition more precise, consider a computational model that comes with some size measure (e.g., number of wires for circuits, depth for decision trees, etc.). We use the term *computer* to refer to a specific instantiation of this model (e.g. a specific circuit, a specific decision tree, etc.). Let $N(s,n)$ denote the number of distinct Boolean functions over n bits that are computed by a computer of size at most s. Then, the standard counting argument says the probability that a random function over n bits can be computed by a computer of size at most s is at most

$$\frac{N(s,n)}{2^{2^n}}.$$

We prove the following observation.

Theorem 4. *Fix a computational model that comes with some size measure, and let $N(s,n)$ be defined as above. Let $n,m,s\in\mathbb{N}$, and let $f:\{0,1\}^n\to\{0,1\}$ be a uniformly distributed function. Then, the probability that choose_f^m or agree_f^m can be decided by a computer of size at most s is at most*

$$\frac{N(s,m\cdot n)\cdot\binom{2^n}{\le 2m-2}}{2^{2^n}}$$

where $\binom{a}{\le b}\stackrel{\text{def}}{=}\binom{a}{0}+\binom{a}{1}+\binom{a}{2}+\ldots+\binom{a}{b}$ for non-negative integers $a\ge b$.

Observe that the expression $\binom{2^n}{\le 2m-2}$ in the latter probability is negligible compared to 2^{2^n} unless m is very close to 2^n. Thus, as long as m is not too large, this factor will not affect the probability significantly. On the other hand, the fact that we count the number of computers over $m\cdot n$ variables rather than n variables can affect the probability significantly, depending on the function $N(s,n)$, and this is the bottleneck in the following application.

The following corollary lists the immediate consequences of Theorem 4 for some important computational models. Essentially, it says that for Boolean circuits, choose_f^m and agree_f^m are as hard as a random function when $m<\varepsilon\cdot 2^n$. For other models (formulas, depth complexity, decision trees), we get a slightly worse lower bound (although we could have gotten the "right" lower bound when $m=\mathrm{poly}(n)$).

Corollary 5. *There exists a universal constant $\varepsilon>0$ such that the following holds. Let $f:\{0,1\}^n\to\{0,1\}$ be a uniformly distributed function and let $m\le\varepsilon\cdot 2^n$. Then, each of the following events occurs with probability $1-o(1)$ (where the $o(1)$ is a decreasing function of n):*

- *The circuit-size complexity of choose_f^m and agree_f^m is $\Omega(\frac{2^n}{n})$.*
- *The formula-size complexity of choose_f^m and agree_f^m is $\Omega(\frac{2^n}{n})$.*