

Holographic Computation of Balanced Succinct Permanent Instances*

(Preliminary Version)

Shlomi Dolev¹, Nova Fandina¹, Joseph Rosen²

¹ Department of Computer Science

Ben Gurion University of the Negev, Israel

² Department of Electrical and Computer Engineering

Ben Gurion University of the Negev, Israel

Abstract. Galperin and Wigderson proposed a succinct representation for graphs, that uses number of bits that is logarithmic in the number of nodes. They proved complexity results for various decision problems on graph properties, when the graph is given in a succinct representation. Later, Papadimitriou and Yannakakis showed, that under the same succinct encoding method, certain class of decision problems on graph properties becomes exponentially hard. In this paper we consider the complexity of the Permanent problem when the graph/matrix is given in a restricted succinct representation. We present an optical architecture that is based on the holographic concept for solving balanced succinct permanent problem. Holography enables to have exponential copying (roughly, $n \times n$ in each iteration) rather than constant copying (e.g., doubling in each iteration).

1 Introduction

The paradigm change in current computing from a single core to multi-core is a dramatic change, a change from sequential computing to parallel computing. Optical computing has benefits in parallel computations. Previous recent works suggested optical solutions for *NP*-Complete problems. In this work we present optical solution for a restricted variant of *NEXP* time hard task.

Optical holography is a technique for recording/reading data by using physical properties of coherent light, interference and diffraction. To record a simple *hologram* the laser beam is split into two separate beams of light. One beam (an object beam) illuminates the object. As a result some light reflects from the object and falls on the recording medium. Simultaneously the second beam (a reference beam) is directed to the same recording medium. The interaction of the two beams creates an interference

* Partially supported by the Lynne and William Frankel Center for Computer Science, Ben-Gurion University of the Negev, Israel. ICT Programme of the European Union under contract number FP7-21570 (FRONTS), and Rita Altura Trust Chair in Computer Sciences. Emails: {dolev, fandina}@cs.bgu.ac.il, rosen@ee.bgu.ac.il

pattern, called a hologram of an object. When the original reference beam illuminates the hologram, diffraction occurs and the object beam is reconstructed. The light observed by eyes defines the original object in three dimensions.

There are various applications of holography in different fields. For example in security, medicine and art. We are interested in *holographic processing and holographic data storage*, as techniques for processing and maintaining data.

Holographic data storage approach encodes the data in holograms that are stored in the volume of a photosensitive medium, rather than (the typical) two dimensional storage devices. Thus allowing to store a huge amount of data that is proportional to the volume of the storage device. Today's technology enables to write approximately a terabit of data into holographic crystal of one cubic centimeter. Future technology may significantly increase the amount of information [5]. In addition to the high density, holographic data storage provides an ability for fast parallel reading/writing by using a single flash of laser light. Modern holographic memory devices allow data transfer rate as high as a billion bits per second.

All these advantages have inspired us to start a farther investigation of holographic data storage taking it in the direction of computing rather than only storage.

The current paper introduces a new approach that may be used in employing optics for solving exponential hard computational problem.

Definition 1. Let A be an $n \times n$ matrix, such that:

$$\forall a_{ij} \in A, \quad a_{ij} \in Z, \quad 0 \leq a_{ij} \leq p$$

for some natural number p . A permanent of matrix A is defined as:

$$\text{perm}(A) = \sum_{\sigma} \prod_{i=1}^n A_{i\sigma(i)}$$

Where the summation is over all the permutations σ of n elements.

In [7] Valiant showed that the permanent problem of integer matrix is $\#P$ -Complete. In [4] Lipton proved that the Permanent is *random-self-reducible* implying that the permanent is hard on the worst case as it is hard on the average case.

In order to find a problem that is *NEXP time hard on average*, we refer to the notation of *succinct representation* of graph, that was proposed by Galperin and Wigderson [3]. Papadimitriou and Yannakakis showed that certain *NP-Complete* problems on graph with succinct inputs are *NEXP time hard* [6].

We consider the permanent problem with succinctly represented inputs, as follows:

Definition 2. *Succinct Permanent problem defined as:*

input: An $O(\log^k n)$ sized boolean circuit C representing an $n \times n$ integer matrix A (with bounded entries) where k is some constant integer.

output: the permanent of matrix A .

Boolean circuits with only one output gate represent a binary matrix. To represent a general integer matrix a circuit with $O(k \times \log n)$ output gates can be used. We can represent such a circuit with $O(k \times \log n)$ number of circuits with one output gate. Each output gate corresponds to one bit in the binary representation of the integer number. Therefore, this scheme of representation enables to encode integer matrices with both positive and negative values up to $O(n^k)$.

Notice, that the length of representation of the value of permanent of succinctly represented matrix can be exponential in the size of the input. Therefore, we suggest a decision version of the permanent with succinct inputs.

Definition 3. *Succinct Zero-Permanent problem defined as:*

input: An $O(\log^k n)$ sized boolean circuit C representing an $n \times n$ integer matrix A (with bounded entries) where k is some constant integer.

output: $\text{Permanent}(A) == 0$.

The aim of this paper is to establish the computational complexity of Succinct Zero-Permanent problem and introduce a new efficient optical device for solving it. In the sequel we prove that the Succinct Zero-Permanent problem is NEXP time hard and present a new efficient optical device for solving instances of the Succinct Zero-Permanent that are represented by balanced circuits, or random combination of such circuits.

The rest of the paper is organized as follows: in Section 2 the complexity of Succinct Permanent problem is considered. Section 3 describes the input for the optical device. In Sections 4 and 5 we describe the integrated part of the optical architecture and its holographic implementation. The description of the complete optical device for solving succinct permanent problem is provided in Section 6. We complete the paper with discussions and certain directions for future research.

2 Complexity of the Succinct Permanent Problem

Various optical solvers for NP Complete problems were proposed. They are beneficial due to the assumption that $P \neq NP$. We suggest considering computational problems that are provable harder than the problems in P. Namely, NEXP time hard problems.

Definition 4. *Let ϕ be a boolean formula. Then, $\#\phi$ denote the number of satisfying assignments of ϕ . Let C_ϕ be a succinct circuit representation of formula ϕ . Then, $\#C_\phi$ denote the number of satisfying assignments of ϕ .*

Next we prove that the Succinct Zero-Permanent is NEXP time hard in the worst case.

Theorem 1. *Succinct Zero-Permanent is NEXP time hard.*

Proof. By results in [6] Succ-3SAT decision problem is NEXP time hard. We reduce this problem to Succinct Zero-Permanent in the following way. In [7] Valiant presented a polynomial time reduction from #3 SAT to the Permanent problem of integer matrix.

Given an instance ϕ of the 3SAT, the reduction constructs a directed, weighted graph G , with weights $-1, 0, 1, 2, 3$, such that

$$\text{Permanent}(G) = 4^{t(\phi)} \times s(\phi)$$

where $t(\phi)$ is twice the number of occurrences of literals in ϕ , minus the number of clauses in ϕ , and $s(\phi)$ is the number of satisfying assignments of ϕ .

This construction is based on the composition of graph structures in highly regular fashion. An output graph G is obtained by combination a polynomial number of *track*, *interchange* and *junction* structures. The order and type of structures are predefined by boolean formula and can be efficiently obtained from the formula. Namely, given two $c|\log x|$ -bit integers, the indices of nodes of graph G , it can be determined in polynomial time (of the length of the integers) whether there is an edge between these nodes. The algorithm reads at most polylogarithmic number bits of ϕ . Call this algorithm A .

Let f denote the reduction transformation of Valiant. Given C_ϕ - an $O(\log^k n)$ -sized boolean circuit representation of ϕ , combine an algorithm A with C_ϕ to obtain a polylogarithmic description of the graph $G = f(\phi)$. Finally, using a relation between $\text{Permanent}(G)$ and the number of satisfying assignments of ϕ , we obtain: if $\text{Permanent}(G) = 0$, then $\#\phi = 0$, and therefore there is no satisfying assignments for ϕ ; if $\text{Permanent}(G) > 0$, then $\#\phi > 0$ implying the existence of satisfying assignments for ϕ . Note, that $4^{t(\phi)}$ is a positive integer number. \square

Therefore, the Succinct Zero-Permanent of integer matrix (with both positive and negative values) problem is a good candidate for being solved with optical computing techniques. But does the hardness on the worst case suffice? We are interesting in problem that is hard on most of the inputs or on a big fraction of the inputs. We conjecture that the Succinct Zero-Permanent problem is hard on average as it is hard on the worst case. That implies that it is exponentially hard to solve this problem in most of the inputs.

3 Circuits as Inputs

We start with a description of the format of the input data supported by the Succinct Permanent solver device.

Encoding scheme for input. As was mentioned earlier, we solve a computational problem assuming succinctly encoded inputs. Various techniques for a short description of the data were proposed. We use the Small Circuit Representation (SCR) scheme, proposed by Galperin and Wigderson [3].

The input is a boolean $O(\log^k n)$ sized circuit, for some k , representing a particular matrix of size $n \times n$. The exact form of such SCR circuits is not a theoretical concern. In other words, the internal configuration, such as, the types of the used logic gates, fan-in/fan-out of this gates, is not important for achieving the complexity results. It remains the same, as long as the chosen model of circuits is complete, and as long as $O(\log^k n)$ space is used. We call the model of circuits with n inputs *complete*, if and only if, for each boolean function with n variables there is a boolean circuit from the model that computes this function.

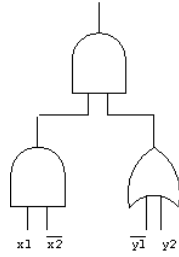


Fig. 1. Small Boolean circuit that represents a 4×4 matrix.

0	0	0	0
0	0	0	0
1	1	0	1
0	0	0	0

Fig. 2. Matrix that represented by circuit in Fig. 1

We will consider circuits that contain only AND, OR and NOT gates. In particular, we choose to deal with boolean formulas, i.e, boolean circuits with $f_{an-out} = 1$ for all gates. It is a well known fact, that such a model is complete [8]. Without loss of generality we can assume that NOT operations are applied only to the input variables of the circuit. If negations appear higher up in the circuit we can move them down to the inputs, using DeMorgan's law. By using the associative and the commutative properties of the AND and OR operations we assume that all gates in the circuit has $f_{an-in} = 2$. Thus, we consider boolean circuits that have a topology of a binary tree.

Furthermore, in this work, we will assume a topology of a complete binary tree without duplication of the inputs. We call such a circuits a *balanced* boolean circuits. To obtain a larger set of boolean circuits, we combine a random set of balanced circuits with OR or AND operations. We believe that such a random set of instances is hard on average. Fig. 1 and Fig. 2 depict a boolean circuit and the 4×4 matrix it represents.

The optical architecture we build consists of two main parts: a holographic preprocessing unit and an optical Permanent solver. The preprocessing unit is an optical device that generates all matrices that can be defined by balanced small circuits using a bounded space to store this data. Thus, this device defines the size of the inputs of the Succinct Permanent problem that can be solved by the architecture. For each integer n the Preprocessing unit generates all matrices of size $n \times n$, that have a balanced Small Circuit Representation in $\log(\log n)$ number of iteration steps. This unit runs only once to build such matrices. The Succinct Permanent solver acts as follows: given a balanced boolean circuit representing a matrix, the solver finds an appropriate matrix stored in the medium, and outputs it to the Permanent solver to obtain a solution.

4 Preprocessing Unit

In this section we provide a detailed description of the method used by the Preprocessing unit to generate the matrices. In the preprocessing phase we generate and store all matrices of size $n \times n$, that has a balanced Small Circuit Representation. In this section we describe an algorithm for this task.

The idea is to produce a general matrix that contains all matrices of size $n \times n$ that can be encoded with $O(\log^k n)$ sized balanced boolean circuit. We accomplish this task by using an iterative algorithm. It starts with a general matrix corresponding to boolean circuits with one level of gates. By iterative copying phases, the algorithm extends the general matrix of i leveled circuits to a general matrix corresponding to $i + 1$ leveled circuits.

Definition 5. Let BC_i be the set of all balanced boolean circuits that contains i (full) levels of gates.

Note, that circuits in BC_i have $2 \times 2^{(i-1)}$ input variables, and therefore they represent a $2^{(2^{(i-1)})} \times 2^{(2^{(i-1)})}$ matrices.

Definition 6. Let G_i be an $2^{2^i} \times 2^{(2^{(i+1)}-1)}$ binary matrix. We call it a General Matrix for BC_i , if and only if, for each column J_c of G_i there exists a boolean circuit $C \in BC_i$ such that

$$M(C) = J_c \quad \text{and,}$$

for each $C' \in BC_i$ there exists a column J'_c in G_i such that

$$M(C') = J'_c$$

where $M(C)$ is a matrix encoded by the circuit C .

Note that each column of G_i represents a matrix that can be encoded by circuit with i levels of gates. There are $2^{(2^{(i+1)}-1)}$ such a matrices, each of size $2^{(2^{(i-1)})} \times 2^{(2^{(i-1)})}$. Next we describe the initialization stage and the inductive step.

Initialization stage. Method starts with building an initial General Matrix G_1 . Each column in G_1 is a matrix that can be encoded by a boolean circuit from B_1 .

M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8
0	0	0	1	0	1	1	1
0	0	1	0	1	0	1	1
0	1	0	0	1	1	0	1
1	0	0	0	1	1	1	0

Fig. 3. The General Matrix for BC_1

Induction stage. Algorithm 1, that is described in Fig. 4 extends G_i to G_{i+1} . Algorithm 1 defines a skeleton for G_{i+1} in terms of *blocks* of fixed size (lines 2 to 4). Each cell of G_i turns to be a block of the size of G_i . The order of blocks is exactly the order of cells in G_i . These blocks are filled with data according to G_i (lines 5 to 11). The data contained in G_i is copied into all blocks that correspond to cells with value 1. All the rest of the blocks are filled with the value 0. All blocks together compose the part of G_{i+1} , that we call H^1_{i+1} . See Fig. 5 for a schematic description of the obtained matrix H^1_{i+1} following the first part of an iteration. The second part, H^2_{i+1} , is obtained by negation and reflection applied to H^1_{i+1} (lines 12 to 14 of the code). Finally,

$$G_{i+1} = H^1_{i+1} \circ H^2_{i+1}$$

where the \circ operation is a concatenation.

```

1:  $\diamond$  Build  $H^1_{i+1}$ :
2: define a new binary matrix of size  $2^{(2^{(i+1)})} \times 2^{(2^{(i+2)} - 2)}$ 
3: divide this matrix into  $2^{2^{(i+1)}} \times 2^{(2^{(i+1)} - 1)}$  sized blocks. {As a result we get a structure
   composed of  $2^{2^i}$  rows of blocks, each row contains of  $2^{(2^{(i+1)} - 1)}$  blocks.}
4: enumerate blocks in fashion of matrix indexing. {For  $1 \leq \hat{i} \leq 2^{2^i}, 1 \leq \hat{j} \leq 2^{2^i}$ 
   block  $B_{\hat{i}, \hat{j}}$  is in  $\hat{i}$ 's row and  $\hat{j}$ 's column of blocks.}
5: for  $i = 1$  to  $2^{2^i}$  do
6:   for  $j = 1$  to  $2^{(2^{(i+1)} - 1)}$  do
7:     if ( $G_i[i, j] == 1$ ) then
8:        $B_{\hat{i}, \hat{j}} \leftarrow G_i$ 
9:     end if
10:  end for
11: end for
12:  $\diamond$  Build  $H^2_{i+1}$  :
13:  $H^2_{i+1} \leftarrow \text{NOT}(H^1_{i+1})$ 
14:  $H^2_{i+1} \leftarrow \text{REFLECT}(H^2_{i+1})$ 
15:  $\diamond$   $G_{i+1} \leftarrow (H^1_{i+1}) \circ H^2_{i+1}$ 

```

Fig. 4. Algorithm 1. Create General Matrix

The correctness of the Algorithm 1 is proved in the next theorem.

Theorem 2. Given G_i Algorithm 1 returns G_{i+1} .

Proof. For each cell in G_{i+1} we define a double index

$$\langle \langle \hat{i}, k \rangle, \langle \hat{j}, l \rangle \rangle$$

The first pair indicates the row's coordinates, and the second corresponds to the coordinates of the column. To get the second index in each pair, we enumerate rows and columns in each block.

To prove correctness, we prove the following two lemmas.

0	0	0	Copy of G_1	0	Copy of G_1	Copy of G_1	Copy of G_1
0	0	Copy of G_1	0	Copy of G_1	0	Copy of G_1	Copy of G_1
0	Copy of G_1	0	0	Copy of G_1	Copy of G_1	0	Copy of G_1
Copy of G_1	0	0	0	Copy of G_1	Copy of G_1	Copy of G_1	0

Fig. 5. H^1_2 part of the General Matrix for G_2

Lemma 1. For each $C \in BC_{i+1}$ there is column J of G_{i+1} , s.t. $M(C)=J$.

Proof. Let C be some circuit from BC_{i+1} .

First case: an output gate of C is an AND gate. Let L, R be the circuits, s.t. $C = (L \text{ AND } R)$. Therefore $L, R \in BC_i$.

Hence, by the induction assumption there exist J_L, J_R columns of G_i s.t.

$$M(L) = J_L \quad M(R) = J_R$$

Let j_l, j_r be the indices of these columns in G_i . Consider a column M in G_{i+1} with index $\langle \hat{j}_l, \hat{j}_r \rangle$. We show that

$$M(C) = M$$

and this will complete the proof.

We need to show that for all values of boolean variables $x_1, x_2, \dots, x_{2^i} \quad y_1, y_2, \dots, y_{2^i}$ it holds that

$$(1) \quad M(C)[x_1 \ x_2 \ \dots \ x_{2^i}, \ y_1 \ y_2 \ \dots \ y_{2^i}] = M[x_1 \ x_2 \ \dots \ x_{2^i}, \ y_1 \ y_2 \ \dots \ y_{2^i}] \quad (2)$$

Note, that by definition

$$(1) = C[x_1 \ \dots \ x_{2^i}, \ y_1 \ \dots \ y_{2^i}]$$

where $C[x_1 \ x_2 \ \dots \ x_n]$ is an output value of C on inputs x_1, \dots, x_n .

$$(2) = G_{i+1}[\langle \widehat{x_1 \ \dots \ x_{2^i}}, \ y_1 \ \dots \ y_{2^i} \rangle, \langle \hat{j}_l, \hat{j}_r \rangle]$$

Recall that $C = (L \text{ AND } R)$, therefore,

$$C[x_1 \ \dots \ x_{2^i}, \ y_1 \ \dots \ y_{2^i}] = L[x_1 \ \dots \ x_{2^i}] \text{ AND } R[y_1 \ \dots \ y_{2^i}]$$

If $L[x_1 \dots x_{2^i}] = 0$ then $G_i[x_1 \dots x_{2^i}, j_l] = 0$ and $C[x_1 \dots x_{2^i}, y_1 \dots y_{2^i}] = 0$. Hence, by the construction, block $B_{\widehat{x_1 \dots x_{2^i}}, \hat{j}_l}$ in G_{i+1} is filled with 0. In particular $G_{i+1}[\langle \widehat{x_1 \dots x_{2^i}}, y_1 \dots y_{2^i} \rangle, \langle \hat{j}_l, j_r \rangle] = 0$.
 If $L[x_1 \dots x_{2^i}] = 1$ then $G_i[x_1 \dots x_{2^i}, j_l] = 1$ and $C[x_1 \dots x_{2^i}, y_1 \dots y_{2^i}] = R[y_1 \dots y_{2^i}]$. Hence, by construction, G_i is copied to block $B_{\widehat{x_1 \dots x_{2^i}}, \hat{j}_l}$ in G_{i+1} . Therefore, column with index j_r in this block is a copy of the column with index j_r of G_i . It follows that $G_{i+1}[\langle \widehat{x_1 \dots x_{2^i}}, y_1 \dots y_{2^i} \rangle, \langle \hat{j}_l, j_r \rangle] = R[y_1 \dots y_{2^i}]$.
 This completes the first case of the Lemma 1. Consider the second case.

Second case: an output gate of C is an OR gate.

We make a reduction to the previous case. Define a circuit $\hat{C} = NOT(C)$. By DeMorgan law an output of \hat{C} is an AND gate. Hence, there is column M in G_{i+1} , s.t. the matrix encoded by \hat{C} is M . Therefore, C represents a matrix that is $NOT(M)$. By construction, there is some column \hat{M} in G_{i+1} s.t. $\hat{M} = NOT(M)$. \square

Lemma 2. For each column J of G_{i+1} there is $C \in BC_{i+1}$, s.t. $M(C)=J$.

Proof. Let J be some column of G_{i+1} .

First case: column J is in the (H^1_{i+1}) .

Let $\langle \hat{j}_l, j_r \rangle$ be the index of the J in G_{i+1} . By induction assumption, for the columns C_{j_l} and C_{j_r} in G_i there are circuits L and R in B_i s.t.

$$M(L) = C_{j_l} \quad M(R) = C_{j_r}$$

Define circuit

$$B = (L \text{ AND } R)$$

Note, that $C \in B_{i+1}$ as required.

Use the same arguments as in the previous Lemma to prove that $M(B) = J$.

Second case: column J is in the (H^2_{i+1}) .

Hence, by construction, there is column \hat{J} in (H^1_{i+1}) s.t. $J = NOT(\hat{J})$. Therefore, there is a circuit C that represents \hat{J} . Applying a logical NOT to the output of C and DeMorgan law results in a circuit \hat{C} that represents J . \square

Lemma 1 proves the if condition, namely, that if C is a circuit form BC_{i+1} then there is a column in G_{i+1} that is represented by C , and Lemma 2 proves the only if condition, hence the correctness of the theorem. \square

5 Preprocessing by Holography

We propose an optical architecture using holographic approach to implement the above algorithm. To the best of our knowledge, this is the first technique that can perform exponential copying, namely producing n copies, where n is the size of the input, rather

than producing a constant number of copies in each iteration. The optical device works in phases. In phase i the input is G_i recorded on a photosensitive film and the output is G_{i+1} recorded on a photosensitive film, which is the input to the next phase. For the initialization phase we prepare a film with G_1 . For the induction phase we introduce the system described in Fig. 6.

Implementation of a phase consists of two stages: hologram recording and reconstruction.

Recording stage. Laser beam illuminates an input matrix placed on an input plane. Then the beam is split into two separated beams. One of the beams is directed to a $4f$ correlator that is capable of performing a duplication of the input matrix. Another beam propagates through a magnification system designed to enlarge and duplicate an input matrix. Output beams from both systems are directed to a photograph plate where their interference pattern is recorded.

As shown in Fig. 6 an input binary matrix is represented on a film and placed on the input plane P1. A white rectangle on the slide represents a logical '1' in the matrix, and a black rectangle on the slide represents a logical '0'.

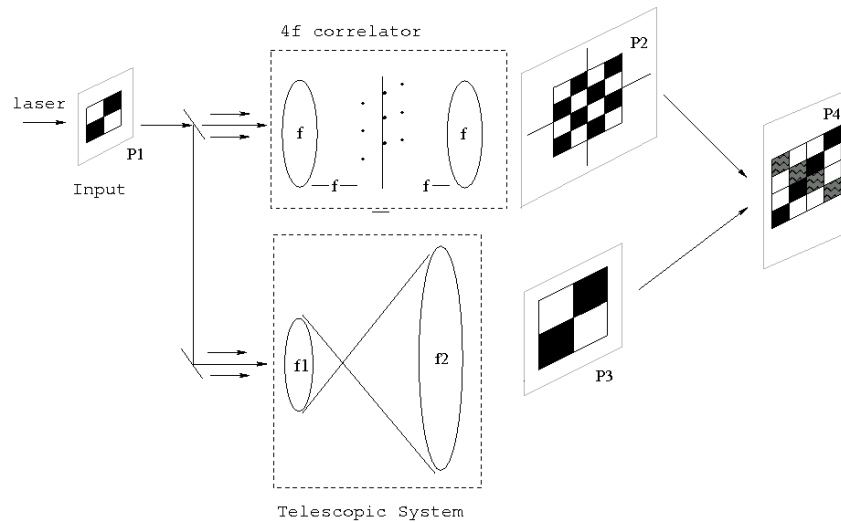


Fig. 6. Hologram Recording

An output from the $4f$ system is presented on plane P2. In each phase we calibrate the distance between holes in a pinholes array to yield the right gap between the duplications, according to the size of the input matrix on the i -th phase.

Enlarged image of the input matrix is on plane P3. A scale rate is determined by focal lengths of two lenses in the following form:

$$M = \frac{f_2}{f_1}$$

Plan P4 represents the image recorded on the film. Interference fringes are created only at positions that correspond to logical '1' at both beams. We develop this film and use it in the reconstruction stage.

Reconstruction stage. To get a final representation of the input matrix to the next phase we perform a reconstruction of the hologram as depicted in Fig. 7.

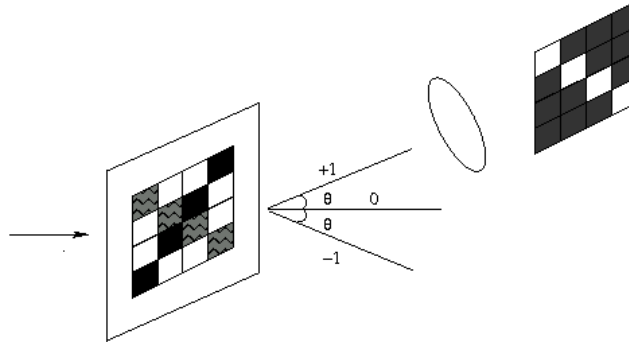


Fig. 7. Hologram Reconstruction

The hologram is illuminated by laser. The beams with direction angle θ are transmitted by lens toward a photosensitive plate. The resulting image is recorded on the film and is used as input matrix for the next phase.

Note, that we described a system for building part of G_{i+1} . In particular, the result of the described phase is H^1_{i+1} . We add a pinhole array device to the telescopic system to enable both enlargement and duplication of the input matrix. Thus, we will get two identical resulting films. Developing one of them in a positive form and another in a negative form to obtain H^1_{i+1} and H^2_{i+1} .

6 An Optical Architecture for Succinct Permanent Problem

Now we present the whole architecture for solving the balanced succinct permanent problem. Given an input in the form of a circuit we need to output its related matrix in a fast manner using optics. Then use the matrix as input to an existing optical architecture for solving a permanent problem.

Several optical devices for solving a permanent problem were proposed. In [2] Dolev and Fitoussi suggest architecture for computing permanent of binary matrices in polynomial time. In this device the answer is obtained by detecting an intensity of the light, that propagates through built set of masks. Thus, to solve Zero-Succinct Permanent problem we only should detect if there some light that propagates through the masks. We assume the existence of such a device and use an optical architecture for solving the permanent problem of integer valued matrix, that was proposed in [1] by Anter and Dolev. This device solves instances of integer matrices with positive values. We should modify this architecture to handle matrices with negative values as well. We

leave this task to our future work. We mention that our recent results conclude that a decision version of Succinct Permanent problem, applying modulo given prime operation is NEXP time hard. Thus, we can solve it using optical device in [1] without any modifications. In fact, for input boolean circuit, we should just detect if the result is zero modulo a prime (randomly chosen from a small set of primes).

We represent a balanced boolean circuit C with $2 \times \log n$ inputs in heap form. Inputs with NOT operation are represented by the label ' N '. All other inputs are represented by label ' G '. AND, OR operations are represented by labels ' A ', ' O ' respectively. Following we present a simple recursive algorithm that returns an index of the column of the General matrix $G_{\log(2 \times \log n)}$ representing a matrix of input circuit C .

<p>Require: heap H representing balanced boolean circuit C with $2 \times \log n$ inputs Ensure: index of the column of General Matrix that corresponds to $M(C)$</p> <pre> 1: $H_1 \leftarrow \text{LeftChild}(H)$ 2: $H_2 \leftarrow \text{RightChild}(H)$ 3: $i \leftarrow \text{MatrixIndex}(H_1)$ 4: $j \leftarrow \text{MatrixIndex}(H_2)$ 5: if $H[1] == 'A'$ then 6: return $(i - 1) \times \text{sizeof}(G_{\log(\log n)}) + j$ 7: end if 8: if $H[1] == 'O'$ then 9: return $(i - 1) \times \text{sizeof}(G_{\log(\log n)}) + j + \text{sizeof}(G_{\log(\log n)})^2$ 10: end if </pre>

Fig. 8. Algorithm 2. Computing an index of General Matrix

The algorithm for computing the index in the matrix, called Algorithm 2, appears in Fig. 8. *Left(Right)Child* denotes a heap rooted at left (right) child of H . Algorithm 2 performs 2 recursive calls (lines 3 to 4). According to the type of the output gate of the input circuit, and the results of the recursive calls, algorithm computes an exact index of the column (lines 5 to 9). *sizeof* refers to the number of the columns of the matrix. It can be computed directly from n . Algorithm 2 finishes the recursive calls when H_1 and H_2 are heaps with only one gate, and thus, it can return an answer immediately. The correctness of the algorithm follows from the correctness of Algorithm 1.

Using Algorithm 2 and acousto-optic modulators the architecture outputs a matrix encoded by balanced boolean circuit. Then, an optical permanent solver provides a solution. Several acousto-optic modulators can select randomly in parallel several balanced boolean circuits. By applying optical AND and OR operations to chosen circuits we obtain a combined circuit that encodes a more general instance. In this way we construct a bigger set of instances. We conjecture that such a random set encode hard instances.

7 Discussion

We presented a solution for the balanced Zero-Succinct Permanent that is a restrictive case of the Zero-Succinct Permanent problem. To solve instances that corresponds to integer matrices (not only binary matrices) we represent an integer matrix by composing a set of $O(k \times \log n)$ balanced boolean circuits, obtaining a short boolean circuit with $O(k \times \log n)$ outputs. Outputs of such a circuit represent in a natural way an integer value in binary basis.

Given the proven hardness on average of the integer Permanent problem, we conjecture that integer Zero-Succinct Permanent problem is hard on the worst case as it is on the average.

The preprocessing stage architecture uses a photosensitive film for data storage. We mention, that holographic data storage may be used instead, for obtaining a faster performance and a reduction in the physical size of the architecture.

References

1. A. Anter and S. Dolev. Optical solution for hard on average #p-complete instances (using exponential space for solving instances of the permanent). *Natural Computing*, 9:891–902, December 2010.
2. S. Dolev and H. Fitoussi. Masking traveling beams: Optical solutions for np-complete problems, trading space for time. *Theor. Comput. Sci.*, 411(6):837–853, 2010.
3. H. Galperin and A. Wigderson. Succinct representations of graphs. *Inf. Control*, 56:183–198, April 1984.
4. R. Lipton. New directions in testing. *Distributed Computing and Cryptography, DIMACS Series on Discrete Mathematics and Theoretical Computer Science*, 2:191–202, 1991.
5. C. R. Moon, L. S. Mattos, B. K. Foster, G. Zeltzer, W. Ko and H. C. Manoharan. Quantum Phase Extraction in Isospectral Electronic Nanostructures. *Science*, Vol. 319 no. 5864 pp. 782–787, 8 February 2008.
6. C. H. Papadimitriou and M. Yannakakis. A note on succinct representations of graphs. *Inf. Control*, 71:181–185, December 1986.
7. L. G. Valiant. The complexity of computing the permanent. *Theoretical Computer Science*, 8(2):189 – 201, 1979.
8. I. Wegener. The complexity of Boolean functions. Wiley-Feubner Series in Computer Science. B. G. Teubner and John Wiley & Sons, Stuttgart; Chichester; New York, 1987.