The Traveling Beams
Optical Solutions for Bounded NP-Complete Problems

Shlomi Dolev*, Hen Fitoussi*

Abstract

Architectures for optical processors designed to solve bounded instances of NP-Complete problems are suggested. One approach mimics the traveling salesman by traveling beams that simultaneously examine the different possible paths. The other approach uses a pre-processing stage in which $O(n^2)$ masks are constructed, each representing a different edge in the graph. The choice and combination of the appropriate (small) subset of these masks yields the solution. The solution is rejected in cases where the combination of these masks totally blocks the light and accepted otherwise.

We present detailed designs for basic primitives of the optical processor. We propose designs for solving Hamiltonian path, Traveling Salesman, Clique, Independent Set, Vertex Cover, Partition, 3-SAT, and 3D-matching.

Keywords: Optical computing, NP-Complete problems

1 Introduction

The basic element used for computing is a switching element [6]. One such basic element in the scope of electronic circuitry is the transistor that is used to implement basic logic gates, such as logical *and* and logical *or*. The technology today seeks multi-core solutions in order to cope with the clock frequency limitations of VLSI technology. Namely, to implement on a single chip parallel/distributed system where a bus is used for communication among the processing units. Therefore, the communication overhead associated with distributed/parallel processing would be reduced dramatically. One may take the multi-core technology to the extreme - having a very large number of cores that are incorporated into the processing by sending signals over high-speed buses, maybe using optical/laser communication instead of traditional buses [9].

Optical communication maybe chosen due to the free space transmission capabilities of laser beams or the need to transmit signals from/to the processing unit through fiber optic channels. In such cases one may try to avoid the optical to digital conversion (and the digital to optical conversion) and use optical switches for computing. The straightforward solution

*Department of Computer Science, Ben-Gurion University of the Negev, Israel, {dolev,henf}@cs.bgu.ac.il.
Partially supported by the Lynn and William Frankel Center for Computer Science and the Rita Altura Trust Chair in Computer Science.

is to implement optical logic gates, such as logical *and* gate and logical *or* gate, a design that directly maps the current VLSI design to an all optical processor [3, 8].

On the other hand, electronic computers are not structured as mechanical computers, such as the Babbage machine [5], and it is possible that optical computers should be designed differently as well. In fact, some success in using many beams in free space for computing has been recently reported [7]. The design of [7] is based on parallel optical multiplication. The use of similar multiplication devices to solve bounded NP-Complete problems is suggested in [11]. Still, use of the fact that beams propagate in three dimensions is limited in the multiplication architectures of [7, 11] as the propagation of beams is in approximately the same direction. The beam traversal time is not used in the architectures that use Multiplication; we propose to use the time dimension as well.

The seminal work of [10] demonstrates the mapping between beam propagation and the computation of the deterministic Turing machine. In [1] use of a mapping similar to the non-deterministic Turing machine by (amplifying and) splitting beams is suggested. The mapping can be viewed as a theoretical existence proof for a solution, rather than an efficient solution. Knowing that a solution for a (bounded) NP-Complete problem instance exists, we seek for the most efficient solution in terms of the number of beams used, the number of optical elements (or location in space used to represent a computation state), the energy needed in terms of the maximum number of beams that should be split from a single source beam (fan-out), and the number of locations a beam needs to visit (and possibly split) from its creation until its final detection of arrival. We extend the design for the TSP primitive suggested in [1] and present a design for all (six) basic NP-Complete problems listed in [4]: Hamiltonian Path, Clique, Vertex Cover, Partition, 3-SAT, and 3D-matching. Note that polynomial reduction between NP-complete Problems is not used here, since we are concerned with a constant blowup in the instance size. Our goal is to solve the largest possible instance for each of these basic problems.

There are two main approaches used for the traveling beam Architecture; the first is a mapping of the graph nodes to physical locations in space and propagation of beams according to the edges of the input graph instance. The second approach propagates beams along a computation tree such that the leaves represent all possible solutions and the delay in propagation from the root to each leaf corresponds to the "value" of the specific combination.

We also present a totally different architecture called the *coordinated holes in mask-made-blackbox*. In this architecture a set of masks with "holes" are chosen from $n^2 - n$ pre-computed masks, according to the input instance. A solution exists only if the combined masks do not block all beams.

Paper organization. The next section describes the settings used for our designs. Section 3 details the design for the six basic NP-Complete problems. In particular, we extend the discussion on solving the Hamiltonian-path to gain some intuitive insight that is later used in describing the solutions for the next five problems. In Section 4 we present a totally different architecture for solving the Hamiltonian-path. Section 5 concludes the paper.

2 Settings of the Optical Computing Device

The optical micro-processor simulates a non-deterministic Turing machine. In a deterministic Turing machine, the next configuration is uniquely defined by applying the transition function δ to the current configuration (the transition function δ defines for each configuration the next configuration it yields in the computation). In a non-deterministic Turing machine the transition relation δ is a set of pairs of configurations such that $(c_1, c_2) \in \delta$ if and only if a configuration c_1 yields configuration c_2, meaning it is possible to go from the first configuration to a number of different configurations in a single move.

We will think of the non-deterministic computation as a directed graph. In this graph, each configuration is represented by a node, a directed edge connects two nodes v_1 and v_2 if it is possible to go from the configuration v_1 represents to the configuration v_2 represents in a single move. The graph constructed is a tree, in which the initial configuration is the root, and all the final configurations are the leaves.

In the optical micro-processor, each configuration corresponds to a 3D *location*. Laser or other beam creator (i.e., electronic beam) is used as a source at the initial configuration. The transition from one configuration to all the following configurations simultaneously is simulated by splitting the light at each *location* and propagating it in parallel to all the following configurations, as determined by the transition relation δ. According to the input, we may use barriers to block light between two configurations so that the transition between them is not allowed, or we may use an arrangement similar to an almost parallel arrangement of mirrors that will delay the transition of light between two configurations.

The constructed optical micro-processor enables us to use the parallel qualities of light to explore all possible paths of computations simultaneously. Light at a certain *location* indicates the feasibility of the correlated computation, while the absence of light indicates that the computation is not feasible. We will (possibly use a prism to direct all outputs to a single *location*, and) use light detectors at the leaves *locations*, in order to check whether and when light arrived at one of the final accepting configurations. According to this information, we will decide if the output is *accepted*, or the output is *rejected*.

In the sequel, we will use the term *column* to describe a group of *locations* that share the same (x,y) coordinates in space. These *locations* usually represent configurations that share a common attribute in the computation.

In the following section we describe the architectures designed to solve the basic NP-Complete problems that are presented in [4]. The design (of the Hamiltonian-path architecture) is based on [1], we add an analysis of the *fan-in*, *fan-out*, *efficiency factor*, where the *efficiency factor* is defined as the number of *locations* that light may reach divided by the number of possible solutions, and *depth*, where the *depth* is the maximal number of *locations* a beam traverses from the source to the detectors.

3 Architectures for Basic NP-Complete Problems

3.1 Hamiltonian-path

Given as input a directed graph $G = (V, E)$, the objective is to determine whether G contains a Hamiltonian-path.

Definition 3.1. *A path is a sequence of vertices such that from each vertex there is an edge to the next vertex in the sequence. A* Hamiltonian-path *is a path which visits each vertex of the graph exactly once.*

The architecture. The configurations in this architecture represent different paths of length zero to n, where n is the number of vertices in the graph. The initial configuration is a path of length zero, the final configurations are all possible paths of length n.
The arrangement of the configurations in space. The *locations* are ordered in n columns, $c_1, \ldots, c_n$, one column per each vertex. *Location* in column c_i represents a path that ends in vertex v_i. In addition to the arrangement in n columns, the *locations* are arranged in n levels, $level_1, \ldots, level_n$. Configurations in $level_i$ represent paths of length i. *Locations* in column c_j and level $level_i$ correspond to all different paths of length i which end in vertex v_j.

The following procedure is used to determine which path corresponds to a *location* in column c_j level $level_i$:

- As stated before, the level of the *location* determines the length of the path, the column of the *location* determines the last vertex in the path. This *location* represents a path of size i, which ends in vertex v_j.
- In order to determine the previous vertex in the path, we divide the array of *locations* in this column and level into n, and check in which part (sub-array) our *location* appears. A *location* which appears in part k indicates that the previous vertex is v_k.
- We will repeat the former action $i - 1$ times, each time with the sub-array we obtained earlier, until we will determine the entire path.

The column of vertex v_1 in a graph with three vertices is shown in Figure 1. The *locations* in this column are ordered in three levels – $level_1$, $level_2$, and $level_3$. We will demonstrate how to determine which path corresponds to the *location* marked by the dot. The *location* appears in $level_3$ in the column of v_1. Therefore the path is a path of length three which ends in vertex v_1. We divided $level_3$ into three sub-arrays; the *location* appears in the second sub-array, therefore the previous vertex in the path is v_2. We continue and divide the second sub-array into three sub-arrays. The *location* appears in the third sub-array, meaning the previous vertex is v_3 and the path represented by this *location* is $(v_3 \rightarrow v_2 \rightarrow v_1)$.

We would like to block the light from reaching *locations* which represent paths that contain a repetition of a vertex. In order to do so, we use a *mask*. We view a mask as a screen (e.g., transparency) of material carrying patterns that are either transparent or opaque to the wavelengths used; the mask is created *a priori* regardless of the input graph. The mask will

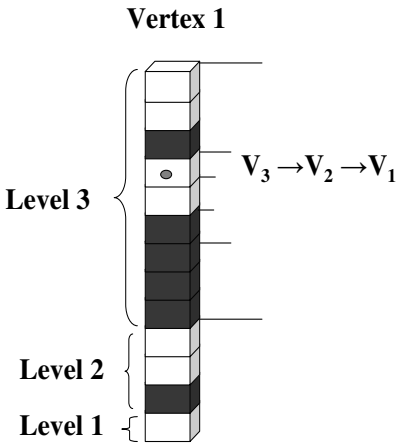


Figure 1: Illustration of the Hamiltonian-path column.

allow further propagation of light to *locations* that represent a feasible path, and will block the light from reaching *locations* which represent paths that contain a repetition of a vertex.

The algorithm of creating the mask is presented in Figure 2. It is an iterative process which occurs level after level. We copy the mask of the previous level n times, then we divide the array of *locations* (in this level) into n, and blacken the i part. The creation of the mask can be done by n^3 iterated processes of optical copying in a way similar to the process suggested in [11].

Procedure $CreateMask(c_j)$:
1. **for** $level_i = level_1$ to $level_n$ **do**
2. **if** $level_i = level_1$
3. **then** the mask is transparent
4. **else**
5. **for** $k = 1$ to n **do**
6. copy the mask of $level_{i-1}$ from column c_j
7. blacken part j of the mask of $level_i$

Figure 2: Creating the mask.

The masked column of vertex v_1 is shown in Figure 1. *Locations* colored in black indicate that the mask is opaque to the wavelength used and the light is blocked. *Locations* colored in white indicate that the mask is transparent. According to the algorithm, $level_1$ is transparent. The second level is created by copying the transparent mask of the first level three times, then dividing $level_2$ into three, and blackening the first part. The creation of the third level is done by copying the mask of the second level three times, then dividing $level_3$ into three, and blackening the first part.

To create a mask for a certain level we have to copy the former level n times; this takes n iterations. There are n columns, n levels in each column, so the whole process of creating the mask takes n^3 iterations.

The transition relation. At the initial configuration the beam of light is split into n beams which propagate simultaneously to $level_1$ in n different columns. In this way we get configura-

tions that represent n different paths of length one (the configuration in column c_i represents the path (v_i)).

In all *locations* (except the initial one) the beam of light is amplified and split into $(n-1)$ beams. These beams propagate to $(n-1)$ configurations that are located in the other $(n-1)$ columns. Light propagates from a configuration in $level_i$ to $(n-1)$ configurations in $level_{i+1}$. Configuration located in column c_j will propagate light to part j of the next level. The yielding configurations represent the extension of the path (represented by the first *location*) with the vertex each column represents. For example, in the *location* that represents the path (v_1), the beam of light is split into $(n-1)$ beams which propagate simultaneously to $(n-1)$ *locations* in $level_2$ in columns $c_2, \ldots, c_n$; in this way we get configurations that represent $(n-1)$ different paths of length two (for example, the configuration in column c_i represents the path $(v_1 \rightarrow v_i)$).

The transition relation is determined according to the input graph G. If the edge $(v_i, v_j) \notin G$ then the transition between a configuration in column c_i to a configuration in column c_j is not legal and should be blocked. In order to block light between configurations we will use $n(n-1)$ barriers, one barrier between every possible pair of columns. If $(v_i, v_j) \notin G$ then a barrier will block the light that propagates from column c_i to column c_j.

Detectors in $level_n$ will indicate if a beam arrived to a *location* l in $level_n$ in one of the columns. If the detector sensed a beam then there exists a path of length n with no vertex repetitions, and the transition between any two sequential vertices in the path represented by the *location* l is allowed. Meaning, there exists a Hamiltonian-path in the graph and the output is *accepted*, otherwise the output is *rejected*.

The Hamiltonian-path architecture for a graph with three vertices is illustrated in Figure 3. Vertices v_1 and v_2 are not connected, thus a barrier blocks the light between column c_1 and column c_2. The mask is transparent in *locations* colored in white, and opaque in *locations* colored in black. Light propagates from the initial configuration to $level_1$ in all three columns. To simplify the illustration, we show further propagation from column c_1 only. In column c_1 the beam is split into two beams. The beam directed to column c_2 is blocked by the barrier, the other beam arrives at $level_2$ in column c_3, then the beam is split into two beams which propagate simultaneously to $level_3$ in columns c_1 and c_3. The *location* in column c_1 represents the path $(v_1 \rightarrow v_3 \rightarrow v_1)$ which is not feasible; the mask blocks the light from reaching this *location*. The *location* in column c_2 represents the path $(v_1 \rightarrow v_3 \rightarrow v_2)$ which is a Hamiltonian-path, therefore the output is *accepted*.

Lemma 1. *In the Hamiltonian-path architecture:*

- *max(fan-out)* $= n$.
- *max(fan-in) = one.*
- *efficiency factor* $= n$.
- *depth* $= n$.

Proof: The number of outgoing beams at the initial *location* is n. At any other *location* the number of outgoing beams is $(n-1)$. Thus, the fan-out is n.

The number of incoming beams at each *location* is either zero (if the light was blocked or if it is the initial *location*) or one. Thus, the fan-in is one.

There are $n!$ possible Hamiltonian-paths, thus the number of possible solutions is $n!$. For didactic purposes, the architecture described contains *locations* which light can not reach. This can be easily modified in a way that will not include unreachable *locations*. The *locations* which light can reach correspond to all feasible paths of length one to n. There are $n \cdot (n-1) \cdot (n-2) \cdots (n-k)$ feasible paths of length k; hence, the number of all reachable *locations* is $n + n(n-1) + \cdots + n! \leq (n+1)!$. Therefore, the efficiency factor is less than $\frac{(n+1)!}{n!} = n$.

Beams in this architecture propagate from a configuration which corresponds to a path of length i to a configuration corresponding to a path of length $i+1$. The initial *location* represents a path of length zero, where the final *locations* represent paths of length n. Therefore, the beam traverses n *locations* before reaching a final configuration, and the depth is n. ■

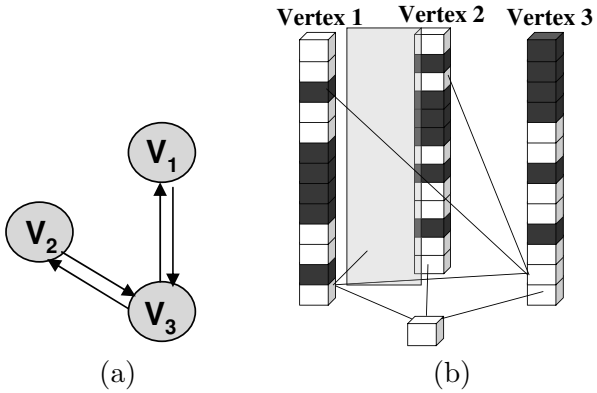


Figure 3: Illustration of the Hamiltonian-path architecture.

Obtaining the Traveling Salesman Problem (TSP) from Hamiltonian-path. In the TSP problem we have to determine whether there exists a Hamiltonian-path with distance smaller than d. The same arrangement may be used for solving the TSP problem by introducing appropriate delays of beams traversing from one column to the other. Meaning, if the distance between v_i and v_j is a, then a beam traversing from column c_i to column c_j will be delayed by a time units. Detectors will be used in $level_n$ of all the columns to check if there exists a *location* to which a beam of light arrived within d time units. If such a *location* exists, the output is *accepted*, otherwise the output is *rejected*. Alternatively, the first beam that arrives at $level_n$ defines the minimal distance. In case no beam arrives within $n\cdot$ *max-edge-delay* (where max-edge-delay is proportional to max-edge-weight), then there is no feasible path.

3.2 Clique

Given as input an undirected graph $G = (V, E)$, and an integer k, the objective is to determine whether G contains a clique of size k.

Definition 3.2. *A* clique *is a set of vertices $S \subseteq V$, such that $\forall v_1, v_2 \in S$, $(v_1, v_2) \in E$.*

The architecture. The configurations in this architecture represent different subsets of vertices of size zero to n, where n is the number of vertices in the graph. The initial configuration is the empty set, the final configurations are all possible subsets.

The architecture is built out of n *vertex architectures*, and an additional column, the AND column. The vertex architecture of v_i will indicate for each set S, such that $v_i \in S$, whether all the other vertices in S are connected to v_i. A set S is a clique if and only if $\forall v_i \in S$ all the other vertices in S are connected to v_i. The AND column will indicate whether the set is a clique.

The arrangement of the configurations in space.

Vertex architecture. The configurations in the vertex architecture are ordered as a tree, with $n+1$ columns, where configurations in depth i represent all possible subsets of $v_1 \dots v_i$. The left half of the configurations in depth i correspond to subsets S such that $v_i \in S$, where the right half of the configurations in depth i correspond to subsets which do not contain v_i.

The propagation of beams between configurations is done in the following manner. In each *location* the beam is amplified and split into two beams. These beams propagate to two configurations that are located in the next column. Meaning, beam propagates from a configuration in depth i which corresponds to the set S to two configurations in depth $i+1$, the transition to the right configuration represents the set S, and the transition to the left configuration represents the set $\{v_{i+1}\} \cup S$.

For example, in the *location* that represents the initial configuration, meaning the empty set, the beam is split into two beams of light which propagate simultaneously to two *locations* in the first column. The right configuration corresponds to the empty set, the left configuration corresponds to the set $\{v_1\}$.

The transition relation. In order to estimate the size of the sets, we will delay the light beams that pass through certain *locations* by d time units. The delay is done in any transition to the right, thus light will reach a configuration in the final column, which corresponds to a set S of size i, after $d \cdot (n-i)$ time units (a delay of d time units per each vertex $v_i \notin S$). In addition, we will use barriers to block the light from reaching configurations corresponding to sets which are not a clique.

In each vertex architecture v_i: If $(v_i, v_j) \notin E$, we will block the light from reaching configurations that correspond to sets that contain v_j, we will place the barriers before the left half of *locations* of depth j. In addition, we want to block the light from reaching *locations* corresponding to sets that do not contain v_i. In order to achieve this goal we will place a barrier before the right half of the *locations* in depth i. In this way, light will reach a leaf configuration corresponding to a set S in the vertex architecture of v_i, only if $v_i \in S$ and all the other vertices in S are connected to v_i.

We will number the leaf *locations* and the *locations* in the AND column as $l_1 \dots l_{2^n}$. Light propagates from a leaf *location* l_j in the vertex architecture to *location* l_j in the AND column.

In the AND column we will use threshold, just like choosing photographic film sensitivity, to block the light in *locations* where less than k beams of light arrived. In this way, sets which are not clique are blocked. A lens will concentrate the outgoing beams of the AND column to a single detector that will indicate whether a beam arrived after $d \cdot (n-k-0.5)$ time units

(the sensor is activated after $d \cdot (n - k - 0.5)$ time units to ignore sets with size smaller than k; light from such sets will arrive at the detector after $d \cdot (n - k)$ time units).

If the detector sensed a beam after $d \cdot (n - k - 0.5)$ time units, then the output is *accepted*, otherwise the output is *rejected*.

The clique architecture is illustrated in Figure 4. *Locations* colored in gray delay the light, *locations* colored in black block the light. Threshold on the number of incoming beams to *locations* in the AND column allow the light to propagate to the detector only if all k incoming beams arrive.

Determining which set S corresponds to a configuration is done by traversing the tree. A turn to the right in depth i in the path from the root to this leaf indicates that $v_{i+1} \notin S$; a left turn indicates that $v_{i+1} \in S$.

Lemma 2. *In the clique architecture:*

- *max(fan-out) = two.*
- *max(fan-in) = n.*
- *efficiency factor = $2n + 1$.*
- *depth = $n + 1$.*

Proof: In each *location*, in the vertex architecture the beam of light is split into two, except for the leaf configurations, where there is one outgoing beam. Thus, the fan-out is two.

Only one beam of light reaches each *location* in the vertex architecture. To any *location* in the AND column there are n possible incoming beams, therefore the fan-in is n.

If k is unknown every subset of V is a possible solution. Thus, the number of possible solutions is 2^n. The AND column contains 2^n configurations (one configuration per each possible subset of $v_1 \dots v_n$). The number of *locations* in a vertex architecture is $\sum_{i=0}^{n} 2^i = 2^{n+1}$. Thus, the number of *locations* needed in this architecture is $n \cdot 2^{n+1} + 2^n = (2n + 1)2^n$. Therefore, the efficiency factor is $\frac{(2n+1)2^n}{2^n} = 2n + 1$.

Beams in the vertex architecture propagate from a configuration in depth i to a configuration in depth $i + 1$; the maximal depth is n. Thus, a beam traverses n *locations* in the vertex architecture. From the final column in the vertex architecture, the beam traverses to the AND column, and from there it passes the detector. Therefore, the beam traverses $n + 1$ *locations* before reaching the detector, and the depth is $n + 1$.

■

Obtaining independent-set from clique. In the independent-set problem we need to determine whether there exists a set S of size k, such that $\forall v_1, v_2 \in S$, $(v_1, v_2) \notin E$.

We can use a similar architecture to the one described for the clique problem. As in the previous architecture, we will delay the light in any transition to the right and use barriers to block light, but here we will block the light between adjacent vertices. In each vertex architecture v_i: If $(v_i, v_j) \in E$, we will block the light from reaching configurations that correspond to sets that contain v_j, we will place the barriers before the left half of *locations* in depth j. In addition, we want to block the light from reaching *locations* corresponding to sets that do not contain v_i. In order to achieve this goal, we will place a barrier before the right

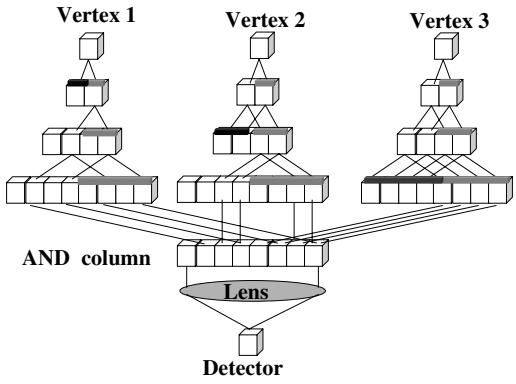


Figure 4: Illustration of the clique architecture.

half of the *locations* in depth i. In this way, light will reach a leaf configuration corresponds to a set S in the vertex architecture of v_i, only if $v_i \in S$ and all the other vertices in S are not connected to v_i.

In the AND column we will use threshold in order to block the light in *locations* where less than k beams of light arrived.

If the detector sensed a beam after $d \cdot (n - k - 0.5)$ time units, then the output is *accepted*, otherwise the output is *rejected*.

3.3 Vertex cover

Given as input an undirected graph $G = (V, E)$, and an integer k, the objective is to determine whether G contains a vertex cover of size k.

Definition 3.3. *A* vertex cover *is a set of vertices $S \subseteq V$, such that $\forall v_i \notin S$, $\exists v_j \in S$ for which $(v_i, v_j) \in E$.*

In order to solve this problem we will use the following observation:

Observation 3.1. *There exists a vertex cover of size k in graph G, if and only if there exists a clique of size $n - k$ in the complementary graph $\overline{G}$.*

Given a graph G and an integer k, we will use the clique architecture with $\overline{G}$ and $n - k$ as an input. This reduction does not increase the size of the input.

The next lemma is identical to Lemma 2, which was proven in subsection 3.2.

Lemma 3. *In the vertex cover architecture:*

- *max(fan-out) = two.*
- *max(fan-in) = n.*
- *efficiency factor = $2n + 1$.*
- *depth = $n + 1$.*

3.4 Partition

Given a set of integers $S = \{a_1, \ldots, a_n\}$, the objective is to determine whether there exists a subset $S_1 \subset S$ such that $\sum_{a_i \in S_1} a_i = \sum_{a_i \in S \setminus S_1} a_i$.

The architecture. The configurations in this architecture represent different subsets of length zero to n, where n is the size of the set S. The initial configuration is the empty set, the final configurations are all possible subsets.

The arrangement of the configurations in space. The configurations are ordered as a tree, where configurations in depth i represent all possible subsets of $a_1, \ldots, a_i$. The right half of the configurations in depth i correspond to sets that do not contain a_i, where the left half of the configurations correspond to sets that contain a_i.

The transition relation. In each *location* the beam of light is amplified and split into two beams. These beams propagate into two configurations that are located in the next column. The transition to *locations* that represent sets S_1 such that $a_i \in S_1$ delays the light by a_i time units. Meaning, light propagates from a configuration in depth i which corresponds to the set S_i to two configurations in depth $i+1$. The transition to the right configuration represents the set S_i. The transition to the left configuration represents the set $\{a_{i+1}\} \cup S_i$, light is delayed by a_{i+1} time units in this transition.

For example, in the *location* that represents the initial configuration, meaning the empty set, the beam is split into two beams of light which propagate simultaneously to two *locations* in the first column. The right configuration corresponds to the empty set (and does not delay the light), the left configuration corresponds to the set $\{a_1\}$ and the transition to this configuration delays the light by a_1 time units.

The partition architecture for a set of three integers $\{a_1, a_2, a_3\}$ is illustrated in Figure 5. The transition to a configuration in depth i which is colored in gray delays the light in a_i time units. Light reaches the final configuration which correlates to the set $\{a_1, a_2, a_3\}$ after $a_1 + a_2 + a_3$ time units.

A partition of the set S exists if and only if a beam of light reached one of the leaf configurations after $\frac{\sum_{a_i \in S} a_i}{2}$ time units.

We will use a lens to concentrate the light that goes from the leaf configurations to a single detector. The detector will be used to identify arriving beam after $\frac{\sum_{a_i \in S} a_i}{2}$ time units. If a beam arrived after $\frac{\sum_{a_i \in S} a_i}{2}$ time units then the output is *accepted*, otherwise the output is *rejected*.

Determining which subset corresponds to a leaf configuration is done by traversing the tree. A turn to the right in depth i in the path from the root to this leaf indicates that a_{i+1} is in the subset, and a left turn indicates that a_{i+1} is not in the subset.

Lemma 4. *In the partition architecture:*

- *max(fan-out) = two.*
- *max(fan-in) = one.*
- *efficiency factor = two.*
- *depth = n.*

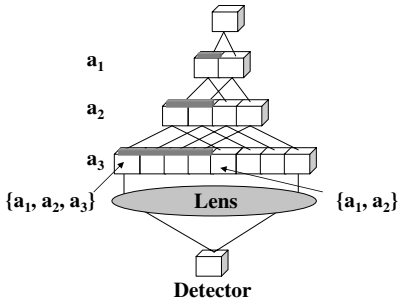


Figure 5: Illustration of the partition architecture.

Proof: In each *location* the beam of light is split into two, thus the fan-out is two. Only one beam of light reaches each *location*, therefore the fan-in is one.

The solutions are all possible subsets, meaning there are 2^n possible solutions. The *locations* in depth i correspond to all possible subsets of $a_1 \dots a_i$, therefore the number of *locations* in depth i is 2^i, and the total number of *locations* is $\sum_{i=0}^{n} 2^i = 2^{n+1}$. Therefore, the efficiency factor is $\frac{2^{n+1}}{2^n} = 2$.

A beam propagates from a configuration in depth i to a configuration in depth $i+1$. The final configurations are in depth n. Therefore, the depth of this architecture is n. ■

3.5 3-SAT

Given a 3-CNF formula φ as input, the objective is to determine whether φ is satisfiable.

Definition 3.4. *Formula φ is* satisfiable *if and only if there exists an assignment for the formula's variables under which φ evaluates to* true*.*

The architecture. Configurations in this architecture represent different partial assignments of size zero to n, where n is the number of variables. A partial assignment of size i assigns values to the variables $x_1 \dots x_i$. The initial configuration is the empty assignment, the final configurations are all possible assignments of $x_1 \dots x_n$.

The architecture is built out of l *clause architectures*, where l is the number of clauses and an additional column, the AND column. In order to determine whether φ is satisfiable, we will check separately whether the assignment satisfies each of the formula's clauses (using the clause architecture). Assignment satisfies φ if and only if it satisfies all φ's clauses; the AND column will indicate whether the assignment satisfies all clauses.

The arrangement of the configurations in space.

Clause architecture. As demonstrated in Figure 6, the configurations in the clause architecture are ordered as a tree, where configurations in depth i represent all possible assignments of $x_1 \dots x_i$. The right half of the configurations in depth i correspond to assignments where

x_i evaluates to `true`. The left half of the configurations in depth i correspond to assignments where x_i evaluates to `false`.

The propagation of light between configurations is done in the following manner. In each *location* the beam of light is amplified and split into two beams. These beams propagate to two configurations in the column of the next variable. The yielding configurations represent the extension of the assignment represented by the first *location*. In the right transition the value of the next variable is `true`, and in the left transition it is `false`. For example, in the *location* that represents the initial configuration, meaning the empty assignment, the beam of light is split into two beams which propagate simultaneously to two *locations* in the first column. In this way we get the right configuration in the first column that represents the partial assignment of size one $x_1 =$ `true`, and the left configuration in the first column for the partial assignment $x_1 =$ `false`.

In the sequel we present two methods for implementing the transition relation: The first method uses a reduction in light intensity, the second method uses the delay of light.

The transition relation (with a mask). In this implementation we use a gray scale mask to decrease the intensity of light that passes through certain *locations* by half.

In each clause architecture c: If $x_i \in c$ then the left half of column i is masked using the gray mask; if $\overline{x_i} \in c$ then the right half of column i is masked.

In the leaf configurations, we will use threshold in order to block the light with intensity less than or equal to $\frac{W}{2^3}$, where W is the initial light intensity. The idea is to block the assignments which do not satisfy the clause; in such assignments all three literals are assigned `false`, thus light intensity will be decreased three times. In case the threshold did not block the light, it will propagate to the AND column with correlation to the *location*. We will number the leaf *locations* and the *locations* in the AND column as $l_1 \dots l_{2^n}$. Light propagates from a leaf *location* l_j to *location* l_j in the AND column.

In the AND column we will use threshold to block the light in *locations* where less than l beams of light arrive. In this way assignments where at least one clause evaluates to `false` are blocked. A lens will be used to concentrate all beams from the AND column to one *location* where a detector will be used. If a beam arrives at the detector then the output is *accepted*, otherwise the output is *rejected*.

The transition relation (with delay). In this architecture we will delay the light that goes through certain *locations* by d time units.

In each clause architecture c: If $x_i \in c$ then light that goes through the left half of column i is delayed for d time units; if $\overline{x_i} \in c$ the light that goes through the right half of column i is delayed by d time units. The idea is to delay the light by $3d$ time units in assignments where all three literals are assigned to `false`.

We will number the leaf *locations* and the *locations* in the AND column as $l_1 \dots l_{2^n}$. Light propagates from leaf *location* l_j to *location* l_j in the AND column.

In the AND column we will use threshold to block the light in *locations* to which less than l beams of light arrived, where l is the number of clauses in the formula. The idea is to block the assignments where at least one clause evaluates to `false`.

A lens will be used to concentrate all beams from the AND column to one *location* where a

detector will indicate whether a beam arrived after $2.5d$ time units. If there exists a *location* in the AND column where all l beams arrived in less than $3d$ time units, then there exists a satisfying assignment.

If a beam arrived at the detector after $2.5d$ time units, then the output is *accepted*, otherwise the output is *rejected*.

The 3-SAT architecture for the formula $(x_1 \vee x_2 \vee \overline{x_3}) \wedge (\overline{x_1} \vee \overline{x_2} \vee x_3)$ is illustrated in Figure 6. A configuration colored in gray delays the light by d time units or decreases the light intensity by half.

Determining which assignment corresponds to a certain configuration is done by traversing the tree. A turn to the right in depth i of the path from the root to this leaf indicates that the value of x_{i+1} is `true`, and a left turn indicates that it is `false`.

Lemma 5. *In the 3-SAT architecture:*

- *max(fan-out) = two.*
- *max(fan-in) = l.*
- *efficiency factor = $2l + 1$.*
- *depth = $n + 1$.*

Proof: In each non-leaf *location* in the clause architecture, the beam of light is split into two beams. Each leaf configuration has one outgoing beam. Thus the fan-out is two.

Only one beam of light reaches each *location* in the clause architecture. To any *location* in the AND column there are l possible incoming beans, therefore the fan-in is l.

The possible solutions are all assignments on $x_1 \cdots x_n$, meaning there are 2^n possible solutions. The number of *locations* in depth i of a clause architecture is 2^i, meaning, the total number of *locations* for one clause is $\sum_{i=0}^{n} 2^i = 2^{n+1}$. The AND column contains 2^n configurations (a configuration per each possible assignment on the variables $x_1 \ldots x_n$). Thus, the number of *locations* needed in this architecture is $l \cdot 2^{n+1} + 2^n = (2l + 1)2^n$. The efficiency factor is $\frac{(2l+1)2^n}{2^n} = 2l + 1$.

Beams in the clause architecture propagate from a configuration in depth i (correlating to x_i) to a configuration in depth $i + 1$ (correlating to x_{i+1}). The final column correlates to x_n; therefore, a beam traverses n *locations* in the clause architecture. From the final column in the clause architecture, the beam traverses to the AND column, and from there it passes to a detector. Therefore, the beam traverses $n + 1$ *locations* before reaching a detector, and the depth is $n + 1$.

■

3.6 3D matching

This problem is a generalization of the bipartite matching problem. In the 3D-matching problem, the input is three sets B, G, and H (boys, girls, and homes), such that $|B| = |G| = |H| = n$, and a set of triples T, $T \subseteq B \times G \times H$. The objective is to determine whether there exists a set of n triples in T, such that each boy is matched to a different girl, and each couple has home of its own. We will refer to such an arrangement as a perfect match.

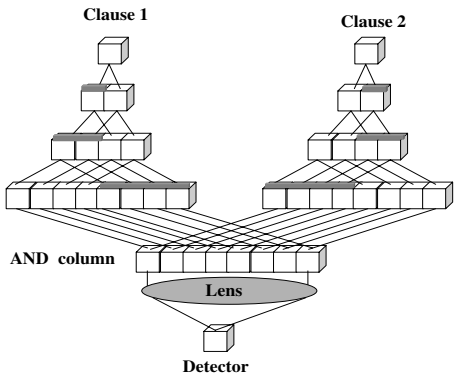


Figure 6: Illustration of the 3-SAT architecture.

Definition 3.5. *A* perfect match *is a set of triples $S \subseteq T$, such that $\forall o_i \in B \cup G \cup H$, o_i appears exactly in one triple in S.*

We will say that a triple t_i is *consistent* with triple t_j if and only if $\forall o_i \in t_i \Rightarrow o_i \notin t_j$. We will construct a graph where each triple is represented by a vertex. An edge connects v_i and v_j if and only if the triple v_i is consistent with triple v_j.

Observation 3.2. *There exists a perfect match if and only if there exists a clique of size n in the constructed graph.*

The architecture. We will use an architecture similar to the clique architecture. The configurations in the architecture for the 3D matching represent different subsets of triples of size zero to t, where t is the size of the set T. The initial configuration is the empty set, the final configurations are all possible subsets. The architecture is built of t triple architectures and an AND column.

The arrangement of the configurations in space. The configurations in the triple architecture are ordered as a tree, with $t+1$ columns, where configurations in depth i represent all possible subsets of $t_1 \dots t_i$. The left half of the configurations in depth i corresponds to subsets S such that $t_i \in S$, where the right half of configuration in depth i corresponds to subsets which do not include t_i.

The propagation of light between configurations is done in the following manner. In each *location* the beam of light is amplified and split into two beams. These beams propagate to two configurations that are located in the next column – meaning light propagates from a configuration in depth i corresponding to the set S to two configurations in depth $i+1$, the transition to the right configuration represents the set S, and the transition to the left configuration represents the set $\{t_{i+1}\} \cup S$.

For example, in the *location* that represents the initial configuration, meaning the empty set, the beam is split into beams of light which are propagated simultaneously to two *locations* in the first column. The right configuration corresponds to the empty set, the left configuration corresponds to the set $\{t_1\}$.

The transition relation. In order to estimate the size of the sets, we will delay the light that passes through certain *locations* by d time units. This delay is done before any transition to the right. Thus, light will reach a configuration in the final column, which correlates to a set of size i, after $d \cdot (t - i)$ time units.

In addition, we will use barriers to block the light from reaching configurations correlated to sets which are not a clique. In each triple architecture t_i: If t_i is not consistent with t_j, we will block the light from reaching configurations that correlate to sets that contain t_j and we will place the barriers before the left half *locations* of depth j. We will number the leaf *locations* and the *locations* in the AND column as $l_1 \dots l_{2^n}$. Light propagates from leaf *location* l_j to the *location* l_j in a in the AND column.

Threshold is used in the AND column to block the light in *locations* where less than n beams of light arrive within $d \cdot (t - n)$ time units. In this way, sets which are not cliques of size n are blocked.

A lens will be used to concentrate all beams from the AND column to one *location* where a detector will indicate whether there exists a *location* in the AND column where all l beams arrived, and therefore the beam passed the threshold.

If light was sensed by the detector within $d \cdot (t - n)$ time units, then the output is *accepted*, otherwise the output is *rejected*.

Determining which set S corresponds to a configuration is done by traversing the tree. A turn to the right in depth i in the path from the root to this leaf indicates that $t_{i+1} \notin S$, and a left turn indicates that $t_{i+1} \in S$.

The next lemma is identical to Lemma 2, which was proven in Subsection 3.2.

Lemma 6. *In the 3D-matching architecture:*

- *max(fan-out) = two.*
- *max(fan-in) = t.*
- *efficiency factor = $2t + 1$.*
- *depth = $t + 1$.*

4 Coordinated Holes in Masks-Made-Blackbox

In the following section we present a different architecture for an optical micro-processor. We will demonstrate how this optical processor is used to solve bounded instances of the Hamiltonian-path problem.

In [11] the authors present an iterative algorithm that produces a binary matrix that represents all possible Hamiltonian-paths using optical copying. Every column in the matrix correlates to a possible edge in the graph, every row in the matrix correlates to a possible Hamiltonian-path. Zero (opaque screen) in the $[i][j]$ entry of the matrix indicates that the edge e_j does not appear in path number i, where one (transparent screen) in the $[i][j]$ entry indicates that the edge e_j does appear in path number i. We will use a similar technique to produce a matrix where zero indicates the existence of an edge, and vise versa.

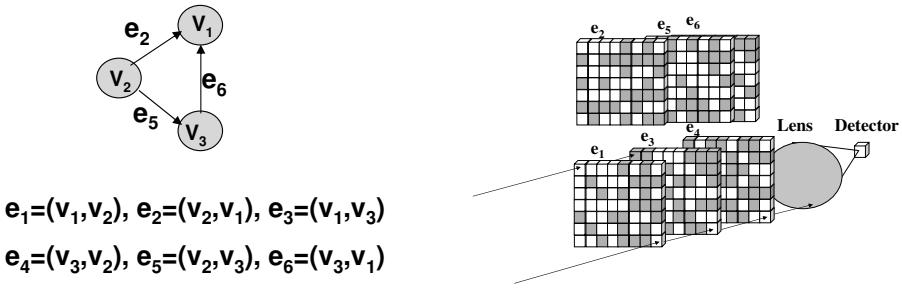
In our architecture we will use each column of the binary-matrix mask as a barrier. There are $n(n-1)$ possible barriers, a barrier per each possible directed edge. The barrier of edge

e_i blocks the light in paths p where $e_i \in p$, and allows the propagation of light in paths p such that $e_i \notin p$.

Given an input graph $G = (V, E)$, we will select a subset of the barriers. If $e_i \in E$ then we do not use the barrier of edge e_i. Otherwise, we will use the barrier of edge e_i which will block the light in paths p where $e_i \in p$.

The masks that are in use (according to the input) are ordered one behind the other in a way that the entry of the path p in one mask is placed behind the entry of the same path p in the other masks. Note that the masks can be in any 3D shape, for example, nested boxes or balls.

Light propagates from the light source, passes through all the masks, and if possible reaches a detector. A lens will be used to concentrate the light from the final configurations to a single detector. The detector indicates whether light passed all barriers and reached the final configuration. If so, it means that there exists a specific path where the barriers did not block the light from passing. Meaning, all edges in this path exist in the graph. If the detector sensed light then the graph contains a Hamiltonian-path and the output is *accepted*, otherwise the output is *rejected*.



(a) (b)

Figure 7: Illustration of the holes in masks-made-blackbox architecture.

The coordinated holes in masks-made-blackbox architecture for a graph with three vertices is illustrated in Figure 7. The edges e_1, e_3, and e_4 do not appear in the graph, thus the correlate barriers are used. To simplify the illustration, we show the propagation of two beams only. The upper left entry correlates to a path that contains the edge e_3, thus the mask of e_3 in this entry is opaque to the wave length and the beam is blocked and does not reach the detectors. The lower right entry correlates to a path that does not include edges e_1, e_3, and e_4, thus this entry is transparent in all three masks, and the beam reaches the detector, which indicates that the graph contains a Hamiltonian-path.

5 Concluding Remarks

The advance in optical communication and computing may well serve as a way to cope with the limitations VLSI technologies now face. We suggest ways to use the natural parallelism of wave propagation in space for solving inherent hard problems in the scope of sequential

or even parallel electronic computers. The existence of recent industrial attempts to produce optical processing devices (e.g., [7]) as well as the limited implementations in our laboratory (e.g., [11]) encourage us to believe that our new designs will be used in practice for solving combinatorial tasks, at least when there are real-time constraints. Some cryptographic usage of optical processors are suggested in [2, 12, 13].

We would like to remark that most of our designs solve in fact the #P version of the problem.

At last, we view our work as a beginning for (having fun in) further investigations on using beams of light and their location in free space for parallel computations.

Acknowledgments. We thank Stephan Messika for discussions in the first steps of this research, and Nati Shaked for fruitful discussions.

References

[1] S. Dolev and N. Yuval, "Optical implementation of bounded non-deterministic Turing machines", US Patent 7,130,093 B2, October 2006.

[2] S. Dolev, E. Korach and G. Uzan, "A Method for Encryption and Decryption of Messages", PCT Patent Application WO 2006/001006, January 2006.

[3] G. Feitelson, *Optical Computing: A Survey for Computer Scientists*, MIT Press, 1988.

[4] M. R. Garey and D. S. Johnson, *Computers and Intractability, a guide to the theory of NP-completeness*, W. H. Freeman and Company, San Francisco, 1979.

[5] A. Hyman, *Charles Babbage: Pioneer of the Computer*, Princeton University Press, 1982.

[6] Z. Kochavi, *Switching and finite automata theory*, McGraw-Hill, 1978.

[7] Lenslet LTD, `http://www.hpcwire.com/hpcwire/hpcwireWWW/03/1017/106185.html`.

[8] A.D. McAulay, *Optical computer architectures*, John Wiley, 1991.

[9] H. Rong, A. Liu, R. Jones, O. Cohen, D. Hak, R. Nicolaescu, A. Fang, and M. Paniccia, "An all-sillicon Raman laser", *Nature*, 433, 292-94, (20 Jan. 2005).

[10] J.H. Reif, D. Tygar, and A. Yoshida, "The Computability and Complexity of Optical Beam Tracing", *31st Annual IEEE Symposium on Foundations of Computer Science*, 1990, pp. 106-114. Also "The Computability and Complexity of Ray Tracing", *Discrete and Computational Geometry*, 11:265-287 (1994).

[11] N. T. Shaked, S. Messika, S. Dolev, and J. Rosen. "Optical Implementation of Combinatorial Processor", A poster in the *Bi-National (Israeli-Italian) Workshop on Optronics* (Il-It-Opt), November 2005. This also appears in N. Shaked, G. Simon, T. Tabib, S. Mesika, S. Dolev, J. Rosen, "Optical processor for solving the traveling salesman problem

(TSP), *Proc. of SPIE Symposium on Optics & Photonics the Optical Information Systems IV Conference*, Vol. 63110G-1, Aug. 2006, San Diego. To appear in N. T. Shaked, S. Messika, S. Dolev, and J. Rosen. "Optical Solution for Bounded NP-Complete Problems", *Journal of Applied Optics*, accepted, October 2006.

[12] A. Shamir, "Factoring Large Numbers with the TWINKLE device", Proc. CHES'99, LNCS 1717 2-12, Springer-Verlag, 1999.

[13] A. Shamir and E. Tromer, "Factoring Large Numbers with the TWIRL Device", proc. Crypto 2003, LNCS 2729, 1-26, Springer-Verlag.