# Optical solution for bounded NP-complete problems

Natan T. Shaked, Stephane Messika, Shlomi Dolev, and Joseph Rosen

We present a new optical method for solving bounded (input-length-restricted) NP-complete combinatorial problems. We have chosen to demonstrate the method with an NP-complete problem called the traveling salesman problem (TSP). The power of optics in this method is realized by using a fast matrix–vector multiplication between a binary matrix, representing all feasible TSP tours, and a gray-scale vector, representing the weights among the TSP cities. The multiplication is performed optically by using an optical correlator. To synthesize the initial binary matrix representing all feasible tours, an efficient algorithm is provided. Simulations and experimental results prove the validity of the new method. © 2007 Optical Society of America

*OCIS codes:* 200.0200, 070.4550, 070.4560, 070.2580.

## 1. Introduction

NP-complete combinatorial problems represent a group of problems that are solvable in polynomial time on a nondeterministic Turing machine and can be transferred from one problem to another by using reductions[1] (which means that if one of these problems can be solved within a certain time complexity, all of the other problems in this group can be solved within almost the same time complexity). In spite of many years of research, there is no known efficient (polynomial-time) solution to the problems belonging to this group on a (deterministic) Turing machine. A typical representative problem of this group is the traveling salesman problem (TSP). The TSP, which has a few common versions, is usually involved in finding the shortest tour of a salesman going through a certain number of cities. The distances (weights) between the cities are known. The main constraints of this problem are that the salesman has to go through all of the cities and visit each city only once. Many examples of TSP applications can be found in the literature.[2] However, much of the work on the TSP is not only motivated by its applications but also by the fact that the TSP provides a sound platform for studying the general schemes applied to a wide range of combinatorial optimization problems. The TSP as well as other NP-complete problems can be used as an efficient tool for testing the performance of computational systems, i.e., better computational systems can solve the TSP faster. Therefore, the TSP can be used as a reliable means for comparing the performances of optical computing versus conventional computing.

Solving the TSP by checking all the possible tours may be very expensive in terms of time complexity.[1] For that reason, several heuristic methods have shown ways to find the best TSP solution by skipping tours predicted to be less than the best one.[3] Other methods use approximations and prefer a relatively good solution obtained quickly rather than the best solution, which may not be obtained within a reasonable computation time.[4] Previous research works on optical TSP solvers mostly implemented well-known TSP solvers by using suitable optical systems. For example, in Ref. 5, a TSP solver based on a Kohonen neural network[4] was implemented optically. Using these approximation or heuristic methods, a relatively large number of TSP cities can be handled.

However, approximation methods do not always find the global optimal solution, whereas heuristic methods are able to solve the TSP for certain instances only. In fact, it can be shown[6] that for any heuristic method there are easy-to-find cases for which the TSP cannot be solved within a reasonable time. In addition, the computation time of the heuristic methods may be unexpected (depends on the TSP instance) and may be even longer than the time it takes to explore every possible tour in an exhaustive manner (due to unsuccessful attempts to em-

The authors are with Ben-Gurion University of the Negev, P.O. Box 653, Beer-Sheva 84105, Israel. N. T. Shaked (natis@ee.bgu.ac.il) and J. Rosen are with the Department of Electrical and Computer Engineering. S. Messika and S. Dolev are with the Department of Computer Science.

Received 15 March 2006; revised 10 September 2006; accepted 11 October 2006; posted 12 October 2006 (Doc. ID 68905); published 25 January 2007.

0003-6935/07/050711-14$15.00/0

ploy optimization techniques). Thus, in applications where deadlines must be met, such heuristics are not a good choice, and in fact, one may prefer to use the exhaustive search (exploring all tours) rather than using heuristics.

The current study introduces a new method of using optics to solve the TSP of a fully-connected graph by checking all of its feasible tours. The method is based on an optical processor, which performs a fast matrix–vector multiplication between a binary matrix (light and dark spot matrix), representing all TSP feasible tours, and a weight vector (gray-scale spot vector), representing the finite weights between the TSP cities. This system yields a vector, which represents the lengths of the TSP tours. The matrix–vector multiplication is performed optically by using an optical correlator.[7–9] The binary matrix, used for the matrix–vector multiplication, contains only the TSP feasible tours. This matrix is synthesized using a new iterative algorithm. This algorithm utilizes only a small number of loops and duplicates the relatively long vectors with no loops. In addition, the algorithm produces the binary matrix so that this matrix can be used to solve any TSP with the same number or fewer cities.

Note that the TSP used here is a bounded (input-length-restricted) TSP, and therefore the design proposed in this paper does not imply that there is a polynomial solution to an NP-complete problem; i.e., the design does not imply that P = NP.[1] However, the design is aimed at increasing the maximal number of cities for which a TSP solution can still be obtained within a reasonable and predefined time.

Also note that more NP-complete problems can be solved with the proposed method. For example, the Hamiltonian path problem[1] that indicates whether there is a path with a length smaller than infinity (which implies that part of the edges may be blocked or has an infinite weight) is an NP-complete problem on its own. The solution to this problem can be obtained using the proposed optical system by representing each unblocked edge by a numerical 1 (light) and each blocked edge by a numerical 0 (dark). Then, a certain light intensity obtained in the output of the optical system means that a Hamiltonian path exists. Using the same proposed method, solving the Hamiltonian path problem is easier to implement than the TSP since the weight vector is binary, and all that is needed in the output of the optical system is to decide whether there is a certain light intensity there. However, we have chosen to demonstrate the method on the TSP, in which a gray-scale weight vector is used, since it is a more general case from the point of view of the optical system although both cases are considered as NP-complete problems and solving one of these problems means (up to a polynomial-time reduction) solving the other one as well.

The rest of the paper is organized as follows: Section 2 presents definitions of the TSP versions considered in the paper. Section 3 explains the idea behind the new method, whereas the optical implementation of this method is explained in Section 4. Sections 5 and 6 present simulation and experimental results, respectively. Section 7 summarizes the entire paper, discusses certain conclusions, and suggests future research directions.

## 2. General Properties of the Traveling Salesman Problem

Figure 1 shows an example of a TSP containing seven cities. The cities are shown in this figure as circles. The successive numbers (indices) of the cities are written beside them. The number written on each edge connecting a pair of cities represents the weight (distance) between the two cities. The shortest tour in this case is shown in the figure by using bold lines. This version of the TSP is called the symmetric TSP since the weight between the $i$th and the $j$th cities is equal to the weight between the $j$th and the $i$th cities. In this case, the number of feasible tours is given by

$$K_{\text{symmetric}} = (N-1)!/2, \tag{1}$$

and the number of weights is given by

$$M_{\text{symmetric}} = N(N-1)/2. \tag{2}$$

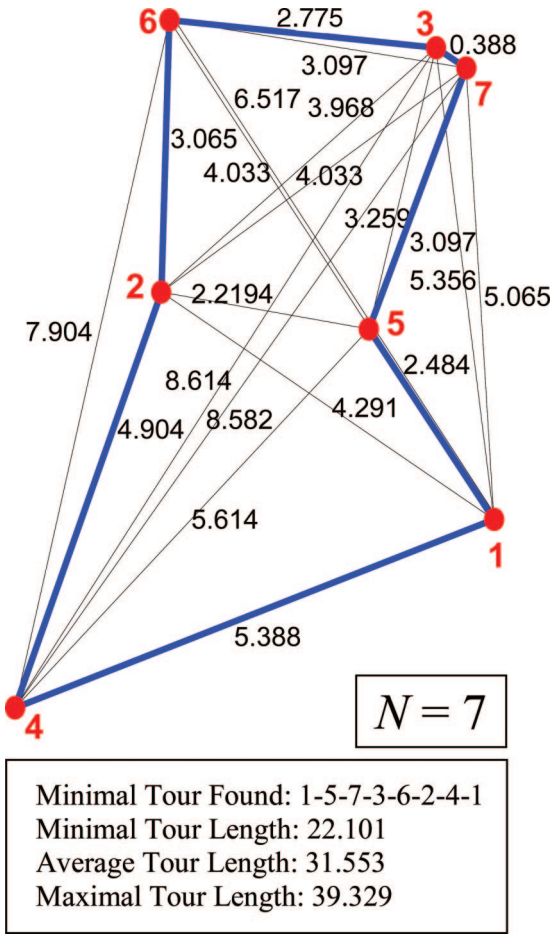



Fig. 1. (Color online) Example of a seven-city symmetric TSP. The best (shortest) tour is indicated by bold lines.

However, it is also possible to define an asymmetric version of the TSP. In such a version, the weight from the $i$th to the $j$th cities may be unequal to the weight from the $j$th to the $i$th cities.[2] In this case, the number of feasible tours is given by

$$K_{\text{asymmetric}} = (N-1)!, \tag{3}$$

and the number of weights is given by

$$M_{\text{asymmetric}} = N(N-1). \tag{4}$$

Based on Eqs. (1) and (2), it can be shown that the number of feasible tours for a 25-city TSP is the large number $3.1022 \times 10^{23}$ for a symmetric TSP and $6.2045 \times 10^{23}$ for an asymmetric TSP. Even assuming that a conventional computer is able to check $10^9$ tours per second,[10] it will take more than 9.8 million years to check all feasible tours for the symmetric 25-city TSP and twice as long for the asymmetric 25-city TSP. This is of course an unreasonable computation time.

## 3. Description of the Optical Traveling Salesman Problem Solver

This section presents the new approach for solving the TSP by optics. The power of optics in this approach is realized by using a fast and parallel matrix–vector multiplication. The current section explains the idea behind this approach rather than the chosen optical technique of the multiplication, discussed later in Section 4.

### A. Calculating the Tour Lengths by Using a Matrix–Vector Multiplication

According to the proposed TSP solver, a binary matrix **b** representing all feasible TSP tours is multiplied by a weight vector **w** representing the weights between the TSP cities. A 1 in the $i$th row and in the $j$th column of the binary matrix means that the $i$th tour length contains the $j$th weight of the weight vector. To demonstrate this, let us take, for example, the case of the symmetric four-city TSP. In this case ($N = 4$), there are [according to Eq. (2)] $M_{\text{symmetric}} = N(N-1)/2 = 4 \times 3/2 = 6$ weights, which are $w_{1,2}$, $w_{1,3}$, $w_{1,4}$, $w_{2,3}$, $w_{2,4}$, and $w_{3,4}$, and [according to Eq. (1)] $K_{\text{symmetric}} = (N-1)!/2 = 3!/2 = 3$ different tours, which are

$$\begin{aligned}
&\text{Tour 1:}\quad \text{City 1} \rightarrow \text{City 2} \rightarrow \text{City 3} \rightarrow \text{City 4} \rightarrow \text{City 1},\\
&\text{Tour 2:}\quad \text{City 1} \rightarrow \text{City 3} \rightarrow \text{City 4} \rightarrow \text{City 2} \rightarrow \text{City 1},\\
&\text{Tour 3:}\quad \text{City 1} \rightarrow \text{City 4} \rightarrow \text{City 2} \rightarrow \text{City 3} \rightarrow \text{City 1}.
\end{aligned} \tag{5}$$

The length $l_i$ ($i = 1, 2, 3$) of each of these tours is the sum of the weights coinciding with the edges of the tour:

$$\begin{aligned}
&\text{Tour 1:}\quad l_1 = w_{1,2} + w_{2,3} + w_{3,4} + w_{1,4},\\
&\text{Tour 2:}\quad l_2 = w_{1,3} + w_{3,4} + w_{2,4} + w_{1,2},\\
&\text{Tour 3:}\quad l_3 = w_{1,4} + w_{2,4} + w_{2,3} + w_{1,3}.
\end{aligned} \tag{6}$$

In this case ($N = 4$), the weight vector can be defined as follows:

$$\mathbf{w} = [w_{1,2},\ w_{1,3},\ w_{1,4},\ w_{2,3},\ w_{2,4},\ w_{3,4}]^T, \tag{7}$$

whereas the coinciding binary matrix, which has the dimensions of $[K_{\text{symmetric}}] \times [M_{\text{symmetric}}] = [3] \times [6]$ (three rows, six columns), is

$$\mathbf{b} = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 \end{bmatrix}. \tag{8}$$

The result of the multiplication between the binary matrix **b** [Eq. (8)] and the weight vector **w** [Eq. (7)] is a length vector **l** representing all the lengths of the feasible TSP tours:

$$\begin{aligned}
\mathbf{l} = \mathbf{b} \cdot \mathbf{w} &= \begin{bmatrix} w_{1,2} + 0 + w_{1,4} + w_{2,3} + 0 + w_{3,4} \\ w_{1,2} + w_{1,3} + 0 + 0 + w_{2,4} + w_{3,4} \\ 0 + w_{1,3} + w_{1,4} + w_{2,3} + w_{2,4} + 0 \end{bmatrix} \\
&= \begin{bmatrix} w_{1,2} + w_{1,4} + w_{2,3} + w_{3,4} \\ w_{1,2} + w_{1,3} + w_{2,4} + w_{3,4} \\ w_{1,3} + w_{1,4} + w_{2,3} + w_{2,4} \end{bmatrix} = \begin{bmatrix} l_1 \\ l_2 \\ l_3 \end{bmatrix}.
\end{aligned} \tag{9}$$

One can see that the final result of Eq. (9) is equivalent to Eq. (6). This means that by performing this matrix–vector multiplication, we carry out both the multiplication between the TSP weights and the binary matrix elements and the summation of the relevant weights in order to obtain each of the feasible TSP tour lengths. The smallest element in this column vector corresponds to the best (shortest) TSP tour. The same method can be implemented for any symmetric or asymmetric TSP.

### B. Synthesizing the Binary Matrix

It is assumed that a binary matrix representing all the feasible TSP tours is given prior to the computing stage. Therefore, we provide a new iterative algorithm that synthesizes this binary matrix in an efficient way. The new algorithm is adequate for the more general case of an asymmetric TSP. This means that it produces a $K_{\text{asymmetric}} \times M_{\text{asymmetric}} = [(N-1)!] \times [N(N-1)]$ binary matrix and that both symmetric and asymmetric TSPs can be handled by this algorithm. The proof that this algorithm produces the desired binary matrix is given in Appendix A. A possible optical implementation of this algorithm is going to be proposed in a later paper.

The algorithm starts with a binary matrix representing the case of a three-city TSP and extends the matrix iteratively to a binary matrix that fits a problem of a higher number of cities. The algorithm is

composed of two stages: the initialization stage and the induction stage. In the initialization stage, we build the binary matrix of a three-city TSP, which contains only two feasible tours: City 1 → City 2 → City 3 → City 1 and City 1 → City 3 → City 2 → City 1 (since the algorithm is suited for asymmetric TSPs, the two tours are different from each other). This matrix is given as follows:

$$\mathbf{b}_{\text{initial}} = \mathbf{b}_{N=3} = \begin{array}{c} \\ T_1 \\ T_2 \end{array} \begin{array}{c} \begin{array}{cccccc} w_{1,2} & w_{1,3} & w_{2,1} & w_{2,3} & w_{3,2} & w_{3,1} \end{array} \\ \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 \end{bmatrix} \end{array}, \tag{10}$$

where $T_i$ indicates the binary matrix row, which represents the $i$th tour. For this initial matrix, the corresponding tour lengths are $l_1 = w_{1,2} + w_{2,3} + w_{3,1}$ and $l_2 = w_{1,3} + w_{3,2} + w_{2,1}$.

The order of the weights in this three-city TSP binary matrix as well as the order of the weights in any other binary matrix synthesized by the algorithm is assigned according to the following rule:

$$\begin{aligned} \mathbf{w} = \big[ & w_{1,2},\ w_{1,3},\ w_{1,4},\ w_{1,5},\ \ldots,\ w_{1,i},\ \ldots,\ w_{1,N}, \\ & \underline{w_{2,1}},\ w_{2,3},\ w_{2,4},\ w_{2,5},\ \ldots,\ w_{2,i},\ \ldots,\ w_{2,N}, \\ & \underline{w_{3,2}},\ \underline{w_{3,1}},\ w_{3,4},\ w_{3,5},\ \ldots,\ w_{3,i},\ \ldots,\ w_{3,N}, \\ & w_{4,2},\ w_{4,3},\ \underline{w_{4,1}},\ w_{4,5},\ \ldots,\ w_{4,i},\ \ldots,\ w_{4,N}, \\ & \qquad\qquad \ldots \\ & w_{N,2},\ w_{N,3},\ w_{N,4}, w_{N,5},\ \ldots,\ w_{N,i},\ \ldots,\ w_{N,N-1},\ \underline{w_{N,1}} \big]^T. \end{aligned} \tag{11}$$

The meaning of Eq. (11) is simply taking all the weights having a second index of 1 and indicated by an underline in Eq. (11) (for example, $w_{3,1}$, $w_{4,1}$, $w_{N,1}$, and so on) and inserting them instead of the weights $w_{i,i}$ (for example, $w_{3,3}$, $w_{4,4}$, $w_{N,N}$, respectively).

In the induction stage, we transform the binary matrix representing all feasible tours for an $N$-city TSP into a new binary matrix representing all feasible tours for an $(N + 1)$-city TSP. This is done by the following steps:

I. Define a new empty binary matrix in the size of $[N!] \times [(N + 1)N]$.

II. Divide the new matrix into $N$ horizontal sections, each section containing $(N - 1)!$ rows.

III. In the first section of the new matrix, fill the column belonging to the weight $w_{1,2}$ (the left column) with numerical ones.

IV. Use the old matrix to fill the rest of the first section of the new matrix: From the old matrix, take the column belonging to the weight $w_{i,j}$ and copy it into the column belonging to the weight $w_{i+1,j+1}$ in the first section of the new matrix (for example, $w_{1,2} \rightarrow w_{2,3}$, $w_{2,3} \rightarrow w_{3,4}$, etc.), except in the case of $j = 1$ (for example, $w_{2,1}$, $w_{4,1}$, etc.) in which the column belonging to the weight $w_{i,j}$ in the old matrix is copied into the column belonging to the weight $w_{i+1,j}$ in the first section of the new matrix (for example, $w_{2,1} \rightarrow w_{3,1}$, $w_{3,1} \rightarrow w_{4,1}$, etc.).

V. Fill the unfilled positions in the first section of the new matrix with numerical zeros. Figure 2(a) demonstrates steps I–V for the transition from the $N = 3$ binary matrix to the $N = 4$ binary matrix.

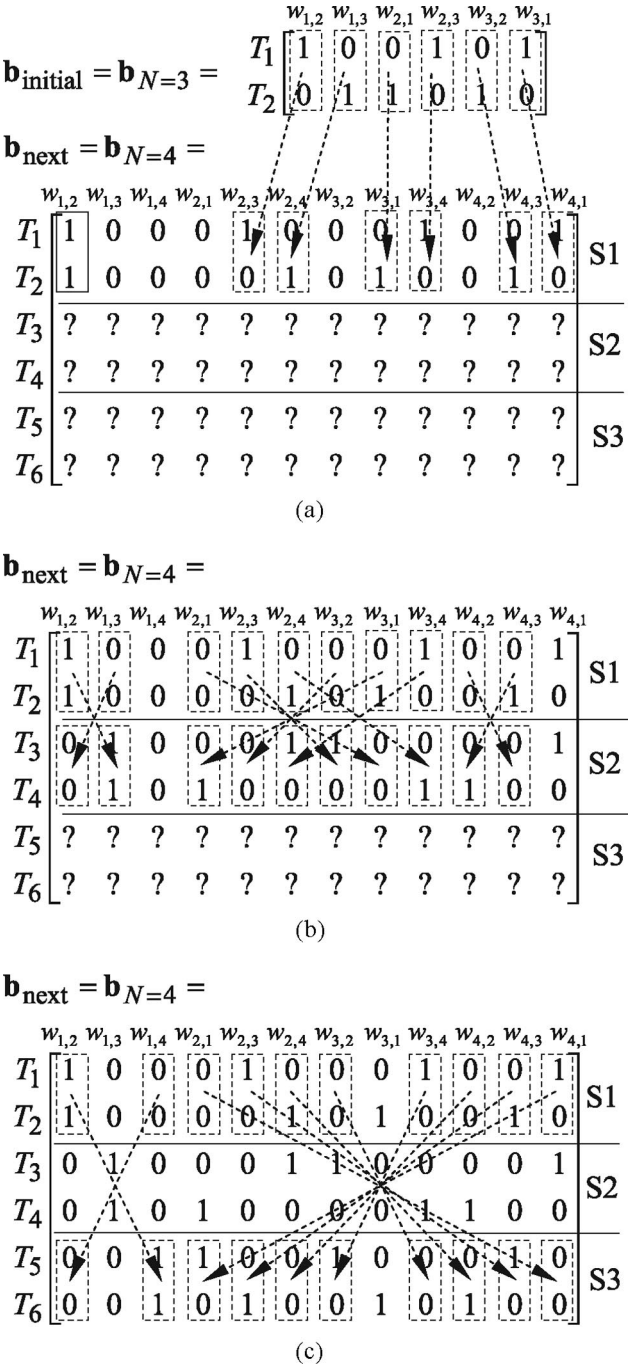



Fig. 2. Example of the induction stage of the algorithm that synthesizes the binary matrix—the transition from the $N = 3$ binary matrix to the $N = 4$ binary matrix: (a) steps I–V, (b) step VI, part 1—use section S1 of the new binary matrix to fill section S2 of this matrix, (c) step VI, part 2—use section S1 of the new binary matrix to fill section S3 of this matrix.

VI. Use the first section of the new matrix to fill entirely the other sections of this matrix: Copy the columns of the first section into the columns of the $k$th section ($k = 2, \ldots, N$), but change the position of the columns by swapping the indices $2 \leftrightarrow (k + 1)$ of

the corresponding weights. The columns that are not related to the weights containing these indices should be copied unchanged. For example, if we want to fill the second section, we use the first section but swap the indices $2 \leftrightarrow 3$ of the corresponding weights. This means that $w_{1,2}$ of the first section goes to $w_{1,3}$ of the second section, $w_{1,3}$ of the first section goes to $w_{1,2}$ of the second section, $w_{2,3}$ of the first section goes to $w_{3,2}$ of the second section, etc. The columns that are not related to a weight containing the indices of 2 or 3 are copied unchanged. For instance, $w_{1,4}$ of the first section goes to $w_{1,4}$ of the second section. The filling of the second section of the $N = 4$ binary matrix by using the first section of this matrix is demonstrated in Fig. 2(b), whereas the filling of the third section of the $N = 4$ binary matrix by using the first section of this matrix is demonstrated in Fig. 2(c).

A mask representing the binary matrix of a six-city TSP is shown in the bottom of Fig. 3. In this mask, 1s are represented by white rectangles, whereas 0s are represented by black rectangles. As indicated by the dashed rectangle in the upper-right part of the $N = 6$ mask, the easy-to-modify $N = 5$ mask is displayed there. The difference between this mask and the exact $N = 5$ mask, shown in Fig. 4(a), is only a few zero columns, which do not affect the solution anyway. In the same way, as shown in Fig. 3, the easy-to-modify $N = 5$ mask contains the easy-to-modify $N = 4$ mask [the exact mask of which is shown in Fig. 4(b)], whereas the easy-to-modify $N = 4$ mask contains the easy-to-modify $N = 3$ mask [the exact mask of which is shown in Fig. 4(c)].

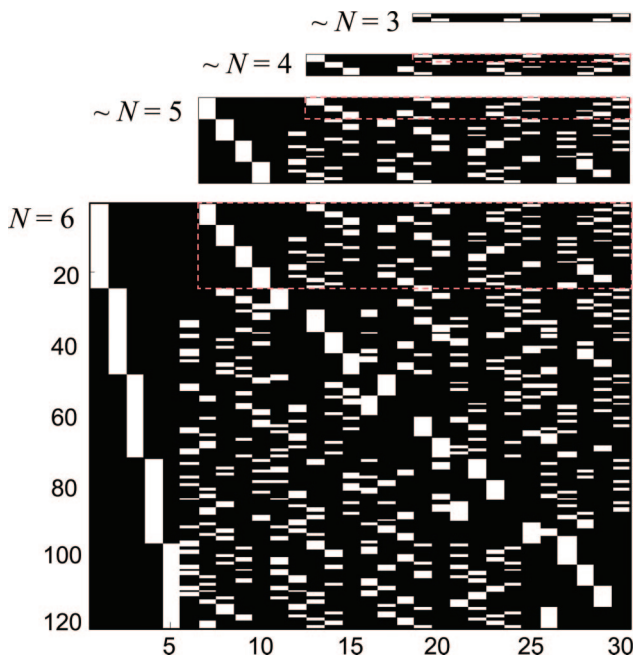



Fig. 3. (Color online) The exact mask (white = 1, black = 0) that represents the binary matrix of $N = 6$, contains the easy-to-modify mask of $N = 5$. The latter mask contains the $N = 4$ easy-to-modify mask, which contains the $N = 3$ easy-to-modify mask.

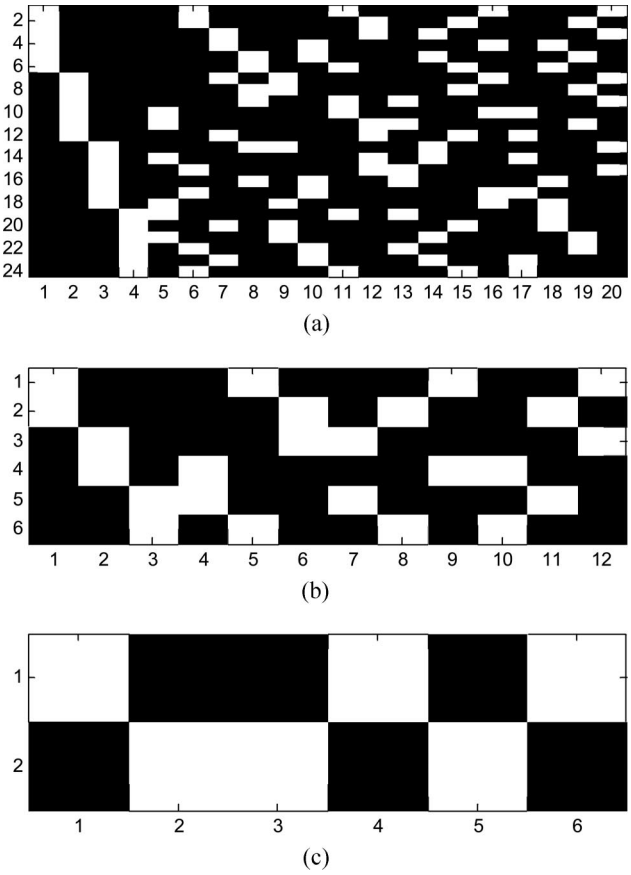



Fig. 4. Exact masks (white = 1, black = 0) that represent the binary matrices of (a) $N = 5$, (b) $N = 4$, and (c) $N = 3$.

One of the advantages of the new algorithm is the use of a small number (of the order of $N^4$) of loops and the duplications of large vectors [of the order of $(N - 1)!$] without loops. This feature enables an implementation of this algorithm in optics. This will be presented in a future paper. Another advantage of the new algorithm is that the $(N - 1)$-city TSP binary matrix is contained in the $N$-city TSP binary matrix. This means that once the $N$-city TSP binary matrix is synthesized, it can be easily used to solve all TSPs with $N$ or fewer cities. These advantages are in addition to the inherent advantage of the current TSP solver, according to which after the binary matrix of an $N$-city TSP is synthesized, all $N$-city TSPs can be solved optically by only changing the weight vector in the matrix–vector multiplication and without changing the binary matrix that is synthesized only once.

### 4. Analysis of the Optical System

To perform the matrix–vector multiplication described above in an optical way, any optical matrix–vector multiplier can be used. In this paper, we have chosen to use the JTC (joint transform correlator)[7,8] to perform this multiplication. The operation of the JTC is divided into two main stages: the recording stage and the reading stage.

In the recording stage, the input plane is composed of the representations of both the weight vector **w** and the binary matrix **b** separated from each other.[7] In the binary matrix **b**, a 0 is represented by a black rectangle, whereas a 1 is represented by a white rectangle. Note that in the optical implementation, the binary matrices are transposed compared to the binary matrices used in Section 3. This means that in the optical implementation, each column in the binary matrix represents a different feasible TSP tour, whereas each row in this matrix represents a different edge (or weight).

In the weight vector representation, the weight $w_{i,j}$ is represented by a gray-scale rectangle, for which the gray-scale level $GS_{i,j}$ is given as follows:

$$GS_{i,j} = GS_{\max} - \frac{w_{i,j} - \min(\mathbf{w})}{\max(\mathbf{w}) - \min(\mathbf{w})}\{GS_{\max} - GS_{\min}\}, \tag{12}$$

where $\min(\mathbf{w})$ and $\max(\mathbf{w})$ are the minimum and the maximum values of the weight vector, respectively, and $GS_{\min}$ and $GS_{\max}$ are the minimal and maximal gray-scale levels that may be used, respectively. Note that by using Eq. (12), the lowest gray-scale level in the representation of the weight vector is always $GS_{\min}$, whereas the highest gray-scale level is always $GS_{\max}$. In this way, the entire gray-scale range can be used. Also notice that by using Eq. (12), higher weights get lower gray-scale levels, whereas lower weights get higher gray-scale levels.

Assuming that the binary matrix **b** is represented on the input plane by the 2D function $b$ centered around the point $(+X_b, -Y_b)$ and that the weight vector **w** is represented on the input plane by the 2D function $w$ centered around the point $(-X_w, +Y_w)$, the input plane is mathematically expressed as the following:

$$U_1(x_1, y_1) = [b(x_1 - X_b, y_1 + Y_b) + w(x_1 + X_w, y_1 - Y_w)] \times \sum_n \operatorname{rect}\left[\frac{x_1 \cos\theta + y_1 \sin\theta - n\alpha}{\alpha/2}\right], \tag{13}$$

where the last sum is a 1D grating with a cycle length of $\alpha$, tilted by an angle of $\theta$ to the horizontal axis and added to the input plane in order to duplicate the joint spectra off axis and thus avoid the intense noise induced at the origin of the Fourier plane by the spatial light modulator (SLM) displaying the input plane.

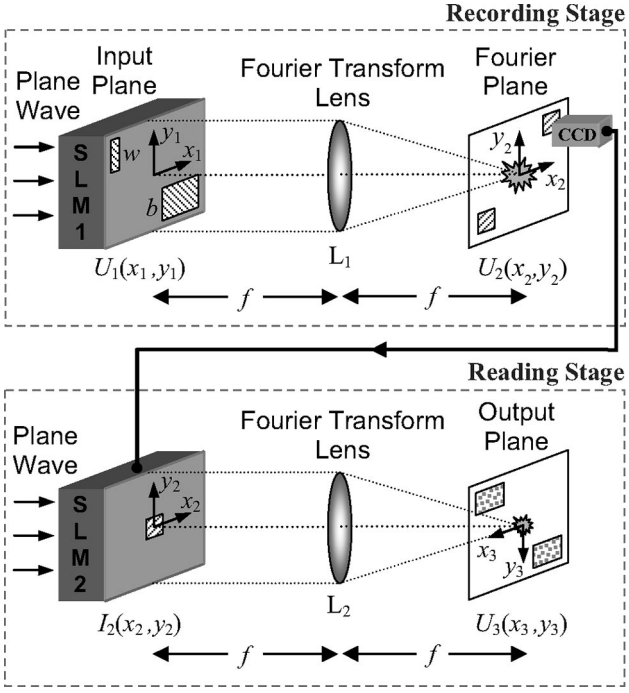



Fig. 5. JTC scheme.

distribution given in Eq. (13) is obtained on the rear focal plane of the positive lens and is given by

$$U_2(x_2, y_2) = A_1\big[B(\mu x_2, \mu y_2)e^{-j2\pi\mu(x_2X_b - y_2Y_b)} + W(\mu x_2, \mu y_2)e^{j2\pi\mu(x_2X_w - y_2Y_W)}\big] * \left[\operatorname{sinc}\left(\frac{\mu\alpha(x_2\cos\theta + y_2\sin\theta)}{2}\right) \times \sum_m \delta\left(x_2\cos\theta + y_2\sin\theta - \frac{m}{\mu\alpha}\right)\right], \tag{14}$$

where $B$ and $W$ are the Fourier transforms of the input-plane functions $b$ and $w$, respectively; $*$ denotes convolution; $A_1$ is a constant resulting from the Fourier transform; and $\mu = 1/(\lambda f)$ (where $f$ is the focal length of the positive lens $L_1$ and $\lambda$ is the wavelength of the illuminating beam). Only the intensity of the first diffraction order, centered around the point $(1/(\mu\alpha\cos\theta), 1/(\mu\alpha\sin\theta))$ in the Fourier plane, is recorded as follows:

$$\begin{aligned} I_2(x_2, y_2) &= A_2\big|B(\mu x_2, \mu y_2)e^{-j2\pi\mu(x_2X_b - y_2Y_b)} + W(\mu x_2, \mu y_2)e^{j2\pi\mu(x_2X_w - y_2Y_w)}\big|^2 \\ &= A_2\big[|B(\mu x_2, \mu y_2)|^2 + |W(\mu x_2, \mu y_2)|^2 + B(\mu x_2, \mu y_2)W^*(\mu x_2, \mu y_2)e^{-j2\pi\mu[x_2(X_b + X_w) - y_2(Y_b + Y_w)]} \\ &\quad + B^*(\mu x_2, \mu y_2)W(\mu x_2, \mu y_2)e^{j2\pi\mu[x_2(X_b + X_w) - y_2(Y_b + Y_w)]}\big], \end{aligned} \tag{15}$$

The upper part of Fig. 5 illustrates the recording stage of the JTC. The Fourier transform of the field

where $B^*$ and $W^*$ are the complex conjugate functions of $B$ and $W$, respectively, and $A_2$ is a constant.

The lower part of Fig. 5 illustrates the reading stage of the JTC. A transparency with an amplitude transmittance proportional to the intensity written in Eq. (15) is illuminated again by a plane wave and Fourier transformed again by another positive lens $L_2$ (assuming for simplicity that the focal lengths of $L_1$ and $L_2$ are the same). This Fourier transform is obtained on the rear focal plane of the lens. This plane is used as the output plane of the JTC. The output image on this plane is given by the following distribution:

$$\begin{aligned} U_3(x_3, y_3) = (A_2/\mu)[&b(x_3, y_3) \otimes b(x_3, y_3) + w(x_3, y_3) \\ &\otimes w(x_3, y_3) + b(x_3, y_3) \otimes w(x_3, y_3) \\ &* \delta(x_3 - (X_b + X_w), y_3 + (Y_b + Y_w)) \\ &+ w(x_3, y_3) \otimes b(x_3, y_3) \\ &* \delta(x_3 + (X_b + X_w), y_3 - (Y_b + Y_w))], \end{aligned} \quad (16)$$

where $\otimes$ denotes correlation. The first two terms of Eq. (16) are useless, whereas the third and fourth terms are the cross-correlation expressions between the functions $b$ and $w$. The third term of Eq. (16) is centered at coordinates $(X_b + X_w, -Y_b - Y_w)$, whereas its fourth term is centered at coordinates $(-X_b - X_w, Y_b + Y_w)$. Each of these terms is a mirror reflection of the other term. The cross-correlation expressions between the functions $b$ and $w$ are actually correlation matrices, in each of which the middle row indicates the lengths of the corresponding tours represented by the corresponding columns of the binary matrix. Since Eq. (12) is used for encoding the weights (which means that higher weights get lower gray-scale levels), the higher the peak of light in the middle row of the correlation matrix, the shorter the tour corresponding to it. That is why the highest peak in the middle row of the correlation matrix corresponds to the best (shortest) TSP tour. The exact forms of the Fourier plane and of the output plane are demonstrated in the next sections by using few TSP examples.

The problem of extracting the highest peak from a length vector, which contains $K_{\text{symmetric}} = (N - 1)!/2$ elements for the symmetric case or $K_{\text{asymmetric}} = (N - 1)!$ elements for the asymmetric case, can be handled by using a variable optical threshold device[11] or implementing optically a winner-takes-all neural network.[4,12] This implementation is out of the scope of the current paper. However, as explained in the first section, the Hamiltonian path NP-complete problem can be solved with the same optical design by just deciding whether there is a certain light intensity in the output of the optical system and without detecting the highest peak location. Moreover, the solution to a #P-complete problem can be obtained by counting the number of paths represented as constant peaks of light in the middle row of the correlation matrix.

As explained above, the TSP length vector is represented by the peaks displayed across the middle row of the correlation matrix. However, the peaks shown in the upper and lower rows of this matrix also contain valuable information. These peaks represent the correlations between part of a shifted version of the weight vector and part of the binary matrix. If the minimal gray-scale level is zero (or black), the maximal weight in the weight vector is represented by a black rectangle [according to Eq. (12)]. Therefore, the peaks appearing in the $k$th row below the middle row of the correlation matrix are related to a new weight vector obtained by shifting the original weight vector $k$ positions down and inserting the maximum weight of the original weight vector $k$ times in the beginning of the new weight vector. For example, if the weight vector is $\mathbf{w} = [w_{1,2}, w_{1,3}, w_{1,4}, w_{2,3}, w_{2,4}, w_{3,4}]^T$ and its maximum value is $w_{\max}$, the third row below the middle row of the correlation matrix corresponds to the new weight vector: $\mathbf{w} = [w_{\max}, w_{\max}, w_{\max}, w_{1,2}, w_{1,3}, w_{1,4}]^T$. This new weight vector defines a new TSP. In a similar way, the peaks appearing in the $k$th row above the middle row of the correlation matrix are related to a new weight vector obtained by shifting the original weight vector $k$ positions up and inserting the maximum weight of the original weight vector $k$ times in the end of the new weight vector. In the above-mentioned example, it can be said that the second row above the middle row of the correlation matrix corresponds to the new weight vector: $\mathbf{w} = [w_{1,4}, w_{2,3}, w_{2,4}, w_{3,4}, w_{\max}, w_{\max}]^T$, which again defines another new TSP. Actually, the correlation matrix has $(2M - 1)$ rows (where $M$ is the number of TSP weights). This means that besides the main TSP defined in advance, for which the length vector is represented by the middle row of the correlation matrix, there are $(2M - 2)$ other TSPs solved simultaneously and represented by the other rows in the correlation matrix except the middle row. This provides an additional advantage of using a correlator to perform the matrix–vector multiplication rather than using other matrix–vector multipliers.[13,14]

## 5. Simulation Results

In this section, two TSP examples are simulated. The first simulation is designed to solve a seven-city symmetric TSP. In this example, the binary matrix is synthesized by using the new algorithm explained in Subsection 3.B, and we do not use a diagonal grating on the input plane. This grating must be used in the real SLM-based implementation, but it is not necessary for the simulation case. The second simulation solves a five-city TSP and imitates the real optical experiment described in Section 6. Therefore, a diagonal grating is added to the input plane this time.

### A. Seven-City Traveling Salesman Problem Simulation Using the Binary Matrix Algorithm

In the first simulation, we use the proposed method to solve the seven-city symmetric TSP shown in Fig. 1. Using Eqs. (1) and (2), the number of feasible tours in this case is $K_{\text{symmetric}} = (N - 1)!/2 = (7 - 1)!/2 = 360$, whereas the number of weights in this case is $M_{\text{symmetric}} = N(N - 1)/2 = 7(7 - 1)/2 = 21$. As shown in Fig. 1, the shortest tour in this example (which is

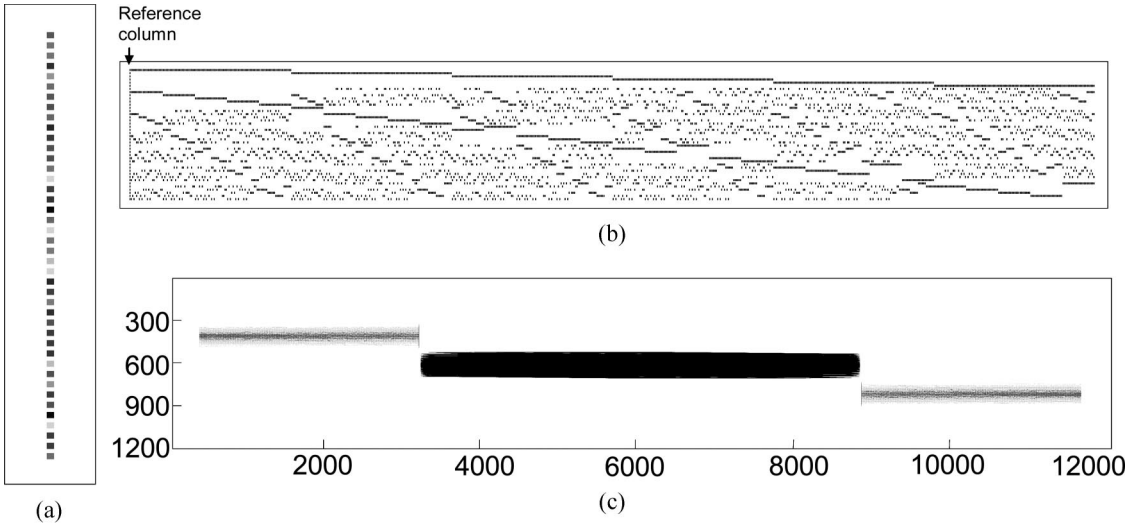



Fig. 6. JTC input and output planes for the seven-city TSP shown in Fig. 1. Note that the contrast of all images in this figure is inverted for a better visualization: (a) weight vector, (b) binary matrix including a reference column on the left, (c) output plane.

indicated by the bold lines in the figure) is City 1 → City 5 → City 7 → City 3 → City 6 → City 2 → City 4 → City 1. The length of this tour is the sum of the weights of the edges connecting the cities in the tour: $w_{1,5} + w_{5,7} + w_{3,7} + w_{3,6} + w_{2,6} + w_{2,4} + w_{1,4} = 2.484 + 3.097 + 0.388 + 2.775 + 3.065 + 4.904 + 5.388 = 22.101$. To synthesize the binary matrix, we use the new algorithm described in Subsection 3.B. Since this algorithm is suitable for an asymmetric TSP, it generates a binary matrix with a double number of tours (for $N = 7$: 720 tours) and a double number of weights (for $N = 7$: 42 weights) compared with the symmetric case. Therefore, the weight vector is also twice as long compared to that of the symmetric case. The expected results in this case are two identical optimal solutions—a tour and its reversed-order version. By getting these results, we actually demonstrate the algorithm for both the symmetric and the asymmetric cases. Although solving a symmetric TSP with the new binary matrix algorithm presented in Subsection 3.B seems like a waste, let us keep in mind that the size multiplication factors (2 for the weight vector and 4 for the binary matrix) are constant and independent of $N$. This is a worthwhile cost, especially if $N$ is large, due to the advantages of the new algorithm.

As shown in Fig. 6(a), the weight vector in this case ($N = 7$) has $M_{\text{asymmetric}} = 42$ elements. This vector is located at the left-upper part of the JTC input plane. The binary matrix shown in Fig. 6(b) has $M_{\text{asymmetric}} = 42$ rows and $K_{\text{asymmetric}} + 1 = 721$ columns. This matrix is located at the right-lower part of the JTC input plane. Note that an additional column is added to the left side of the original 720-column binary matrix. The additional column is a reference column, and it is used to simplify finding both the first column and the middle row of the correlation matrix on the output plane. This result is demonstrated next.

Only a single order appears on the JTC Fourier plane resulting from the input plane described above. On the other hand, the JTC output plane shown in Fig. 6(c) contains three orders: the zero order, which is centered around the center of the output plane as predicted by Eq. (16), and the two side orders, which contain the valuable information.

Figure 7 illustrates the analysis of the JTC output plane shown in Fig. 6(c). As explained before, a reference column, added to the left side of the binary matrix (in the JTC input plane) shown in Fig. 6(b), appends an additional column on the side of the correlation matrices as shown in Fig. 6(c). A vertical cross section across the left column of the right correlation matrix (at the $x = 8917$ coordinate of the JTC output plane) is shown in Fig. 7(a). The highest peak in this cross section appears at the $y = 833$ coordinate of the JTC output plane. This peak should appear exactly in the middle of this column since it indicates a maximum correlation between the weight vector and the reference column. This is the reason why $y = 833$ is the middle-row coordinate of the correlation matrix. Next, we look at the horizontal cross section across the $y = 833$ coordinate of the JTC output plane. This cross section is shown in Fig. 7(b). As shown in this figure, this cross section contains 721 peaks coinciding with the 721 columns of the binary matrix. Ignoring the first peak, obtained from the reference column, the highest peaks in this cross section are for the tour indices 281 and 478. These indices in the binary-matrix columns indicate that the first tour contains the weights $w_{1,4}$, $w_{2,6}$, $w_{3,7}$, $w_{4,2}$, $w_{5,1}$, $w_{6,3}$, and $w_{7,5}$, which means the tour City 1 → City 4 → City 2 → City 6 → City 3 → City 7 → City 5 → City 1, whereas the second tour contains the weights $w_{1,5}$, $w_{2,4}$, $w_{3,6}$, $w_{4,1}$, $w_{5,7}$, $w_{6,2}$, and $w_{7,3}$, which means the tour City 1 → City 5 → City 7 → City 3 → City 6 → City 2 → City 4 → City 1. According to these results, one tour is indeed the reversed-order tour of the other one, and both tours are equal to the best (shortest) tour indicated by the bold lines in Fig. 1.

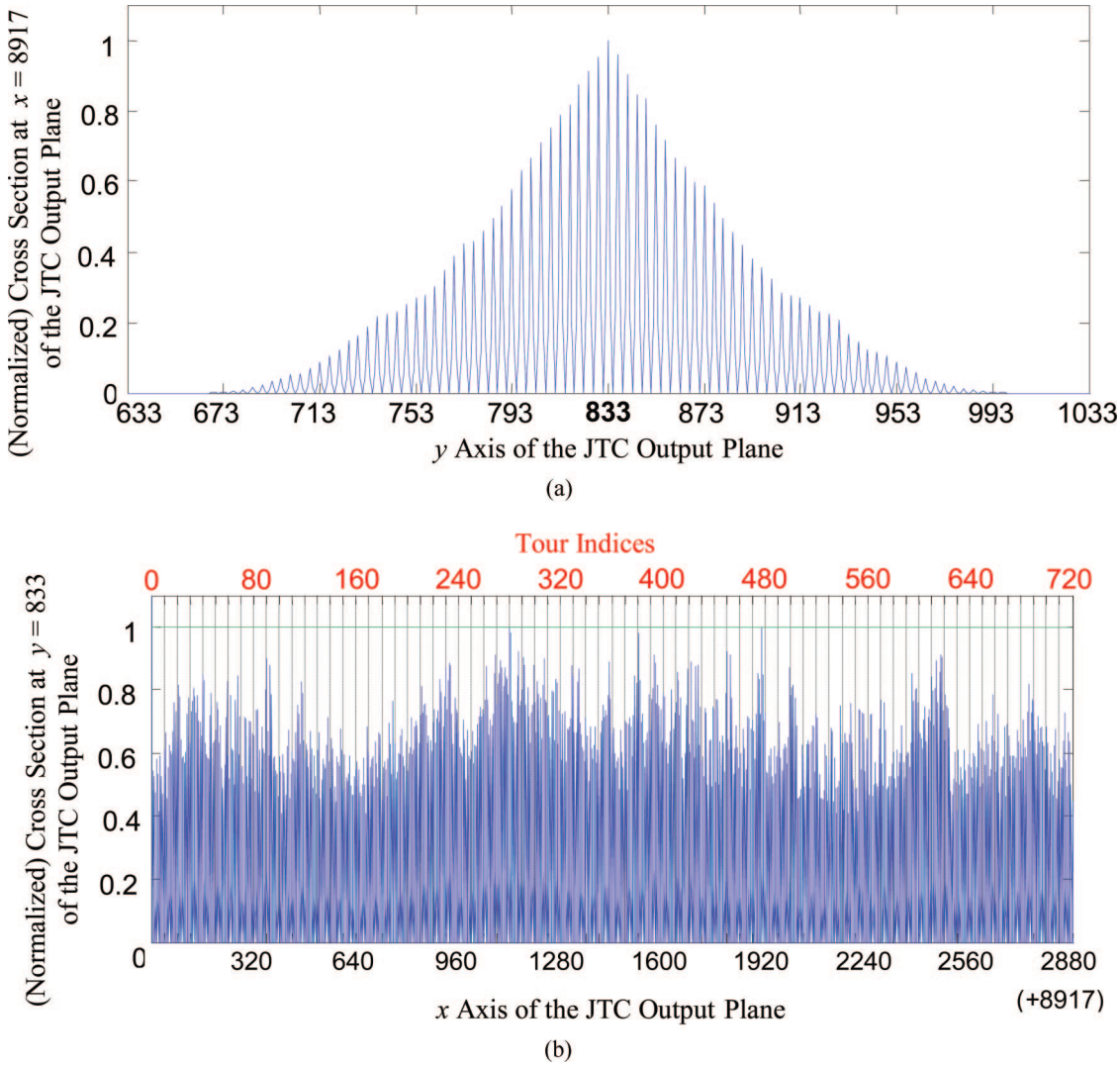



Fig. 7. (Color online) (a) Cross section across the left column (at $x = 8917$) of the correlation matrix shown in the right-lower part of Fig. 6(c). (b) Cross section across the middle row (at $y = 833$) of the correlation matrix shown in the right-lower part of Fig. 6(c).

### B. Five-City Traveling Salesman Problem Simulation with a Grating on the Input Plane

Figure 8 illustrates the five-city symmetric TSP solved in the second simulation and also in the optical experiment. The best (shortest) tour is indicated in this figure by bold lines. In this simulation, we do not use the new algorithm for synthesizing the binary matrix because the aim here is to demonstrate the operation of the optical correlator as a TSP solver. For a low number of five cities, the new algorithm, which is suitable for the more general case of an asymmetric TSP, is not required, and since the SLM has a limited active area, it is even not desired. The weight vector in this case ($N = 5$) has $M_{\text{symmetric}} = 10$ elements. This vector is located at the left-upper part of the JTC input plane as shown in Fig. 9(a). The binary matrix in this case has $M_{\text{symmetric}} = 10$ rows and $K_{\text{symmetric}} + 1 = 13$ columns. This matrix is located at the right-lower part of the JTC input plane as shown in Fig. 9(a). A high-frequency diagonal grating is added to this input plane. To better illustrate the grating, Fig. 9(b) shows an enlarged picture of the left-upper portion of the binary matrix shown in Fig. 9(a). The grating on the input plane is not necessary for the simulation alone, but it is necessary for the SLM-based optical experiment presented in the next section. Figure 10 shows the Fourier plane obtained by Fourier transforming the JTC input plane. Contrary to the JTC Fourier plane of the first simulation, as a result of the diagonal grating added to the JTC input plane, the current JTC Fourier plane contains side orders. To obtain the JTC output plane, one of the side orders is Fourier transformed. This output plane is shown in Fig. 11(a). It contains three orders, and again we choose one of the side orders. An enlarged version of the right-bottom side order is shown in Fig. 11(b). This pattern is the correlation matrix in which the highest peak of the middle row indicates the shortest tour of the TSP. The left column in the correlation matrix is a result of the reference column added to the left of the binary matrix on the input plane. A vertical cross section across the left column

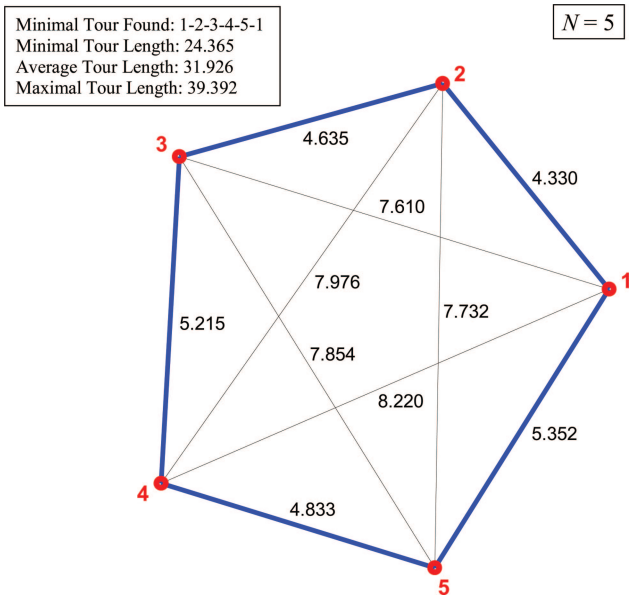



Fig. 8. (Color online) Five-city symmetric TSP solved in both the second simulation and the optical experiment. The best (shortest) tour is indicated by bold lines.

of the correlation matrix helps us find the middle row of the correlation matrix. A horizontal cross section across the middle row of this matrix is shown by the dashed curve in Fig. 11(c). From this figure, it is obvious that tour 9 represented by column 10 of the binary matrix is the winner. If one checks this column in the binary matrix, he sees that the tour represented by this column is composed of the following weights: $w_{1,2}$, $w_{1,5}$, $w_{2,3}$, $w_{3,4}$, and $w_{4,5}$. This means that the best tour is City 1 → City 2 → City 3 → City 4 → City 5 → City 1. This is indeed the shortest tour indicated by the bold lines in Fig. 8.

## 6. Experimental Results

The experiment introduced in this section is based on the second simulation presented in the previous section. In the experiment, the JTC input plane shown in Fig. 9(a) is displayed on a computer-controlled SLM (CRL Opto XGA3, 1024 × 768 pixels). This input plane modulates an expanded laser (Uniphase 1144/P, 17 mW, 632.8 nm, HeNe-polarized laser) beam and Fourier transformed by a positive lens. To magnify the image displayed on the JTC Fourier plane, a negative lens is used right before the original JTC Fourier plane. Since a diagonal grating is used on the JTC input plane, there are side orders on the JTC Fourier plane. A CCD camera (Sony XC75-CE), located on the JTC Fourier plane and centered around one of the side orders, records the intensity pattern and sends it back to the computer. This intensity pattern is shown in Fig. 12(a).

In the next stage, we use the computer again to write the spatial spectrum on the SLM. Then, a positive lens is used to perform an additional Fourier transform and a negative lens is used to magnify the image displayed on the JTC output plane. This plane

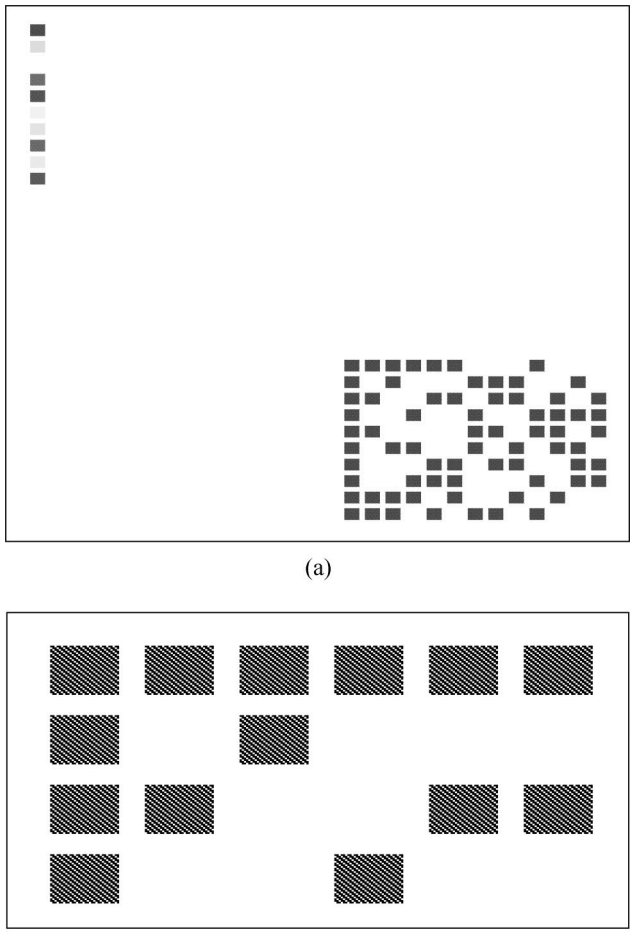



Fig. 9. (a) JTC input plane (contrast-inverted picture) for the five-city TSP shown in Fig. 8. (b) Enlarged picture of the left-upper portion of the binary matrix shown in (a).

contains three orders and each of the side orders contains the useful information. A CCD camera located on the JTC output plane records one of the side orders. This recorded distribution is the correlation matrix, in which the middle row contains a set of peaks representing the matrix–vector product. The

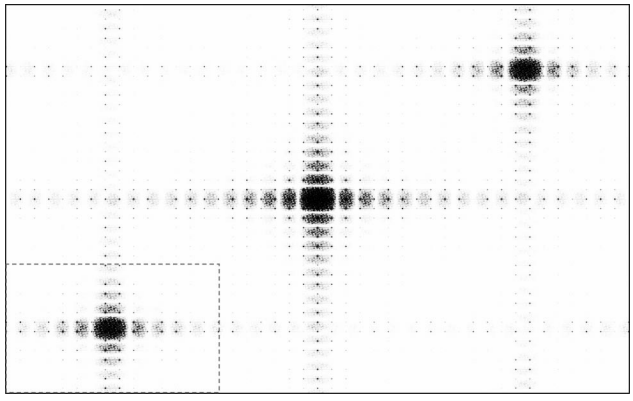
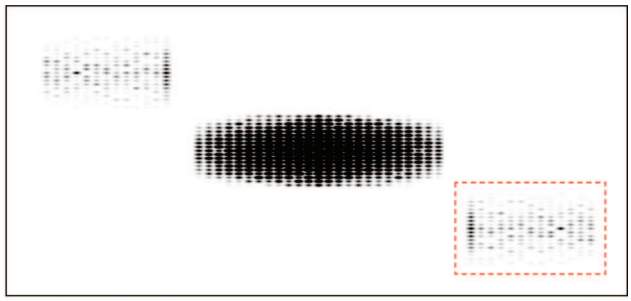
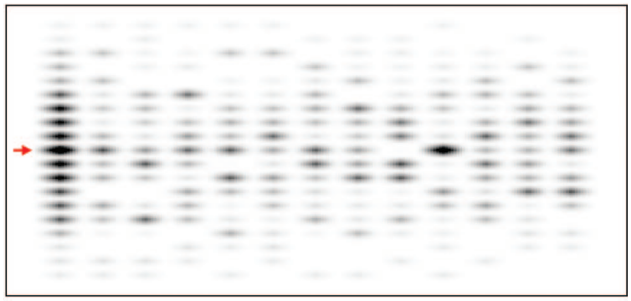


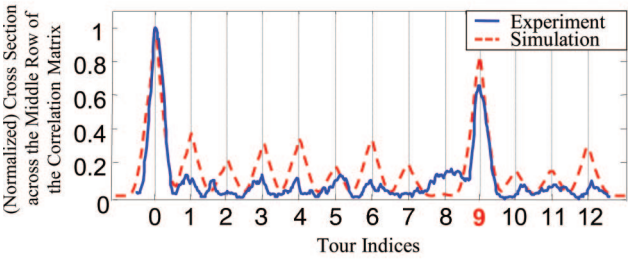
Fig. 10. JTC Fourier plane (contrast-inverted picture) for the five-city TSP shown in Fig. 8.



(a)

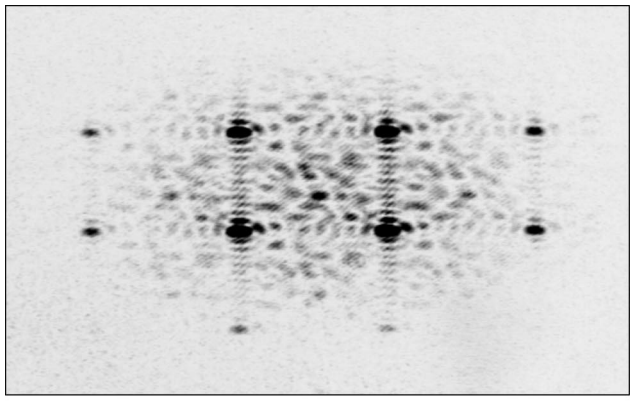


(b)

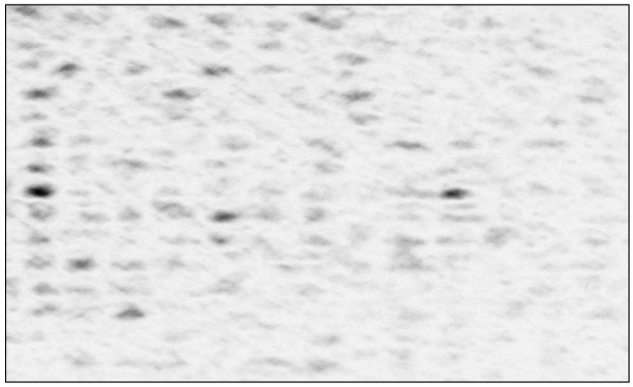



(c)

Fig. 11. (Color online) JTC output plane for the five-city TSP shown in Fig. 8: (a) Entire output plane (contrast-inverted picture) obtained by simulation. (b) Zoomed-in picture of the right-bottom correlation matrix shown in (a). (c) Comparison between the cross section across the middle row of the correlation matrix obtained by simulation (dashed curve) and by experiment (solid curve).



(a)



(b)

Fig. 12. Experimental results (contrast-inverted pictures) for the five-city TSP shown in Fig. 8: (a) One of the side orders on the JTC Fourier plane, (b) one of the correlation matrices on the JTC output plane.

highest peak in this row coincides, as explained in Section 4, with the best (shortest) TSP tour. Figure 12(b) shows the correlation matrix recorded on the JTC output plane by the CCD camera. As explained in Section 5, the first column in this matrix is obtained as a result of the reference column added to the left of the binary matrix on the JTC input plane. A horizontal cross section is taken across the middle row of the correlation matrix. This cross section is shown in Fig. 11(c) by a solid curve and compared with the simulation results marked by a dashed curve. As shown in this figure, in spite of the moderately noisy results, it is obvious that the winning column is indeed column 10 (tour 9). In addition, the graphs show good agreement between the simulation and experimental results. This proves the validity of the proposed optical TSP solver.

Since the different weights are represented by different gray-scale levels, the dynamic range of the gray-scale levels on the slide limits the dynamic range of the weights that can be represented in this slide and hence limits the number of different tour lengths that can be represented by the output peaks. Many close-value weights cause many close-intensity peaks in the output and as a result, it may be hard to differentiate between these peaks. In fact, the modulation resolution of our SLM is 8 bits (256 linear gray-scale levels). This means that only weights that are able to yield integer gray-scale levels between 0 and 255 can be used. Electronic noise in the SLM and CCD camera and the closeness to the zeroth SLM order (especially in the JTC Fourier plane) are additional reasons for the moderate noise that occurs across the solid curve in Fig. 11(c). Implementing the system with film-based slides instead of SLM-based slides may enable us to increase the number of the gray-scale levels and also to eliminate the electronic noise of the SLM (so less noise will be seen in the output of the system). On the other hand, using film-based slides inhibits the real-time properties of the proposed system (accepting the weights in real time and using them to perform the multiplication), since a film-based slide representing the weights would have to be prepared before the processing itself. In any case, the processing of checking all TSP tours by

the proposed system is always done much faster than by a conventional computer.

## 7. Discussion and Conclusion

We have presented a new optical method for obtaining a parallel, fast, and reliable solution to the TSP. The proposed design is based on an optical matrix–vector multiplication. To perform the multiplication, we use the JTC. The multiplication is performed between a binary matrix, representing the TSP feasible tours, and a gray-scale weight vector, representing the TSP weights. The result of this multiplication is a vector containing all the lengths of the TSP feasible tours. The minimum (but maximum in the optical implementation) element in this vector indicates the best (shortest) TSP tour. Using the proposed optical design to solve a TSP of a certain rank requires that the binary matrix has to be produced only once. To synthesize the binary matrix, a new efficient algorithm is proposed. Using the same matrix, other TSPs of the same rank are solved by only changing the TSP weight vector. We have proved the validity of the proposed design by using both simulations and optical experiments. The obtained optical experiment results show good agreement with the simulation results.

Note that other matrix–vector multipliers can be used to implement the proposed method such as the Stanford multiplier.[13,14] However, we have chosen to use a correlator as a matrix–vector multiplier since the binary matrix tends to be very wide (a large number of tours is represented in this matrix), and it may be advantageous to split it into a few parts, each of which represents part of the TSP tours. Then, these parts should be arranged in a rectangular (or other convenient) shape so that the binary matrix plane is exploited in an optimal way. When using a correlator, since the weight vector is correlated with the entire binary matrix plane, the multiplication with the reshaped binary matrix can be performed within a single optical cycle. On the other hand, if a Stanford multiplier is employed, a few cycles (or a few parallel multipliers) may have to be used to perform the multiplication with the reshaped binary matrix.

Figure 13 shows a comparison of the computation times of the optical processor and an electronic processor for various TSPs, both of them performing an exhaustive search (checking all feasible solutions) and ensuring predefined computation times. Note that the vertical axis in this figure is logarithmic.

To estimate the computation time of the electronic processor, working in frequencies of the order of gigahertz, we have assumed that the electronic processor can check $10^9$ tours per second.[10] This means that the electronic processor's calculation time in seconds can be estimated as the number of TSP feasible tours $[(N-1)!/2$, where $N$ is the number of TSP cities$]$ divided by $10^9$.

To estimate the computation time of the optical processor, we have assumed a $4f$ optical system,

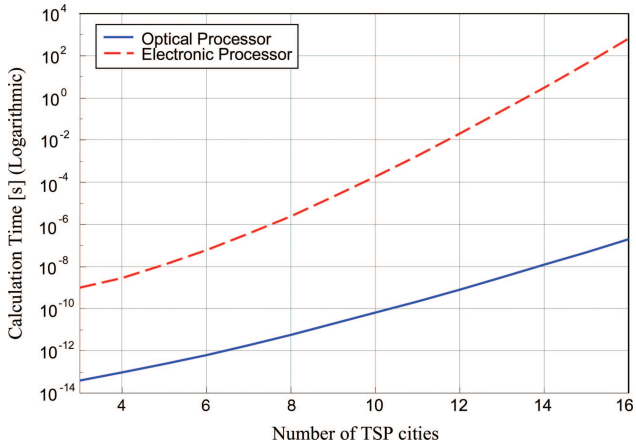



Fig. 13. (Color online) Comparison between the computation times of the optical and electronic processors for various TSP ranks.

which has to grow bigger as the number of TSP cities increases. For the calculation of the optical processor's computation time, as shown in Fig. 13, the binary matrix synthesis is not taken into consideration since it is regarded as preprocessing. This matrix has to be synthesized only once for the biggest TSP, which needs to be solved, no matter what the TSP weights are. Then, even smaller TSPs can be solved by using the same synthesized binary matrix. Therefore, we can regard this stage as a prepreparation of a special optical hardware. An efficient optical method for synthesizing this matrix by the proposed algorithm will be addressed in our future paper.

As can be seen from Fig. 13, TSPs that contain approximately 15 cities can be solved by a single iteration of the proposed optical processor within tens of nanoseconds (can be considered as real-time performance), whereas the electronic processor can perform this exhaustive search within tens of seconds (cannot be considered as real-time performance).

As mentioned before, as the number of TSP cities increases, the size of the optical system increases as well. Therefore, there is a problem in solving TSPs with more than 15 or 16 cities, within a single optical iteration, and by a wavelength in the optical regime. Decreasing the wavelength might help reduce the size of the binary matrix and thus enable the solutions of larger TSPs. Prospective implementations might use wavelengths in the x-ray region or even in the gamma-ray region to solve TSPs that contain a much higher number of cities. However, this option is quite limited due to the currently available light sources, lenses, and SLMs. Another possible solution to the scalability problem is to perform optical iterations (or to concatenate several optical systems). In each iteration (or in each concatenated optical system), a different part of the binary matrix is used. This option is also quite limited due to the relatively low speed of the currently available SLMs. Anyway we believe that the real-time performance of the system (which can be obtained for small TSPs, up to 15

or 16 cities, by using the currently available technologies) together with the predefined calculation time property signify the advantages of the proposed optical method. These properties can be very useful for many fields, such as cryptography,[15] real-time satellite route decisions, etc. In our future research, we will show how the binary matrix algorithm can be efficiently implemented in an optical way, and we will also try to generalize the proposed TSP solver to other hard combinatorial problems.

### Appendix A: Correctness of the Binary Matrix Algorithm

This appendix provides a proof that the new iterative binary matrix algorithm (described in detail in Subsection 3.B) produces a binary matrix, which exactly represents all feasible tours of an $N$-city asymmetric TSP. To do this, three lemmas are introduced and proven.

**Lemma 1:** *There are exactly $(N-1)!$ tours in the N-city asymmetric TSP binary matrix.*

*Proof*: This lemma can be proven by using an induction on $N$. For $N = 3$ cities, there are $(N-1)! = (3-1)! = 2$ feasible tours (so the three-city TSP binary matrix contains two rows). Then, by going from $N$ cities to $(N+1)$ cities, we copy the tours (rows) of the $N$-city TSP binary matrix $N$ times. So, if we assume that the number of tours (rows) in the $N$-city TSP binary matrix is $(N-1)!$, it will be $N(N-1)! = N!$ in the $(N+1)$-city TSP binary matrix. Q.E.D.

**Lemma 2:** *All tours represented in the N-city asymmetric TSP binary matrix are different from each other.*

*Proof*: This lemma can also be proven by using an induction on $N$. For $N = 3$ cities, the two tours represented in the suitable binary matrix are indeed different from each other (since we deal with an asymmetric TSP). Then, we assume that all the tours in the $N$-city TSP binary matrix are different from each other. Step IV in the induction stage of the algorithm (Subsection 3.A) actually transforms all the tours containing $N$ cities (which are different from each other) to incomplete tours starting from City 2 and finishing at City 1 (for example, in the transition from $N = 3$ to $N = 4$, the tour City 1 → City 2 → City 3 → City 1 is transformed to the incomplete tour City 2 → City 3 → City 4 → City 1, whereas the tour City 1 → City 3 → City 2 → City 1 is transformed to the incomplete tour City 2 → City 4 → City 3 → City 1). If the source tours are different from each other, the incomplete target tours are also different from each other. This means that in this step, we do not spoil the difference property of the tours. Prior to step IV, step III of the algorithm actually creates a representation of the edge, which connects City 1 and City 2. This means that the edge City 1 → City 2 is added to the beginning of the incomplete tours created in step IV. Step III also does not spoil the difference property of the tours. This is also the case for step V of the algorithm, in which we just add zeros, and for step VI of the algorithm, in which we just swap City 2 and City $k+1$ ($k = 2, \ldots, N$) until the binary matrix is full. Q.E.D.

**Lemma 3:** *All tours represented in the N-city asymmetric TSP binary matrix are feasible.*

*Proof:* Here also an induction on $N$ is used to prove the lemma. For $N = 3$ cities, we build the initial binary matrix such that the represented two tours are feasible. Then, we assume that the tours represented in the $N$-city TSP binary matrix are feasible. Step IV in the induction stage of the algorithm (Subsection 3.B) just increases the indices of the cities (see the proof of Lemma 2 for details), whereas step III of the algorithm adds the missing edge (City 1 → City 2). This of course produces feasible tours so that the feasibility property of the tours is maintained. Step V of the algorithm, which only adds zeros, and step VI of the algorithm, which only exchanges the roles of two cities, also maintain the feasibility property of the tours. Q.E.D.

**Theorem:** *The N-city asymmetric TSP binary matrix exactly represents all feasible tours of an N-city asymmetric TSP.*

*Proof:* As explained in Section 2, the number of feasible tours for an $N$-city asymmetric TSP is $(N-1)!$. Then, by using Lemmas 1–3, we can conclude that the resulting binary matrix contains all of the $(N-1)!$ feasible tours and only them. Q.E.D.

The authors thank Ephraim Korach, Gilad Goldfard, and Ran Nachmany for their valuable comments in the beginning of this research. This research was supported by the Israel Science Foundation grant 119_03 as well as by the Rita Altura Trust Chair in Computer Sciences.

### References

1. D. Harel, *Algorithmics: The Spirit of Computing* (Addison-Wesley, 1987).
2. G. Reinelt, *The Traveling Salesman: Computational Solutions for TSP Applications* (Springer-Verlag, 1994).
3. J. E. Mitchell, "Branch-and-cut algorithms for combinatorial optimization problems," in *Handbook of Applied Optimization* (Oxford U. Press, 2002), pp. 65–77.
4. K. A. Smith, "Neural networks for combinatorial optimization: a review of more than a decade of research," INFORMS J. Comput. **11,** 15–34 (1999).
5. N. Collings, R. Sumi, K. J. Weible, B. Acklin, and W. Xue, "The use of optical hardware to find good solutions of the travelling salesman problem (TSP)," in Proc. SPIE **1806,** 637–641 (1993).
6. D. Gutfreund, R. Shaltiel, and A. Ta-Shma, "If NP languages are hard on the worst-case then it is easy to find their hard instances," in *Proceedings of the 20th Annual Conference on Computational Complexity (CCC)* (2005). A long version of this paper is available on the web at http://www.cs.tau.ac.il/~amnon/Papers/GST.cc05.journal.ps.

7. J. W. Goodman, *Introduction to Fourier Optics,* 2nd ed. (McGraw-Hill, 1996), pp. 232–246.
8. C. S. Weaver and J. W. Goodman, "A technique for optically convolving two functions," Appl. Opt. **5,** 1248–1249 (1966).
9. A. VanderLugt, "Signal detection by complex spatial filtering," IEEE Trans. Inf. Theory **IT-10,** 139–145 (1964).
10. A. Homaifar, S. Guan, and G. E. Liepins, "Schema analysis of the traveling salesman problem using genetic algorithms," Complex Syst. **6,** 183–217 (1992).
11. D. G. Feitelson, *Optical Computing: A Survey for Computer Scientists* (MIT Press, 1988), pp. 148–163.
12. M. A. Karim and A. A. S. Awwal, *Optical Computing: An Introduction* (Wiley, 1992), pp. 328–356.
13. J. W. Goodman, *Introduction to Fourier Optics,* 2nd ed. (McGraw-Hill, 1996), pp. 282–289.
14. A. D. McAulay, *Optical Computer Architectures: The Application of Optical Concepts to Next Generation Computers* (Wiley, 1991), pp. 289–299.
15. S. Dolev, E. Korach, and G. Uzan, "A method for encryption and decryption of messages," PCT patent application WO 2006/001006 (5 January 2006).