The famous Travelling Salesman Problem (TSP) is an important category of optimization problems [1] that is mostly encountered in various areas of science and engineering. Studying optimization problems motivates to develop advanced techniques more suited to contemporary practical problems [2–7]. Among those, especially the NP hard problems provide an apt platform to demonstrate supremacy of quantum over classical technologies in terms of resources and time. TSP is one such NP hard problem in combinatorial optimization [8, 9] which takes exponential time order for solving by brute force method. Here we propose a quantum algorithm to solve the travelling salesman problem using phase estimation technique. We approach the problem by encoding the given distances between the cities as phases. We construct unitary operators whose eigenvectors are the computational basis states and eigenvalues are various combinations of these phases. Then we apply phase estimation algorithm to certain eigenstates which give us all the total distances possible for all the routes. After obtaining the distances we can search through this information using the quantum search algorithm for finding the minimum [10] to find the least possible distance as well the route taken. This provides us a quadratic speedup over the classical brute force method for a large number of cities. In this paper, we illustrate an example of the travelling salesman problem by taking four cities and present the results by simulating the codes in the IBM’s quantum simulator.

Travelling Salesman Problem (TSP) is a classic optimization problem [1] in the field of computer science. It belongs to an intriguing class of ‘hard’ optimization problems called NP hard [11, 12]. The problem involves a salesman who has to travel N cities, visiting each city once and reaching ultimately at the same city where he started. This cycle of visiting each city once, where each city represents a unique vertex in a graph, and returning to the starting city is known as a Hamiltonian cycle [13]. Each city is connected to other cities with a specific cost associated to each connection. The cost gives an idea of how difficult it is to take the corresponding route. The aim of the salesman is to minimize the cost of travel, satisfying the above described conditions. Even if we break the travelling salesman problem into smaller components, each component will be at least as complex as the initial problem. This is why it belongs to the class of NP hard problems. The most expensive and simplest classical solution to the problem is to find the solution by brute force method. However, the problem becomes impossible to solve when a large number of cities are taken. For N cities, (N-1)! possible iterations are needed to search for the solution, which shoots up very fast as N increases. Other classical approaches to solve the problem include branch and bound algorithms [14–17], heuristics [18–21] and other methods [22–24]. Using branch and bound algorithms the problem has been solved for around 86,000 cities, but the success in branch and bound algorithms depends on certain factors which if not satisfied give us the same complexity as the brute force method. Heuristics approach is based on providing a set of rules on optimal selection of next city to travel. But this does not give optimal solution in every case as heuristics result in approximations.

With the advent of the era of quantum technologies possibilities of solving this problem with quantum computers has come to the limelight with the aim to tackle a much bigger problem of proving \( P = NP \) class. Several quantum algorithms [25] have also been proposed aiming at the same. Goswami et al. [26] have presented a framework for efficiently solving the approximate travelling salesman problem. A quantum heuristic algorithm has been proposed by Bang et al. [27] to solve the travelling salesman problem by generalizing the Grover search. Molyett et al. [28] have given a proof of the quantum quadratic speed up for the Travelling Salesman Problem for bounded-degree graphs. The above mentioned algorithms work only when certain conditions are satisfied, however our algorithm combined with the quantum algorithm for finding the minimum by Durr and Hoyer [10] gives a quadratic speedup over the classical brute force method without further conditions on the problem.

The classical algorithms to solve the problem take input in the form of a matrix say \( A \) such that \( A_{ij} = \phi_{ij} \), where \( \phi_{ij} \) is the cost/distance/time or any other quantity taken to travel from city i to city j. This quantity for the overall journey has to be minimized. Without loss of generality, we take the quantity as cost in the current work. In the problem, the main motivation to take input
as phases stems from the following two facts; first the matrix made of the given distances using the above procedure is not unitary in general which implies that the implementation and manipulation of the operator is not possible on a quantum computer. Second, the phases will get added when we multiply them or take tensor products of states with these phases as coefficients, that is the distances will get added as phases which is required for the search. Hence we represent the input as a matrix B, where

\[ B_{ij} = e^{i(\phi_{ij})}. \]

**TABLE I.** The result of simulation for (4-1)! eigenstates and the theoretical expectations are presented in the Supplementary Information Section

| SL No. | Eigenstate | Theoretical | Experimental |
|--------|------------|-------------|--------------|
| 1      | 11000110   | 100100      | 100000       |
| 2      | 01101100   | 100100      | 101000       |
| 3      | 10001101   | 100000      | 010000       |
| 4      | 01110010   | 100000      | 010000       |
| 5      | 11100001   | 011000      | 101000       |
| 6      | 10110100   | 011000      | 010000       |

Even though the experimental results do not exactly coincide with the theoretical expectations, we can rectify this by taking more qubits for phase estimation, which is its inherent property. To successfully obtain the phase accurate up to n bits with probability of success at least 1 - \( \epsilon \), the number of qubits we need for phase estimation \( t \) is given by the following expression,

\[ t = n + \left\lceil \log \left(2 + \frac{1}{2\epsilon}\right)\right\rceil \]
FIG. 2. The above figure shows the quantum circuit for phase estimation of eigenstate $|11000110\rangle$ which corresponds to the route (Hamiltonian cycle) $1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 1$. We have taken here six qubits for estimating the phase and rest eight qubits for preparing the corresponding eigenstate (all of which are initialized in the $|0\rangle$ state) by the use of pauli X gates on specific qubits. The unitary $U$ is the one described earlier i.e. the tensor product unitaries $U_1 \otimes U_2 \otimes ... U_N$. The part of the circuit in red represents initialization of the eigenstates. The part of the circuit in blue performs the phase estimation method.

Using the proposed algorithm, we are able to create a database of all possible routes that can be taken along with the distance of each. If one devices a quantum algorithm to find the minimum element in an unsorted array, which is faster than the one we currently have, then we can use that algorithm to find the minimum. This gives our algorithm a flexibility, which can be exploited in the future to solve the travelling salesman problem much more efficiently.

Even though our algorithm deals with a very general case, there are certain cases which cannot be directly solved using our algorithm. These are the cases where there are restrictions on routes connecting cities. For instance, city i does not have a route connecting it to city j. This can be thought of as the distance between those cities being infinite. Since our algorithm requires distances that can be normalized such that the total distance for the longest route is less than $2\pi$, this does not bode well for us. Fortunately, there is a way to deal with this exception. We can take the distance between the concerned cities as a very large distance such that the routes containing the path connecting city i and j will have a total distance which will certainly exceed the minimum distance.

**Supplementary Information** is available in the online version of the paper.

**Acknowledgments** B.K.B. is financially supported by DST Inspire Fellowship. S.S. and K.S. acknowledge the support of HBCSE and TIFR for conducting National Initiative on Undergraduate Sciences (NIUS) Physics camp. We are extremely grateful to IBM team and IBM QE project. The discussions and opinions developed in this paper are only those of the authors and do not reflect the opinions of IBM or IBM QE team.

**Author contributions** K.S. and S.S. contributed equally to this work. K.S. and S.S. came up with the efficient algorithm to solve the TSP problem. K.S. and S.S. designed the quantum circuit and simulated using IBM quantum experience. K.S., S.S. and B.K.B. contributed to the composition of the manuscript. K.S., S.S. and B.K.B. has done the work under the guidance of P.K.P.

**Author information** The authors declare no competing financial interests. Correspondence and requests for materials should be addressed to P.K.P. (pprasanta@iiserkol.ac.in).

[1] Mott, B., Job, J., Vlimant, J. R., Lidar, D. & Spiropulu, M. Solving a Higgs optimization problem with quantum annealing for machine learning. *Nature* **550**, 375–379 (2017).

[2] Westerlund, T., Harjunkoski, I. & Isaksson, J. Solving a production optimization problem in a paper-converting mill with MILP. *Comput. Chem. Eng.* **22**, 563–570 (1998).

[3] Deb K. & Sinha A. Solving Bilevel Multi-Objective Optimization Problems Using Evolutionary Algorithms. *Evol. Multi-Criterion Optim.* **5467**, 110–124 (2009).

[4] Wang, Z. On Solving Convex Optimization Problems with Linear Ascending Constraints. *arXiv preprint 1212.4701v7* (2014).

[5] Fioretto, F., Pontelli, E. & William, Y. Distributed Constraint Optimization Problems and Applications: A Survey. *J. Artif. Intell. Res.* (2018).

[6] Zhao, S. & Zhou, J. Solutions to Constrained Optimal Control Problems with Linear Systems. *J. Optim. Theory Appl.* 1–14 (2018).

[7] Butenko, S., Murphey, R., & Pardalos, P. M. Recent Developments in Cooperative Control and Optimization. *Springer* (2003).

[8] Dai, H., Khalil, E. B., Zhang, Y., Dilkina, B. & Song, L. Learning Combinatorial Optimization Algorithms over Graphs. *arXiv preprint 1704.01665v4* (2018).

[9] Hoffman, K. L. Combinatorial optimization: Current suc-
SUPPLEMENTARY INFORMATION: EFFICIENT QUANTUM ALGORITHM FOR SOLVING TRAVELLING SALESMAN PROBLEM

I. 4 CITY EXAMPLE EXPLANATION

The distance matrix for the Travelling salesman problem consisting of four cities is given by

$$D = \frac{1}{2} \begin{bmatrix} 1 & e^{i\phi_{12}} & e^{i\phi_{13}} & e^{i\phi_{14}} \\ e^{i\phi_{21}} & 1 & e^{i\phi_{23}} & e^{i\phi_{24}} \\ e^{i\phi_{31}} & e^{i\phi_{32}} & 1 & e^{i\phi_{34}} \\ e^{i\phi_{41}} & e^{i\phi_{42}} & e^{i\phi_{43}} & 1 \end{bmatrix}$$

(3)

where $\phi_{12} = \phi_{21} = \pi/2$, $\phi_{13} = \phi_{31} = \pi/8$, $\phi_{14} = \phi_{41} = \pi/4$, $\phi_{23} = \phi_{32} = \pi/4$, $\phi_{24} = \phi_{42} = \pi/4$, $\phi_{34} = \phi_{43} = \pi/8$, and the unitaries $U_j$ with $j = 1, 2, 3$ and 4 are as follows

$$U_1 = |00\rangle\langle 00| + e^{i\phi_{21}}|01\rangle\langle 01| + e^{i\phi_{31}}|10\rangle\langle 10| + e^{i\phi_{41}}|11\rangle\langle 11|$$

(4)

$$U_2 = e^{i\phi_{12}}|00\rangle\langle 00| + |01\rangle\langle 01| + e^{i\phi_{32}}|10\rangle\langle 10| + e^{i\phi_{42}}|11\rangle\langle 11|$$

(5)

$$U_3 = e^{i\phi_{13}}|00\rangle\langle 00| + e^{i\phi_{23}}|01\rangle\langle 01| + |10\rangle\langle 10| + e^{i\phi_{43}}|11\rangle\langle 11|$$

(6)

$$U_4 = e^{i\phi_{14}}|00\rangle\langle 00| + e^{i\phi_{24}}|01\rangle\langle 01| + e^{i\phi_{34}}|10\rangle\langle 10| + |11\rangle\langle 11|$$

(7)
We constructed the unitaries $U_j$ by decomposing it as follows;

\[
U_j = \begin{bmatrix}
e^{ia} & 0 & 0 & 0 \\
0 & e^{ib} & 0 & 0 \\
0 & 0 & e^{ic} & 0 \\
0 & 0 & 0 & e^{id}
\end{bmatrix} = \left[ \begin{bmatrix} 1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1 \\
0 & 0 & 0
\end{bmatrix} \otimes \begin{bmatrix} e^{ia} & 0 \\
0 & e^{ib}
\end{bmatrix} \right] \left[ \begin{bmatrix} 1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & e^{i(d+c-a-b)}
\end{bmatrix} \right]
\]

Note that this is the unitary $U_j$, by putting specific values of $a$, $b$, $c$, $d$ we can find decomposition for each $U_1, U_2, U_3, U_4$. For phase estimation we need controlled-$(U_1 \otimes U_2 \otimes U_3 \otimes U_4)$ which is same as $(C - U_1 \otimes C - U_2 \otimes C - U_3 \otimes C - U_4)$, where $C - U_1$ represents controlled-$U_1$. For realizing controlled $U_j$ we implemented controlled each element in the decomposition of $U_j$ in equation 8.

\[
\text{FIG. S1. The figure illustrates our implementation of controlled $U_j$ gate with the use of single qubit unitaries U(a) as described earlier. Here we implement a controlled unitary (C-A) which is equivalent to implementing controlled unitaries that belong to the decomposition of A. In this figure, } x = (d+c-a-b)/2
\]

II. THE EIGENSTATES

Fig. 2 shows the part of the circuit where we estimate the phase corresponding to the route going through cities 1, 2, 3 and 4 in the same order and returning to one. Starting from any other city but following the same order as mentioned will also give the same eigenstate. The eigenstate for any route can be calculated as follows. In a particular route, if we are going from city $i$ to $j$, then each $i$ is uniquely mapped to $j$. Hence we can write $i$ as a function of $j$ i.e, $i(j)$. The eigenstate corresponding to that particular route is,

\[
|\psi\rangle = \bigotimes_j |i(j) - 1\rangle
\]

where $j$ goes from 1 to $n$.

Once these eigenstates are calculated, circuits similar to Fig. 1, with eigenstate qubits initiated to the rest of the eigenstates, can be run in parallel. Then we can search through this database using the quantum search algorithm in the order of $O(\sqrt{(N-1)!})$ steps to find the route with the least cost. If the cost of travelling from city $i$ to city $j$ is the same as city $j$ to city $i$, then we can reduce the number of eigenstates by half of the original value. This means we will be able to search through the data in $O(\sqrt{((N-1)!)/2})$ steps.

III. SUBROUTINES IN IBM QUANTUM EXPERIENCE - CUSTOM TOPOLOGY

In the simulation, we took advantage of the “Add subroutine” under advanced option present in the custom topology [1] and built these unitaries. The qasm code written for the simulation of the circuit (Fig. S2) is given below.

```
#include "qelib1.inc";
qreg q[14];
creg c[6];

#Controlled unitary gate to implement C-U1, C-U2, C-U3 & C-U4
gate uni (a, b, c, d) x, y, z {
```
\begin{verbatim}
cu1 (c-a) x, y;
ul (a) x;
cu1 (b-a) x, z;
ccx x, y, z;
cu1 ((d-c+a-b)/2) x, z;
ccx x, y, z;
cu1 ((d-c+a-b)/2) x, y;
cu1 ((d-c+a-b)/2) x, z;
}

# Controlled U = tensor product of U1, U2, U3 & U4
gate bigU (a, b, c, d, e, f, g, h, i, j, k, l) m, n, o, p, q, r, s, t, u {
  uni (0, a, b, c) m, n, o;
  uni (d, 0, e, f) m, p, q;
  uni (g, h, 0, i) m, r, s;
  uni (j, k, l, 0) m, t, u;
}

# C-U with the values given
gate finU m, n, o, p, q, r, s, t, u {
  bigU (pi/2, pi/8, pi/4, pi/2, pi/4, pi/8, pi/4, pi/8, pi/4, pi/8) m, n, o, p, q, r, s, t, u;
}

# C-U^2

gate finU2 m, n, o, p, q, r, s, t, u {
  finU m, n, o, p, q, r, s, t, u;
  finU m, n, o, p, q, r, s, t, u;
}

# C-U^4

gate finU4 m, n, o, p, q, r, s, t, u {
  finU2 m, n, o, p, q, r, s, t, u;
  finU2 m, n, o, p, q, r, s, t, u;
}

# C-U^8

gate finU8 m, n, o, p, q, r, s, t, u {
  finU4 m, n, o, p, q, r, s, t, u;
  finU4 m, n, o, p, q, r, s, t, u;
}

# C-U^16

gate finU16 m, n, o, p, q, r, s, t, u {
  finU8 m, n, o, p, q, r, s, t, u;
  finU8 m, n, o, p, q, r, s, t, u;
}

# C-U^32

gate finU32 m, n, o, p, q, r, s, t, u {
  finU16 m, n, o, p, q, r, s, t, u;
  finU16 m, n, o, p, q, r, s, t, u;
}

# All the controlled unitaries together
gate fU a, b, c, d, e, f, n, o, p, q, r, s, t, u {
  finU f, n, o, p, q, r, s, t, u;
  finU2 e, n, o, p, q, r, s, t, u;
  finU4 d, n, o, p, q, r, s, t, u;
  finU8 c, n, o, p, q, r, s, t, u;
  finU16 b, n, o, p, q, r, s, t, u;
  finU32 a, n, o, p, q, r, s, t, u;
}

# The rest are the gates for Inverse fourier transform
gate r2 x, y {
  cul (-pi/2) x, y;
}
gate r3 x, y, z {
  cul (-pi/4) x, z;
}
\end{verbatim}
r2 y, z;
}
gate r4 w, x, y, z {
cu1 (-pi/8) w, z;
r3 x, y, z;
}
gate r5 v, w, x, y, z {
cu1 (-pi/16) v, z;
r4 w, x, y, z;
}
gate r6 u, v, w, x, y, z {
cu1 (-pi/32) u, z;
r5 v, w, x, y, z;
}

#The circuit
h q[0];
h q[1];
h q[2];
h q[3];
h q[4];
h q[5];
x q[6];
x q[8];
x q[9];
x q[11];
rf q[0], q[1], q[2], q[3], q[4], q[5], q[6], q[7], q[8], q[9], q[10], q[11], q[12], q[13];
h q[0];
r2 q[0], q[1];
h q[1];
r3 q[0], q[1], q[2];
h q[2];
r4 q[0], q[1], q[2], q[3];
h q[3];
r5 q[0], q[1], q[2], q[3], q[4];
h q[4];
r6 q[0], q[1], q[2], q[3], q[4], q[5];
h q[5];
measure q[0] -> c[0];
measure q[1] -> c[1];
measure q[2] -> c[2];
measure q[3] -> c[3];
measure q[4] -> c[4];
measure q[5] -> c[5];
FIG. S2. The figure depicts the circuit implemented in the custom topology in IBM quantum experience. The codes for the subroutines depicted in the figure are given in the qasm code for the entire circuit presented above. The circuit in the box performs inverse quantum Fourier transform for six qubits.

[1] IBM Quantum Experience https://www.research.ibm.com/ibm-q/