Search on a Hypercubic Lattice using Quantum Random Walk: $d > 2$

Apoorva Patel[1, 2, *] and Md. Aminoor Rahaman[2, †]

[1]*Centre for High Energy Physics, Indian Institute of Science, Bangalore-560012, India*
[2]*Supercomputer Education and Research Centre, Indian Institute of Science, Bangalore-560012, India*
(Dated: February 27, 2010)

Random walks describe diffusion processes, where movement at every time step is restricted to only the neighbouring locations. We construct a quantum random walk algorithm, based on discretisation of the Dirac evolution operator inspired by staggered lattice fermions. We use it to investigate the spatial search problem, i.e. finding a marked vertex on a d-dimensional hypercubic lattice. The restriction on movement hardly matters for $d > 2$, and scaling behaviour close to Grover's optimal algorithm (which has no restriction on movement) can be achieved. Using numerical simulations, we optimise the proportionality constants of the scaling behaviour, and demonstrate the approach to that for Grover's algorithm (equivalent to the mean field theory or the $d \to \infty$ limit). In particular, the scaling behaviour for $d = 3$ is only about 25% higher than the optimal $d \to \infty$ value.

I. SPATIAL SEARCH USING THE DIRAC OPERATOR

The spatial search problem is to find a marked object from a database spread over distinct locations. Its characteristic is that one can proceed from any location to only its neighbours, while inspecting the objects. The problem is conventionally represented by a graph, with the vertices denoting the locations of objects and the edges labeling the connectivity of neighbours. In this work, we study the specific case of searching for a marked vertex on a d-dimensional hypercubic lattice with $N = L^d$ vertices.

The quantum algorithmic strategy for spatial search is to construct a Hamiltonian evolution, where the kinetic part of the Hamiltonian diffuses the amplitude distribution all over the lattice and the potential part of the Hamiltonian attracts the amplitude distribution towards the marked vertex [1]. The optimization criterion for the algorithm is to concentrate the amplitude distribution towards the marked vertex as quickly as possible. Grover constructed the optimal global diffusion operator, but it requires movement from any vertex to any other vertex in just one step. That corresponds to a fully connected graph, or equivalently the mean field theory limit. When movements are restricted to be local (i.e. one can go from a vertex to only its neighbours in one step), the search algorithm slows down. It is then worthwhile to redesign the search algorithm by understanding the extent of slow down due to local movements, and how it depends on the spatial arrangement of the database. That is the question we address quantitatively in this article.

On a periodic lattice, spatial propagation modes are characterized by their wave vectors $\vec{k}$. Quantum diffusion depends on the energy of these modes according to

$U(\vec{k}, t) = \exp(-iE(\vec{k})t)$. The lowest energy mode, $\vec{k} = 0$, corresponding to a uniform distribution, is an eigenstate of the diffusion operator and does not propagate. The slowest propagating modes are the ones with the smallest nonzero $|\vec{k}|$. The commonly used diffusion operator is the Laplacian, with $E(\vec{k}) \propto |\vec{k}|^2$. The alternative Dirac operator, available in a quantum setting, provides a faster diffusion of the slowest modes with $E(\vec{k}) \propto |\vec{k}|$. Its quadratically faster spread makes it better suited to spatial search algorithms [2, 3].

An automatic consequence of the Dirac operator is the appearance of an internal degree of freedom corresponding to spin, whereby the quantum state becomes a multi-component spinor. These spinor components can guide the diffusion process, and be interpreted as the states of a coin [2, 3]. While this is the only possibility for the continuum theory, another option is available for a lattice theory, i.e. the staggered fermion formalism [4]. In this approach, the spinor degrees of freedom are spread in coordinate space over an elementary hypercube, instead of being in an internal space. We follow it to construct a quantum search algorithm on a hypercubic lattice, reducing the total Hilbert space dimension by 2^d and eliminating the coin toss instruction.

A. Bipartite Decomposition

The free particle Dirac Hamiltonian in d dimensions is

$$H_{\text{free}} = -i\vec{\alpha} \cdot \vec{\nabla} + \beta m . \tag{1}$$

On a hypercubic lattice, with the simplest discretization of the gradient operator,

$$\nabla_n f(\vec{x}) = \frac{1}{2}[f(\vec{x} + \hat{n}) - f(\vec{x} - \hat{n})] , \tag{2}$$

and a convenient choice of basis, the anticommuting matrices $\vec{\alpha}, \beta$ are spin-diagonalized to location dependent

*Electronic address: adpatel@cts.iisc.ernet.in
†Electronic address: aminoorrahaman@yahoo.com

signs:

$$\psi \to T\psi, \ H \to THT^\dagger, \quad T = \alpha_d^{x_d}\alpha_{d-1}^{x_{d-1}} \cdots \alpha_2^{x_2}\alpha_1^{x_1}, \quad (3)$$

$$\alpha_n \to \prod_{j=1}^{n-1}(-1)^{x_j} \ , \quad \beta \to \prod_{j=1}^{d}(-1)^{x_j}\beta \ . \quad (4)$$

With this choice, translational invariance holds in steps of 2 (instead of 1), and the squared Hamiltonian,

$$H_{\text{free}}^2 = -\vec{\nabla} \cdot \vec{\nabla} + m^2 \ . \quad (5)$$

has eigenvalues $\sum_n \sin^2(k_n)+m^2$ on a hypercubic lattice. Inclusion of a mass term slows down the diffusion process, but it also regulates the $\vec{k} = 0$ mode. In the present article, we set $m = 0$; in an accompanying article [5], we investigate the advantage a nonzero mass offers when the problem has an infrared divergence.

Even when the Hamiltonian H is local, the discrete time evolution operator $U = \exp(-iH\tau)$ may connect arbitrarily separated vertices. To make U also local, we split H in to block-diagonal Hermitian parts and then exponentiate each part separately [6]. For a bipartite lattice, partitioning of H in to two parts (which we label "odd" and "even") is sufficient for this purpose [7]:

$$H_{\text{free}} = H_o + H_e \ . \quad (6)$$

This partition is illustrated in Fig.1 for $d = 1$ and $d = 2$. Each part contains all the vertices, but only half of the links attached to each vertex. Each link is associated with a term in H_{free} providing propagation along it, and appears in only one of the two parts. H thus gets divided in to a set of non-overlapping blocks of size $2^d \times 2^d$ (each block corresponds to an elementary hypercube on the lattice) that can be exactly exponentiated. The amplitude distribution then evolves according to

$$\psi(\vec{x};t) = W^t\psi(\vec{x};0) \ , \quad W = U_eU_o = e^{-iH_e\tau}e^{-iH_o\tau} \ . \quad (7)$$
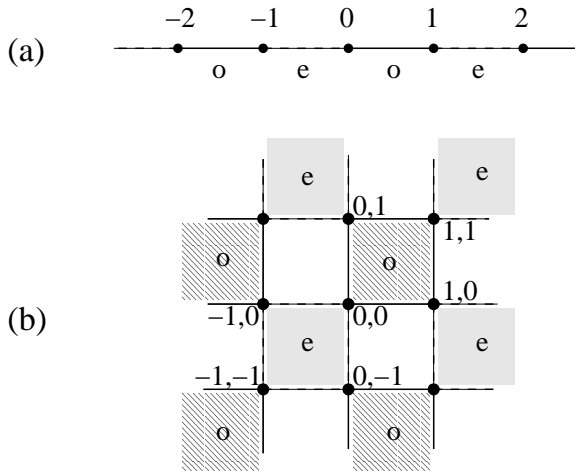

FIG. 1: Bipartite decomposition of the hypercubic lattice in to odd and even parts: (a) for $d = 1$, (b) for $d = 2$.

Each block of the unitary matrices $U_{o(e)}$ mixes amplitudes of vertices belonging to a single elementary hypercube, and the amplitude distribution spreads in time because the two alternating matrices do not commute. Note that W does not perform evolution according to H_{free} exactly. Instead, $W = \exp(-iH_{\text{free}}\tau) + O(\tau^2)$. But the truncation is such that W is exactly unitary, i.e. $W = \exp(-i\tilde{H}\tau)$ for some $\tilde{H}$.

We adopt the convention where the walk operator W is real. For $d = 1$, we set

$$H_o|x\rangle = -\frac{i}{2}\Big[(-1)^x|x + (-1)^x\rangle\Big] \ , \quad (8)$$

$$H_e|x\rangle = \frac{i}{2}\Big[(-1)^x|x - (-1)^x\rangle\Big] \ . \quad (9)$$

The resultant Hamiltonian blocks are $H_o^B = -\sigma_2/2$, and $H_e^B = \sigma_2/2$ when operating on the block with coordinates flipped in sign. With $(H_o^B)^2 = (H_e^B)^2 = \frac{1}{4}I$,

$$U_{o(e)} = cI - 2isH_{o(e)} \ , \quad |c|^2 + |s|^2 = 1 \ . \quad (10)$$

where $s = \sin(\tau/2)$ is a parameter that can be tuned.

These expressions can be rearranged to separate the translationally invariant part of the walk from the location dependent signs. Let the amplitude distribution in a two spinor (or coin) component notation be

$$\Psi(X,t) \equiv \begin{pmatrix} \psi(2x,t) \\ \psi(2x - 1,t) \end{pmatrix} \ . \quad (11)$$

Then the walk operator becomes

$$W = \tilde{U}C \ , \quad C = U_o^B = \begin{pmatrix} c & s \\ -s & c \end{pmatrix} \ , \quad (12)$$

$$\tilde{U}|X\rangle = c|X\rangle - s\sum_{\pm}(\pm\sigma_\pm)|X \pm 1\rangle \ . \quad (13)$$

The operator $C = U_o^B$ just mixes the components of Ψ within the block. The first term of the operator $\tilde{U}$ is stationary, while the other moves the amplitude to neighbouring blocks. The movement is accompanied by chirality flip, a characteristic feature of the Dirac operator, because of the raising and lowering Pauli operators $\sigma_\pm = (\sigma_1 \pm i\sigma_2)/2$.

In d dimensions, the above structure generalises to $2^d \times 2^d$ Hamiltonian blocks that can be expressed in terms of tensor products of Pauli matrices [8]. When operating on hypercubes with coordinate labels $\{0,1\}^{\otimes d}$,

$$H_o^B = -\frac{1}{2}\sum_{j=1}^{d} I^{\otimes(d-j)} \otimes \sigma_2 \otimes \sigma_3^{\otimes(j-1)} \ , \quad (14)$$

and $H_e^B = -H_o^B$ when operating on a hypercube with all coordinates flipped in sign. Note that Eq.(14) is the sum of d terms, which mutually anticommute and whose squares are proportional to identity. This is the characteristic signature of the Dirac Hamiltonian, Eq.(1). The

block-diagonal matrices satisfy $H_o^2 = H_e^2 = \frac{d}{4}I$, and exponentiate to

$$U_{o(e)} = cI - is\frac{2}{\sqrt{d}}H_{o(e)} \ , \ \ |c|^2 + |s|^2 = 1 \ . \qquad (15)$$

The parameter $s = \sin(\sqrt{d}\tau/2)$ can be optimised, to achieve the fastest diffusion across the lattice. Note that the d-dimensional walk is not the tensor production of d one-dimensional walks.

Again separating the degrees of freedom within each local hypercube as $\vec{x} \equiv \vec{X} \otimes \{0,\text{-}1\}^{\otimes d}$, the location dependent signs can be rewritten as operators acting on the spinor (or coin) degrees of freedom. The walk operator takes the form: $W = \tilde{U}C$, $C = U_o^B$ and

$$\tilde{U}|\vec{X}\rangle \ = \ c|\vec{X}\rangle \qquad\qquad\qquad\qquad (16)$$
$$- \ \frac{s}{\sqrt{d}}\sum_{\pm}\sum_{j=1}^{d}(I^{\otimes(j-1)} \otimes (\pm\sigma_{\pm}) \otimes \sigma_3^{\otimes(d-j)})|\vec{X} \pm \hat{j}\rangle \ .$$

B. Search Algorithm

To search for a marked vertex, say the origin, we attract the quantum random walk towards it by adding a potential to the free Hamiltonian,

$$V = \begin{cases} \beta V_0 \ \delta_{\vec{x},0} & \text{(gravitational)} \ , \\ V_0 \ \delta_{\vec{x},0} & \text{(electromagnetic)} \ . \end{cases} \qquad (17)$$

Exponentiation of this potential produces a phase change for the amplitude at the marked vertex. For the steepest attraction of the quantum random walk towards the marked vertex, it is optimal to choose V_0 so as to make the corresponding evolution phase is maximally different from 1, i.e. $e^{-iV_0\tau} = -1$. The sign provided by β does not matter in this case, and the evolution phase becomes the binary oracle,

$$R = I - 2|\vec{0}\rangle\langle\vec{0}| \ . \qquad (18)$$

The search algorithm alternates between diffusion and oracle operators—a discrete version of Trotter's formula involving H_o, H_e and V—yielding the evolution

$$\psi(\vec{x};t_1,t_2) = [W^{t_1}R]^{t_2}\psi(\vec{x};0,0) \ . \qquad (19)$$

Here t_2 is the number of search oracle queries, and t_1 is the number of walk steps between queries. Both should be minimised to find the the quickest solution to the search problem.

Since this algorithm iterates a unitary operator on the initial state, it produces periodic results. As a function of the number of iterations, the probability of being at the marked vertex first increases, reaches a peak value P after t_2 search oracle queries, then decreases down towards zero, and thereafter keeps on repeating this cycle. Grover's algorithm is designed to evolve the quantum state in a two-dimensional Hilbert space (spanned by the initial and marked states), and is able to reach $P = 1$. That does not hold for spatial search, and the resultant values of P are less than 1. The remedy is to augment the algorithm by the amplitude amplification procedure [9], to find the marked vertex with $\Theta(1)$ probability. The complexity of the algorithm is then characterised by the effective number of search oracle queries, $t_2/\sqrt{P}$.

C. Complexity Bounds

Spatial search obeys two simple lower bounds. One arises from the fact that while the marked vertex could be anywhere on the lattice, one step of the local walk can move from any vertex to only its nearest neighbours. Since the marked vertex cannot be located without reaching it, the worst case scenario on a hypercubic lattice requires $\Omega(dL)$ steps. This bound is the strongest in one dimension, where quantum search is unable to improve on the $O(N)$ classical search.

The other bound follows from the fact that spatial search cannot outperform Grover's optimal algorithm, and so it must require $\Omega(\sqrt{N})$ oracle queries. That is independent of d, and stronger than the first bound for $d > 2$. As a matter of fact, the first bound weakens with increasing d, as the number of neighbours of a vertex increase and steps required to go from one corner of the lattice to another decrease. In analogy with mean field theory behaviour of problems in statistical mechanics, we can thus expect that in the $d \to \infty$ limit the locality restriction on the walk would become irrelevant and the complexity of spatial search would approach that of Grover's algorithm. (Note that the maximum value of d is $\log_2 N$ for finite N.) Our results in this work demonstrate that not only is the $\Omega(\sqrt{N})$ complexity scaling achievable for $d > 2$, but one can get pretty close to the asymptotic prefactor $\pi/4$ as well.

Combined together, the two bounds make the complexity of spatial search $\Omega(dN^{1/d}, \sqrt{N})$. The two bounds are of the same magnitude, $\Omega(\sqrt{N})$, for the critical case of $d = 2$. It is a familiar occurrence in statistical mechanics of critical phenomena that interplay of different dynamical features produces logarithmic correction factors in critical dimensions due to infrared divergences. The same seems to be true for spatial search in $d = 2$, where algorithms are slowed down by extra logarithmic factors [2, 3, 10]. It is an open question to design algorithms, perhaps using additional parameters, that suppress the logarithmic factors as much as possible. We look at the special $d = 2$ case in an accompanying article.

The way the lower bounds arise in spatial search also illustrates an interplay of two distinct physical principles, special relativity (or no faster than light signaling) and unitarity. It is well-known that the two are compatible, but just barely so, in the sense that physical theories that are unitary but not relativistic or relativistic but not unitary exist. In spatial search, locality of the quan-

tum walk and the square-root speed up produced by the Dirac equation (through change in the dispersion relation) follow from special relativity. On the other hand, optimality of square-root speed up provided by Grover's algorithm is a consequence of unitarity [11, 12]. Why these two distinct physical principles lead to the same scaling constraint is not understood, and is a really interesting question to explore. At present, however, we have nothing more to add.

We point out that preparation of the unbiased initial state for the search problem, i.e. the translationally invariant uniform superposition state $|s\rangle = \sum_x |\vec{x}\rangle/\sqrt{N}$, is not at all difficult using local directed walk steps. For instance, one can start at the origin, step by step transfer the amplitude to the next vertex along an axis, and achieve an amplitude $L^{-1/2}$ at all the vertices on the axis after L steps. Repeating the procedure for each remaining coordinate direction produces the state $|s\rangle$ after dL steps in total. Clearly, this preparation does not add to the complexity of the algorithm.

II. OPTIMISATION OF PARAMETERS

The fastest search amounts to finding the shortest unitary evolution path between the initial state $|s\rangle$ and the marked state $|\vec{0}\rangle$. This path is a geodesic arc on the unitary sphere from $|s\rangle$ to $|\vec{0}\rangle$, and Grover's optimal algorithm strides along this path perfectly. In our algorithm, Eq.(19), we have replaced Grover's optimal diffusion operator, $G = 2|s\rangle\langle s| - 1$, by the walk operator W^{t_1}. So to optimise our algorithm, we need to tune the parameters appearing in W^{t_1}, i.e. s (or c) and t_1, such that $W^{t_1}R$ approximates a rotation in the two-dimensional $|s\rangle$-$|\vec{0}\rangle$ subspace by the largest possible angle.

The operator G has $|s\rangle$ as an eigenstate with eigenvalue 1. All other states orthogonal to $|s\rangle$, in particular the combination $|s_\perp\rangle = (|s\rangle - \sqrt{N}|\vec{0}\rangle)/\sqrt{N-1}$ in the $|s\rangle$-$|\vec{0}\rangle$ subspace, are its eigenstates with eigenvalue -1. The walk operator $W = U_e U_o$ also has $|s\rangle$ as an eigenstate with eigenvalue 1. But its other eigenvalues are spread around the unit circle, and evolution under $W^{t_1}R$ does not remain fully confined to the $|s\rangle$-$|\vec{0}\rangle$ subspace. If the outward diffusion is not controlled, the probability of remaining in the $|s\rangle$-$|\vec{0}\rangle$ subspace would decay exponentially with the number of iterations. To reach the marked state with a constant probability, therefore, parameters must be tuned so that part of the outward diffusion subsequently returns to the $|s\rangle$-$|\vec{0}\rangle$ subspace.

A. Analytical Criteria

The optimisation condition that makes W^{t_1} as close an approximation to G as possible can be formulated in several different ways. Though they all lead to the same result, describing them separately is instructive.

(1) Maximise the overlap of W^{t_1} and G, i.e.

$$Tr(W^{t_1}G) = Tr(2|s\rangle\langle s| - W^{t_1}) = 2 - \sum_i \langle i|W^{t_1}|i\rangle . \quad (20)$$

$\langle i|W^{t_1}|i\rangle$ is the amplitude for the random walk to return to the starting state $|i\rangle$ after t_1 time steps. Since W is translationally invariant along each coordinate direction in steps of 2, we need to evaluate $\langle i|W^{t_1}|i\rangle$ only for states $|i\rangle$ in an elementary hypercube. Now, any closed path on a hypercubic lattice takes an even number of steps along each coordinate direction, and all location dependent propagation signs (cf. Eq.(4)) get squared to 1 in the process. As a result, $\langle i|W^{t_1}|i\rangle$ is independent of $|i\rangle$, and the optimisation condition becomes minimisation of $A(t_1) \equiv \langle \vec{0}|W^{t_1}|\vec{0}\rangle$.

(2) Make $|s_\perp\rangle$ approximate an eigenvector of W^{t_1}, with an eigenvalue approaching -1, i.e. minimise

$$\langle s_\perp|W^{t_1}|s_\perp\rangle = \frac{1}{N-1}\left(-1 + NA(t_1)\right) , \quad (21)$$

which happens to be the same condition as above. In addition, unitarity of the evolution operator implies

$$|\langle s_\perp|W^{t_1}|s_\perp\rangle| \leq 1 \implies -1 + \frac{2}{N} \leq A(t_1) \leq 1 . \quad (22)$$

(3) Maximise the rotation provided by the operator $W^{t_1}R$ along the $|s\rangle$-$|\vec{0}\rangle$ subspace. This rotation is given by the projection

$$\begin{aligned} \mathrm{Proj}_{s,s_\perp}[W^{t_1}R] &= \begin{pmatrix} \langle s|W^{t_1}R|s\rangle & \langle s|W^{t_1}R|s_\perp\rangle \\ \langle s_\perp|W^{t_1}R|s\rangle & \langle s_\perp|W^{t_1}R|s_\perp\rangle \end{pmatrix} \\ &= \begin{pmatrix} 1 - \frac{2}{N} & \frac{2}{N}\sqrt{N-1} \\ \frac{2}{\sqrt{N-1}}(A(t_1) - \frac{1}{N}) & (1 - \frac{2}{N})\frac{1-NA(t_1)}{N-1} \end{pmatrix} \\ &= \begin{pmatrix} 1 & 0 \\ 0 & \frac{1-NA(t_1)}{N-1} \end{pmatrix}\begin{pmatrix} 1 - \frac{2}{N} & \frac{2}{N}\sqrt{N-1} \\ -\frac{2}{N}\sqrt{N-1} & 1 - \frac{2}{N} \end{pmatrix} . \end{aligned} \quad (23)$$

The factor on the right above is precisely the rotation matrix produced by the operator GR in Grover's algorithm, i.e. rotation by an angle $2\sin^{-1}(1/\sqrt{N})$ in the $|s\rangle$-$|\vec{0}\rangle$ subspace. So the total projection can be viewed as first applying an iteration of Grover's algorithm to the state, and then shrinking the $|s_\perp\rangle$ component by a factor $(1 - NA(t_1))/(N-1)$. The shrinking disappears and the projection tends to an exact orthogonal transformation, as $A(t_1)$ approaches its lower bound $(2 - N)/N$. It also follows that when the shrinking is controlled, cutting down outward diffusion from the $|s\rangle$-$|\vec{0}\rangle$ subspace, the algorithm will have the same $O(\sqrt{N})$ scaling as in case of Grover's algorithm.

The minimisation condition for $A(t_1)$ is obtained above only using translational invariance of the walk operator W, together with $W|s\rangle = |s\rangle$ and $\langle s|W = \langle s|$. It is easy to perform this optimisation numerically, as a function of the parameters s and t_1, and our results are described later. Before that, an understanding of the eigenspectrum of W gives us some idea regarding what to expect.

Since $H_o^2 = H_e^2 = \frac{d}{4}I$, eigenvalues of H_o and H_e are $\pm\sqrt{d}/2$. Moreover, H_o and H_e are traceless, so that half the eigenvalues are $\sqrt{d}/2$ and the other half $-\sqrt{d}/2$. Consequently, eigenvalues of U_o and U_e are $c \pm is = \exp(\pm i\sqrt{d}\tau/2)$, equally divided among complex conjugate pairs. Furthermore, $H_{o(e)}$ and $U_{o(e)}$ have the same eigenvectors.

Since U_o and U_e do not commute, we do not have a general solution for the eigenspectrum of W. Still it is straightforward to work it out in the continuum time limit, $\tau \to 0$. H_{free} is diagonalised by a Fourier transform:

$$
\begin{aligned}
H_{\text{free}}^2|\vec{x}\rangle &= -\frac{1}{4}\sum_{n=1}^{d}\left[|\vec{x}+2\hat{n}\rangle - 2|\vec{x}\rangle + |\vec{x}-2\hat{n}\rangle\right] \\
&\longrightarrow \sum_{n=1}^{d}\sin^2(k_n)|\vec{k}\rangle ,
\end{aligned}
\tag{24}
$$

with a uniform distribution in the Brillouin zone. Up to $O(\tau^2)$, therefore, the eigenvalues of $W(\approx e^{-iH_{\text{free}}\tau})$ are $\exp(\pm i\tau\sqrt{\sum_j \sin^2 k_j})$, with the eigenphases in the interval $[-\tau\sqrt{d}, \tau\sqrt{d}]$. The average value of H_{free}^2 is $d/2$, and the corresponding eigenvalues of W are $\exp(\pm i\tau\sqrt{d/2})$. We find that this average value plays an important role in determination of the optimal parameters.

B. Numerical Optimisation

The considerations of the previous subsection focused on what happens during a single iteration of the algorithm. Over many iterations, there is outward diffusion of the amplitude distribution from the $|s\rangle$-$|\vec{0}\rangle$ subspace, and subsequent partial return. That plays an important role in the overall success of the algorithm, but we have not been able to estimate it analytically. To study the performance of the whole algorithm, therefore, we numerically tune the parameters s and t_1 to (a) maximise the probability of reaching the marked vertex, and (b) minimise the number of search oracle queries required to reach that stage. The ensuing results of our optimisation of P, t_2 and $A(t_1)$ are described in what follows.

We first looked at how the probability distribution evolves as a function of time during the search process. To illustrate our results, we take $s = c = 1/\sqrt{2}$ that corresponds to maximal mixing between even and odd vertices, and analyse the evolution as a function of t_1 and t_2 on a 32^3 lattice. As displayed in Fig.2, the probability at the marked vertex undergoes a cyclic evolution pattern as a function of time t_2. We observe that the evolution is essentially sinusoidal for $t_1 = 1, 2, 3$, and persists for more than 30 cycles without any visible deviation. This behaviour implies that a constant fraction of the quantum state remains in the $|s\rangle$-$|\vec{0}\rangle$ subspace, and the operator $W^{t_1}R$ rotates it at a uniform rate within the subspace, fully matching the exact solution of Grover's

algorithm. The difference among $t_1 = 1, 2, 3$ is that the cycle time and the peak probability vary, meaning that the fraction of the quantum state present in the $|s\rangle$-$|\vec{0}\rangle$ subspace depends on t_1.

For $t_1 > 3$, we find that the evolution rapidly loses its sinusoidal nature and the peak probability plummets. The change in the pattern is so drastic that the algorithm no longer performs a reasonable search. This behaviour implies that most of the quantum state has moved out of the $|s\rangle$-$|\vec{0}\rangle$ subspace and only a negligible component is left behind. An appropriate choice of t_1 is thus crucial for constructing a successful search algorithm.

We also show in Fig.3 what the probability distribution looks like at the instance when the probability at the marked vertex attains its peak value. We note that the distribution is strongly peaked around the marked vertex in an essentially uniform background. The peak
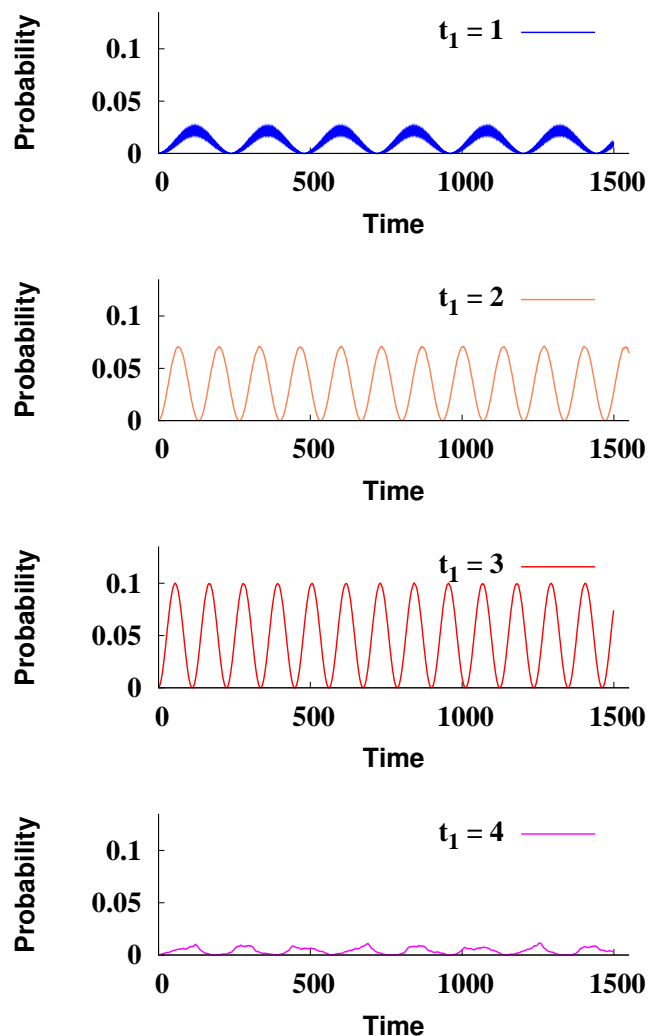

FIG. 2: Time evolution of the probability at the marked vertex for the spatial search problem on a 32^3 lattice. $s = 1/\sqrt{2}$ and t_1 increases from 1 (top) to 4 (bottom).

is sharper when the peak probability is larger, and it almost disappears for $t_1 > 3$. This pattern suggests that the part of the quantum state that diffuses outside the $|s\rangle$-$|\vec{0}\rangle$ subspace is almost featureless.

Combining our observations in this particular example, we find that the best results—the shortest period, the largest peak probability and the sharpest peak—clearly correspond to $t_1 = 3$. The results improve as t_1 increases from 1 to 3, and rapidly deteriorate thereafter. The important fact that all optimal features appear simultaneously at $t_1 = 3$ simplifies tuning of the parameters.

Next we scanned through values of s, for different values of t_1 and in different dimensions, to determine the optimal values at which the peak probability P at the marked vertex is the largest and the time t_2 required to reach it is the smallest. A typical situation, with $t_1 = 3$ on a 32^3 lattice, is shown in Fig.4. We notice that continuous P has a smoother behaviour than discrete t_2, and maximisation of P and minimisation of t_2 occur at roughly the same value of s. So we simplify matters, and select maximisation of P as our optimality condition.

A comparison of our scans, for $t_1 = 3$ and different number of dimensions, is presented in Fig.5. We find that at fixed t_1, the shape of the P vs. s curve is almost dimension independent, and the optimal value of s is approximately the same for all dimensions. We also observe that for fixed s, P roughly scales as 2^{-d}. In the staggered fermion implementation of the Dirac operator, different vertices of an elementary hypercube correspond to different degrees of freedom [4]. So our observation suggests that to a large extent only the degree of freedom corresponding to the marked vertex evolves during the search, while other degrees of freedom remain spectators. Moreover, the feature that the maximum value of P approaches 2^{-d} with increasing d implies that the efficiency of our algorithm increases with d.

All our quantitative results for optimal values of s are collected in Table I. We note that the best performance
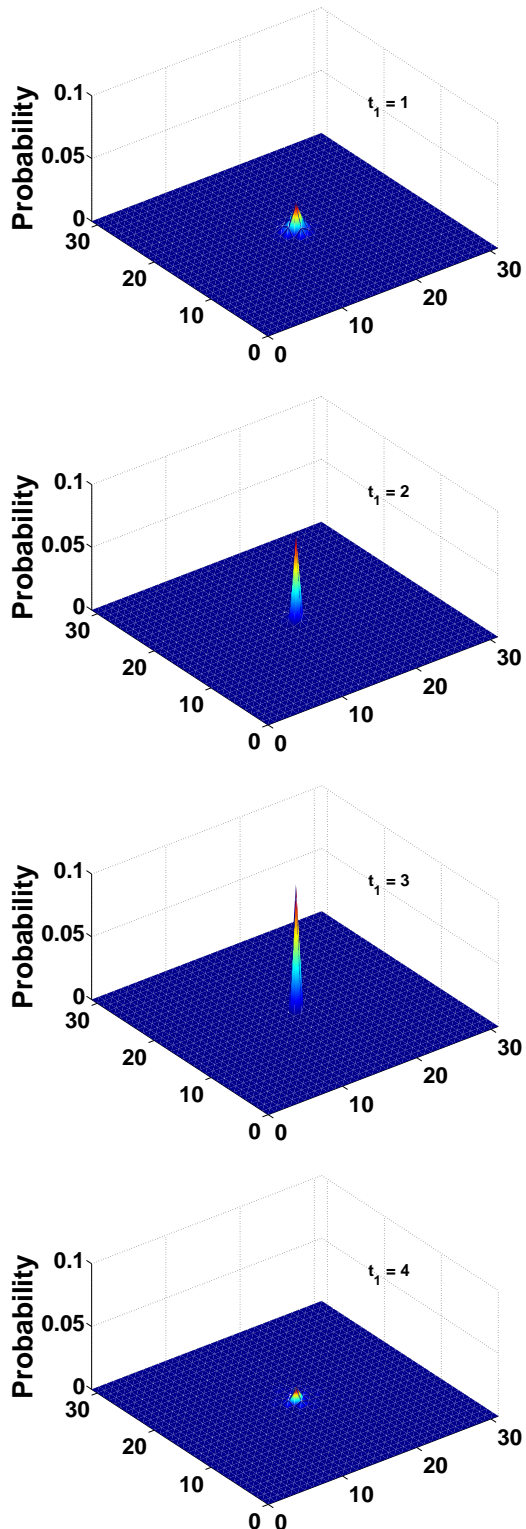

FIG. 3: Probability distribution for the spatial search problem on a 32^3 lattice, at the instance when the probability at the marked vertex $(16, 16, 16)$ attains its peak value. The displayed results are for the $z = 16$ slice. $s = 1/\sqrt{2}$ and t_1 increases from 1 (top) to 4 (bottom).
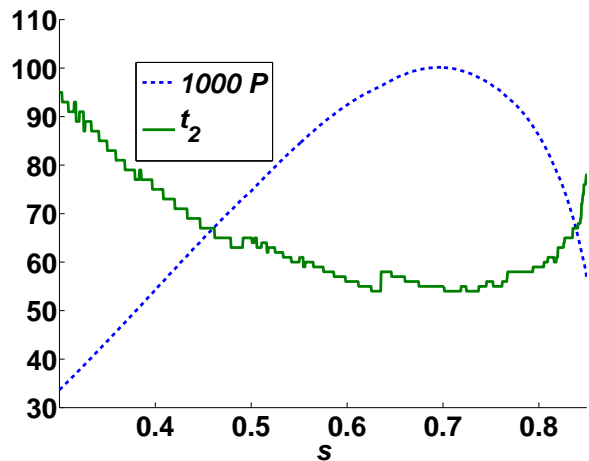

FIG. 4: Optimisation of s for $t_1 = 3$ on a 32^3 lattice.

TABLE I: Results of optimal parameter determination for different t_1 and in different dimensions.

d	L	t_1	Spatial Search				Walk		
			s	t_2	P	θ	s	$A(t_1)_{\min}$	θ
3	32	2	0.9507	59	0.0942	3.551	0.9258	-0.7143	3.346
		3	0.7015	55	0.1001	3.299	0.6737	-0.7618	3.136
		4	0.5363	55	0.1016	3.202	0.5194	-0.7748	3.089
		8	0.2755	54	0.1027	3.158	0.2665	-0.7860	3.052
		20	0.1114	54	0.1030	3.157	0.1074	-0.7890	3.044
4	16	2	0.9541	54	0.0528	3.583	0.9428	-0.7778	3.482
		3	0.6986	54	0.0548	3.281	0.6827	-0.8190	3.188
		4	0.5411	53	0.0553	3.234	0.5257	-0.8300	3.131
		8	0.2771	52	0.0558	3.177	0.2694	-0.8395	3.086
		20	0.1115	52	0.0559	3.160	0.1084	-0.8420	3.072
5	16	2	0.9500	150	0.0276	3.545	0.9535	-0.8182	3.577
		3	0.6920	148	0.0284	3.242	0.6880	-0.8541	3.219
		4	0.5376	147	0.0286	3.211	0.5292	-0.8636	3.155
		8	0.2726	147	0.0288	3.124	0.2710	-0.8718	3.105
		20	0.1108	147	0.0288	3.140	0.1092	-0.8739	3.095
6	8	3	0.6891	51	0.0145	3.225	0.6913	-0.8778	3.238
		20	0.1106	51	0.0147	3.135	0.1094	-0.8951	3.101
7	8	3	0.6932	102	0.0073	3.250	0.6937	-0.8949	3.252
		20	0.1097	102	0.0074	3.109	0.1098	-0.9102	3.112

parameters, P and t_2, depend on d but hardly on t_1 or s. (With increasing t_1, there is a slight increase in P and a small decrease in t_2, but that is a rather tiny improvement.) More interestingly, we find a relation between the optimal values of s and t_1, in terms of the variable

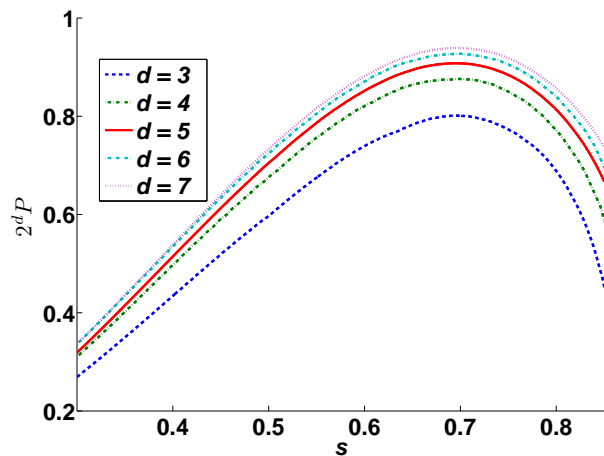$$\theta = \sqrt{2}t_1 \sin^{-1} s = t_1 \tau \sqrt{d/2} \ . \tag{25}$$

FIG. 5: Optimisation of s for $t_1 = 3$ in different dimensions. Lattice sizes $L = 32, 16, 16, 8, 8$ were used for $d = 3, 4, 5, 6, 7$ respectively.

Since $\sin^{-1} s$ describes the unitary rotation performed by the single step operators $U_{o(e)}$, and t_1 is the number of walk steps, θ is a measure of the unitary rotation performed by the operator W^{t_1}. The corresponding operator in Grover's optimal algorithm, G, is a reflection providing a phase change of π. We find that our optimal W^{t_1} is also close to a reflection, in a sense, with $\theta \approx \pi$. Unlike the discrete ± 1 eigenvalues of G though, eigenvalues of W^{t_1} form a distribution. We don't know this distribution in general, but we know it in the limit $\tau \to 0$, as described at the end of Subsection II.A. This limit corresponds to small s and large t_1, and Table I shows that θ indeed gets close to π in this limit. To be precise, this normalisation of θ implies that the optimal $W^{t_1} \equiv e^{-i\tilde{H}\tau t_1}$ is a reflection operator for the "average eigenvalue" of the associated $\tilde{H}^2$. Appearance of the average eigenvalue in the normalisation indicates that all spatial modes contribute to the search process with roughly equal strength.

Table I also lists our results for the variable $A(t_1)$ defined in Subsection II.A, which characterises how close W^{t_1} is to G. We computed $A(t_1)$, by starting with unit amplitude at the marked vertex, and then measuring the amplitude at the marked vertex after t_1 walk steps. At the beginning of this Subsection we mentioned that checking $A(t_1)$ is not as elaborate an optimisation test as using the full search algorithm. Nevertheless, the results corroborate our optimisation of spatial search: the minimum value of $A(t_1)$ is reasonably close to -1, and occurs at approximately the same value of s (and hence θ) that provides the optimal search (the mismatch decreases with increasing d). We also observe that $A(t_1)$ gets closer to -1, suggesting that the algorithm becomes more efficient, with increasing d as well as t_1.

Given the strong correlation between the optimal values of s and t_1, the computational cost of the algorithm is minimised by choosing the smallest t_1. For $t_1 = 1$, Eq.(25) does not have a solution with $\theta \approx \pi$, the optimal value of s is very close to 1, and our algorithm does not work well there. Then $t_1 = 2$ is the choice that minimises the number of walk steps in the algorithm. Even when the emphasis is on optimising P and t_2, most of the improvement can be obtained by going from $t_1 = 2$ to $t_1 = 3$, and we need not go beyond that.

III. SIMULATION RESULTS

We carried out numerical simulations of the quantum spatial search problem, with a single marked vertex and lattice dimension ranging from 3 to 9. The matrices $U_{o(e)}$ are real with our conventions, and that was convenient for simulations. The optimal values of s-t_1 pairs depend a little on the lattice size and dimension. We are only interested in the asymptotic behaviour of the algorithm, however, and so we used the same parameter values for all lattice sizes and dimensions: $s = 0.9539$ for $t_1 = 2$, $s = 1/\sqrt{2}$ for $t_1 = 3$ and $s = 0.5410$ for $t_1 = 4$.

To extract the scaling properties of the algorithm, we

TABLE II: Fit parameters for peak probability P and search oracle queries $t_2/\sqrt{N}$ vs. L.

s	t_1	d	L	a_1	b_1	Error	a_2	b_2	Error	$a_2/\sqrt{a_1}$
0.9539	2	3	64,128,256,512	0.0911	0.0964	2.43×10^{-5}	0.3237	0.1440	4.57×10^{-4}	1.072
		4	16,32,48,64	0.0522	0.0100	3.02×10^{-5}	0.2167	-0.0832	1.05×10^{-3}	0.948
		5	12,16,24,32	0.0275	0.0009	1.70×10^{-6}	0.1486	-0.0243	2.39×10^{-4}	0.896
		6	12,14,16,18	0.0141	0.0001	2.31×10^{-7}	0.1043	-0.0208	1.78×10^{-4}	0.878
		7	6,8,10	0.0072	-0.0004	1.40×10^{-6}	0.0757	-0.0338	2.47×10^{-4}	0.892
		8	6,8	0.0036	—	3.50×10^{-6}	0.0509	—	7.23×10^{-5}	0.848
		9	6	0.0018	—	—	0.0356	—	—	0.839
$1/\sqrt{2}$	3	3	64,128,256,512	0.0968	0.0920	2.73×10^{-6}	0.3141	-0.0306	1.23×10^{-3}	1.010
		4	16,32,48,64	0.0542	0.0076	1.46×10^{-5}	0.2097	0.0151	6.71×10^{-4}	0.901
		5	12,16,24,32	0.0283	0.0010	4.66×10^{-6}	0.1470	-0.0300	2.12×10^{-4}	0.874
		6	12,14,16,18	0.0145	0.0001	5.79×10^{-7}	0.1035	-0.0269	1.88×10^{-4}	0.860
		7	6,8,10	0.0074	-0.0003	1.46×10^{-6}	0.0750	-0.0296	3.39×10^{-4}	0.872
		8	6,8	0.0037	—	3.00×10^{-6}	0.0498	—	4.55×10^{-5}	0.819
		9	6	0.0019	—	—	0.0353	—	—	0.810
0.5410	4	3	64,128,256,512	0.0984	0.0936	1.84×10^{-5}	0.3123	-0.1239	8.46×10^{-4}	0.996
		4	16,32,48,64	0.0548	0.0087	2.24×10^{-5}	0.2103	-0.0500	3.57×10^{-4}	0.898
		5	12,16,24,32	0.0285	0.0013	8.05×10^{-6}	0.1455	-0.0142	2.25×10^{-4}	0.862
		6	12,14,16,18	0.0146	0.0002	7.67×10^{-7}	0.1015	0.0043	6.28×10^{-5}	0.840
		7	6,8,10	0.0074	-0.0003	2.86×10^{-6}	0.0733	-0.0194	2.01×10^{-4}	0.852
		8	6,8	0.0037	—	4.50×10^{-6}	0.0505	—	4.39×10^{-4}	0.830
		9	6	0.0019	—	—	0.0353	—	—	0.810

studied behaviour of P and t_2, as a function of L and d. We found that the finite size effects in our data, as $L \to \infty$ at fixed d, are well described by series in inverse powers of L. In Figs.6-7, we show some of our results,

with the fitting functions:

$$P = a_1 + \frac{b_1}{L} \ , \quad \frac{t_2}{\sqrt{N}} = a_2 + \frac{b_2}{L} \ . \tag{26}$$

All the fit parameters are listed in Table II, where error refers to the r.m.s. deviation of the data from the fit. We
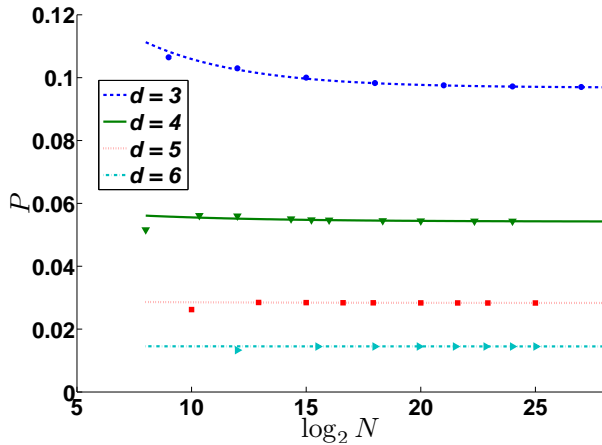

FIG. 6: The peak probability at the marked vertex as a function of database size (ranging from 2^8 to 2^{27}) in different dimensions. The points are the data from the simulations with $t_1 = 3$, and the curves are the fits $P = a_1 + (b_1/L)$.
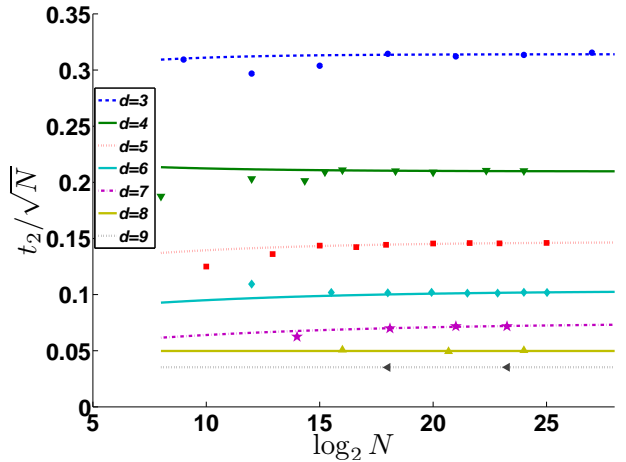

FIG. 7: The number of search oracle queries as a function of database size (ranging from 2^8 to 2^{27}) in different dimensions. The points are the data from the simulations with $t_1 = 3$, and the curves are the fits $(t_2/\sqrt{N}) = a_2 + (b_2/L)$.

do not have sufficient data for $d = 8$ and $d = 9$, to make accurate fits or to see the asymptotic behaviour. Our analysis, therefore, relies on the patterns seen for $d = 3$ to $d = 7$, while we use $d = 8$ and $d = 9$ values only for consistency checks.

We observe the following features:

(1) There is not much difference among the a_1 and a_2 values for different optimised s-t_1 pairs. This confirms our earlier inference that implementing our algorithm for $t_1 = 2$ minimises the running cost.

(2) P approaches a constant as $L \to \infty$, with a_1 decreasing approximately as 2^{-d} with increasing d. As mentioned before, this is a consequence of the fact that in the staggered fermion implementation of the Dirac operator, only the degree of freedom corresponding to the location of the marked vertex on the elementary hypercube participates in the search process. Since the number of vertices in an elementary hypercube is 2^d, P is upper bounded by 2^{-d}. Our algorithm achieves that bound up to a constant multiple close to 1.

(3) b_1 decreases towards zero with increasing d, which is also true for $|b_2|$ to a large extent. Thus the smallest $d = 3$ has the largest finite size corrections to the asymptotic $L \to \infty$ behaviour. This suggests that as the number of neighbours of a vertex increase, the finite size of the lattice becomes less and less relevant.

(4) t_2 is proportional to $\sqrt{N}$ as $L \to \infty$, with a_2 decreasing approximately as $2^{-d/2}$ with increasing d. Thus t_2 scales with the database size as $(L/2)^{d/2} = \sqrt{N/2^d}$. This is in agreement with the fact that, with only one degree of freedom per elementary hypercube participating in the search process, the effective number of vertices being searched is $N/2^d$.

(5) In the combination $t_2/\sqrt{P}$, describing the effective number of oracle queries for our algorithm, the factors of 2^d arising from the number of vertices in an elementary hypercube fully cancel. That makes $t_2/\sqrt{P}$ scale as $\sqrt{N}$, and $a_2/\sqrt{a_1}$ describes the complexity scaling of our algorithm. As listed in the last column of Table II, $a_2/\sqrt{a_1}$ shows a gradual decrease with increasing d, and our results are just above the corresponding value $\pi/4$ for Grover's optimal algorithm. More explicitly, our result for $d = 3$ is about 25% above $\pi/4$, decreasing to about 10% above $\pi/4$ for $d = 7$. These values imply that our algorithm improves in efficiency with increasing d, and is not too far from the optimal behaviour even for the smallest dimension $d = 3$.

To analyse the dimension dependence of our algorithm in more detail, we fit the values in Table II to the functional forms,

$$\log_2 a_1 = c_1 + d_1 d , \quad \log_2 a_2 = c_2 + d_2 d , \tag{27}$$

$$\frac{a_2}{\sqrt{a_1}} = c_3 + \frac{d_3}{d} . \tag{28}$$

The results for $t_1 = 3$ are displayed in Fig.8, showing that the fits work reasonably well.

The fit parameters for the three different optimised s-t_1 pairs are listed in Table III. They support our earlier scaling observations: d_1 close to -1 means that P scales roughly as 2^{-d}, and d_2 close to -0.5 means that t_2 scales roughly as $\sqrt{N/2^d}$. Interpreting Grover's algorithm as the $d \to \infty$ limit of the spatial search algorithm, we expect c_3 to be close to $\pi/4$. It turns out to be somewhat smaller, but is not very accurately determined due to sizable correction from d_3 for the values of d that we have. We believe that simulations for larger lattice sizes and higher dimensions can remedy this problem.

To look at our data in a different manner, we analyse the scaling behaviour of $t_2/\sqrt{NP}$ as a function of the
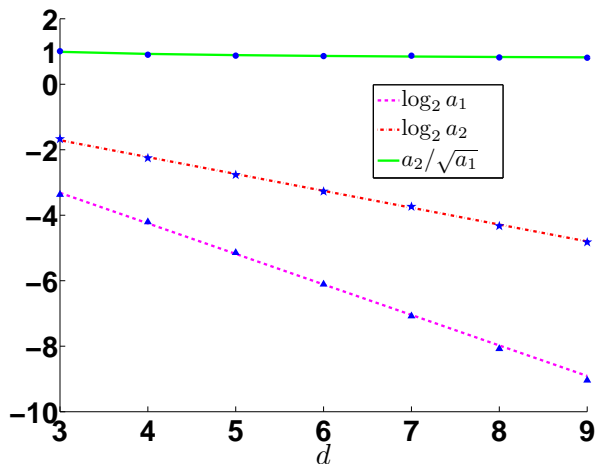

FIG. 8: Behaviour of the asymptotic fit parameters as a function of dimension. The data points are from Table II, with $t_1 = 3$, and the curves are the fits described in Eqs.(27,28).
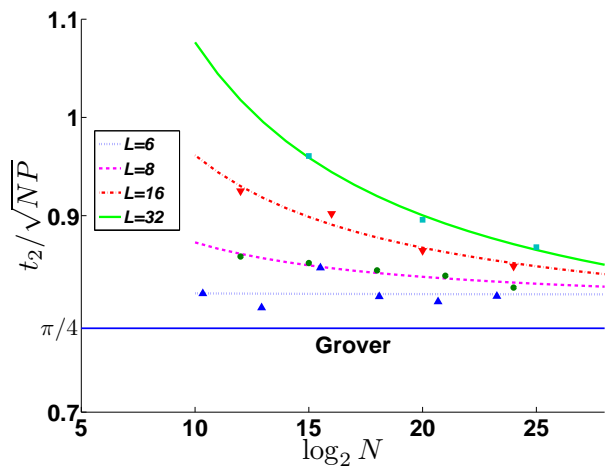

FIG. 9: $t_2/\sqrt{NP}$ as a function of the database size for different lattice sizes. The points are the data for $t_1 = 3$, and the curves are the fits $(t_2/\sqrt{NP}) = a_l + (b_l/d)$. Also shown is the limiting value corresponding to Grover's optimal algorithm.

TABLE III: Fit parameters for $\log_2 a_1$, $\log_2 a_2$ and $a_2/\sqrt{a_1}$ vs. d.

s	t_1	Fitted d	c_1	d_1	Error	c_2	d_2	Error	c_3	d_3	Error
0.9539	2		-0.553	-0.938	4.77×10^{-2}	-0.085	-0.526	2.53×10^{-2}	0.721	0.995	1.71×10^{-2}
$1/\sqrt{2}$	3	3,4,5,6,7,8	-0.458	-0.947	4.48×10^{-2}	-0.138	-0.521	2.57×10^{-2}	0.729	0.791	1.63×10^{-2}
0.5410	4		-0.421	-0.951	3.78×10^{-2}	-0.151	-0.521	2.09×10^{-2}	0.725	0.761	1.32×10^{-2}

database size. In Fig.9, we plot a subset of the data, illustrating how $t_2/\sqrt{NP}$ behaves for fixed lattice size L when the database size is changed by changing d. The approach to the asymptotic behaviour is particularly clear in this form, with $t_2/\sqrt{NP}$ decreasing towards $\pi/4$ as the database size increases. It is also obvious that, for fixed N, our algorithm improves in efficiency with decreasing L (or equivalently with increasing d). This is a consequence of the increase in the number of neighbours of a vertex with increasing d. An important implication is that, given a database of size N, it is best to implement the algorithm using the smallest L (and hence the largest d) available. For our staggered fermion implementation of the Dirac operator, L has to be even, and the smallest L possible is 4. Because of small values of d that we have worked with, and discrete nature of t_2, our data for $L = 4$ do not show smooth asymptotic behaviour. We therefore omitted $L = 4$ from our analysis, and our best results are for $L = 6$.

We can fit the data reasonably well as

$$\frac{t_2}{\sqrt{NP}} = a_l + \frac{b_l}{d} \ . \tag{29}$$

The fit parameters for the three different optimised s-t_1 pairs are listed in Table IV. We note that a_l is close to $\pi/4$ when data for $d > 6$ are available, but it cannot be accurately determined using data for small d only due to sizable corrections from b_l.

TABLE IV: Fit parameters for $t_2/\sqrt{NP}$ vs. d at fixed L.

s	t_1	L	Fitted d	a_l	b_l	Error
0.9539	2	6	5,6,7,8,9	0.766	0.551	1.06×10^{-2}
		8	4,5,6,7,8	0.805	0.293	5.31×10^{-4}
		16	3,4,5,6	0.663	1.132	1.73×10^{-2}
		32	3,4,5	0.625	1.304	5.77×10^{-3}
$1/\sqrt{2}$	3	6	4,5,6,7,8,9	0.819	0.006	1.28×10^{-2}
		8	4,5,6,7,8	0.803	0.234	3.28×10^{-3}
		16	3,4,5,6	0.773	0.471	6.46×10^{-3}
		32	3,4,5	0.724	0.705	3.23×10^{-3}
0.5410	4	6	4,5,6,7,8,9	0.837	-0.092	1.35×10^{-2}
		8	4,5,6,7,8	0.791	0.272	3.79×10^{-3}
		16	3,4,5,6	0.767	0.450	1.29×10^{-3}
		32	3,4,5	0.711	0.726	1.13×10^{-3}

IV. RESULTS FOR MULTIPLE MARKED VERTICES

We also carried out a few simulations of spatial search with more than one marked vertex. The change from the single marked vertex algorithm is that the search oracle now flips the sign of amplitudes at several distinct vertices, say M. To see clear patterns in the results, we needed to keep the marked vertices well separated and avoid interference effects.

With the staggered fermion implementation of the Dirac operator, it matters whether the marked vertices belong to the same corner of the elementary hypercube or to different corners. So we considered various hypercube locations for the marked vertices. In Table V, we compare our illustrative results for two and three marked vertices with those for a single marked vertex. The values for peak probability P and corresponding time t_2 were obtained with $s = 1/\sqrt{2}$ and $t_1 = 3$ on a 64^3 lattice.

When the marked vertices belong to the same corner of the elementary hypercube (e.g. (0,32,32) and (32,32,32)), they involve the same degree of freedom. In that case, we observe that P decreases by a factor of M and t_2 decreases by a factor of $\sqrt{M}$. On the other hand, when the marked vertices belong to different corners of the elementary hypercube (e.g. (0,32,33) and (32,32,32)), they involve different degrees of freedom. In that case, we find that P and t_2 are more or less the same as for the single marked vertex case. This pattern is also followed when some marked vertices are at the same corner of the elementary hypercube and some at different corners.

We can infer the following equipartition rule from these observations. A fixed peak probability (upper bounded by 2^{-d}) is available for each corner degree of freedom of the elementary hypercube. Multiple marked vertices corresponding to the same degree of freedom share this peak probability equally, while different degrees of freedom evolve independently of each other. Apart from this caveat, the dependence of t_2 on M for our spatial search algorithm is the same as in case of Grover's algorithm, i.e. proportional to $\sqrt{N/M}$.

V. SUMMARY

We have formulated the spatial search problem using staggered fermion discretisation of the Dirac operator on a hypercubic lattice in arbitrary dimension. Our search strategy, Eq.(19), mimics that of Grover's algo-

TABLE V: Spatial search results for multiple marked vertices in $d = 3$.

s	t_1	M	Marked vertices	P	t_2
$1/\sqrt{2}$	3	1	(32,32,32)	0.09829	161
		2	(0,32,32), (32,32,32)	0.04919, 0.04919	112, 112
			(0,32,33), (32,32,32)	0.09868, 0.09790	161, 161
		3	(0,0,0), (16,16,16), (32,32,32)	0.03530, 0.03264, 0.03082	94, 92, 92
			(0,0,1), (16,16,16), (32,32,32)	0.09380, 0.05590, 0.04507	161, 117, 109
			(0,0,1), (16,16,16), (33,32,32)	0.09298, 0.09838, 0.10347	157, 161, 162

rithm. The locality restriction on the walk operator becomes less and less relevant as the dimensionality of the space increases, and we expect to reach the optimal scaling behaviour of Grover's algorithm as $d \to \infty$.

We have verified our theoretical expectations by numerical simulations of the algorithm for $d = 3$ to $d = 9$. We clearly demonstrate the approach to the $d \to \infty$ limit, and find that even for $d = 3$ our algorithm is only 25% less efficient. From computational cost point of view, for a fixed database size N, our best parameters are $t_1 = 2$ and the largest d available. With the staggered fermion implementation, our algorithm works in total Hilbert space dimension N. That is a big reduction from the total Hilbert dimension $2^d N$ required by the algorithms that use coin or internal degrees of freedom (for the best $d = \Theta(\log N)$, $2^d = O(N)$ is substantial), and makes our exercise worthwhile.

[1] L.K. Grover, Pramana 56 (2001) 333, arXiv:quant-ph/0109116.
[2] A. Ambainis, J. Kempe and A. Rivosh, Proceedings of ACM-SIAM SODA'05 (ACM Press, New York, 2005), p. 1099, arXiv:quant-ph/0402107.
[3] A.M. Childs and J. Goldstone, Phys. Rev. A 70 (2004) 042312, arXiv:quant-ph/0405120.
[4] L. Susskind, Phys. Rev. D 16 (1977) 3031.
[5] A. Patel, K.S. Raghunathan and M.A. Rahaman, arXiv:1003.????.
[6] A. Patel, K.S. Raghunathan and P. Rungta, Phys. Rev. A71 (2005) 032347, arXiv:quant-ph/0405128.
[7] This is in spite of the fact that Dirac spinors require $2^{[(d+1)/2]}$ components in d dimensions. For example, in $d = 3$, a Rubik's cube can have the central cube labeled "o" and the eight corner cubes labeled "e".
[8] A. Patel, K.S. Raghunathan and P. Rungta, Proceedings of the Workshop on Quantum Information, Computation and Communication (QICC-2005), IIT Kharagpur, India, (Allied Publishers, 2006), p. 41, arXiv:quant-ph/0506221.
[9] G. Brassard, P. Hoyer, M. Mosca and A. Tapp, in *Quantum Computation and Information*, AMS Contemporary Mathematics Series Vol. 305, eds. S.J. Lomonaco and H.E. Brandt (AMS, Providence, 2002), p. 53, arXiv:quant-ph/0005055.
[10] A. Tulsi, Phys. Rev. A78 (2008) 012310, arXiv:0801.0497.
[11] C. Zalka, Phys. Rev. A60 (1999) 2746, arXiv:quant-ph/9711070.
[12] C. Bennett, E. Bernstein, G. Brassard and U. Vazirani, SIAM J. Comput. 26 (1997) 1510, arXiv:quant-ph/9701001.