EFFICIENT OCCUPATION COUNT FOR CERTAIN PROBABILISTIC FERMIONIC MOTION

ERIC PETERSON

ABSTRACT. Performing an occupation count over a spatial region for a system of particles is a very common statistical task—for instance, think of measuring changing atmospheric density. We outline a computational regime that simultaneously offers efficient occupation count as well as efficient system evolution for discretized quantum systems of fermionic particles.

1. CLASSICAL INTRODUCTION

Consider the following very generic situation: we have a (closed) system of (fermionic) particles moving around some spatial context, evolving subject to some physical law. A very common, very basic statistical question one might ask about this system is:

Question 1. How many particles lie within some prescribed region?

Example 2. Consider atmospheric particles, subject to mutual repulsion and to gravity. How dense is the atmosphere as you gain altitude from sealevel?

For our part, as computer scientists, we would like to simulate such systems on a computer and to measure these values through computation. There are two competing computational forces in this situation: *efficient simulation* requires us to be able to quickly *update* the state of the system, and *efficient counting* requires us to be able to quickly perform *occupation counts* over regions in the system. Wisdom in this trade means picking the right tool for the right job: we should explore different data structures in hopes that different ones have different associated access costs for these operations. For sanity's sake, we specialize to the situation of particles occupying positions on a line.

1.1. **Bitstrings.**

We begin by doing the most naive thing possible, with no clever algorithmic knowledge incorporated, by exactly representing the configuration of the particles as bits in a string. Specifically, the n^{th} bit indicates whether the n^{th} position on the line is occupied by a particle. With this representation, updating the state of the system amounts to toggling any bit, which is a single access operation, and so has access complexity of $O(1)$. On the other hand, calculating the number of occupied positions in the first $m \le n$ bits takes m accesses, which has average- and worst-case access complexitiy of $O(n)$.

1.2. **Occupancy counts (Jordan–Wigler).**

We can also do the opposite and precompute all of the occupancy counts for the initial segments of a list of particle occupation states. In this representation, computing the occupancy count of any given initial segment is now just a single lookup, so has access complexity of $O(1)$.[1] The trade is that modiying the configuration is now considerably more complicated: toggling the presence of a particle in the m^{th} position requires updating all of the occupancy counts at and above the m^{th} length, which again has average- and worst-case access complexity of $O(n)$.

1.3. **Fenwick trees.**

There is an intermediate data structure called a *Fenwick tree* that mediates between these extremes by storing *partial* occupancy counts, so that updates only touch some nodes of the tree and full occupany counts also require assembling data from only some of the nodes of the tree. Roughly speaking, the graph is divided into its bottom half, the bottom half of what remains, the bottom half of what remains, ..., and then the algorithm recurses. For instance, a graph with vertices $v_0, \ldots, v_{15}$ gets arranged into a tree as in Figure 1. At each node, we store the occupancy count of it together with all of its children. The key property of this layout of tree is that both its depth *and* its (slanted) width are logarithmic. These dictate the access complexities of the operations: updating the occupation of a given position affects the occupation count of all of the node's parents, and reads of occupany counts require sums across (slanted) branches. Hence, both have access complexity $O(\log n)$.

[1] Indeed, any connected segment requires just two lookups and a subtraction.

$$\begin{tikzcd}
15 \arrow[d] \arrow[drrrr, bend left] \arrow[drrrrrr, bend left] \arrow[drrrrrrr, bend left] & & & & & & & \\
7 \arrow[d] \arrow[drr] \arrow[drrr] & & & & 11 \arrow[d] \arrow[dr] & & 13 \arrow[d] & 14 \\
3 \arrow[d] \arrow[dr] & & 5 \arrow[d] & 6 & 9 \arrow[d] & 10 & 12 & \\
1 \arrow[d] & 2 & 4 & & 8 & & & \\
0 & & & & & & &
\end{tikzcd}$$

FIGURE 1. A Fenwick tree formed from $\{0, \ldots, 15\}$.

1.4. **Segmented Fenwick trees.** If updates are known to be in some sense local, it may be wise to partition the graph into subgraphs so that updating the occupancy count in one region does not affect the partial sums in the other regions at all. At one extreme end, where the graph is completely partitioned into singletons, this recovers the Jordan–Wigler scheme. At the other extreme, with no partitioning, this is identical to the naive Fenwick tree scheme.

2. QUBIT ENCODING OF FERMIONS ON A LINE

Remarkably, this kind of computation is *intrinsic* to simulating quantum systems of fermionic particles. Recall that a system of particles governed by fermionic statistics are controlled by creation operators $a_k^\dagger$ and annihilation operators a_k on a state space V, themselves subject to the following rules:

$$\{a_f, a_g\} = 0, \qquad \{a_f^\dagger, a_g^\dagger\} = 0, \qquad \{a_f, a_g^\dagger\} = \langle f | g \rangle,$$

where $\{r, s\} = rs + sr$ denotes the anticommutator of two operators r and s. Now suppose that we put ourselves into an identical physical situation to the classical exercise considered above: we consider configurations of fermionic particles slotted into n positions on a line. We can record any particular configuration as as a state vector $|f_n \cdots f_1\rangle$, and a particular choice of creation and annihilation operators in this situation is specified by the linear extension of the rules

$$a_j^\dagger \left| f_n \cdots f_1 \right\rangle = \begin{cases} (-1)^{f_{+<j}} \left| f_n \cdots f_{j+1} 1 f_{j-1} \cdots f_1 \right\rangle & \text{when } f_j = 0, \\ 0 & \text{when } f_j = 1, \end{cases}$$
$$a_j \left| f_n \cdots f_1 \right\rangle = \begin{cases} (-1)^{f_{+<j}} \left| f_n \cdots f_{j+1} 0 f_{j-1} \cdots f_1 \right\rangle & \text{when } f_j = 1, \\ 0 & \text{when } f_j = 0, \end{cases}$$

where we have introduced the signs in order to guarantee the anticommutation relations above.[2]

Our goal is to translate this physical system into something simulable by a quantum computer. The state vector we can translate directly into a qubit vector: measuring $|q_n \cdots q_1\rangle$ gives a classical bitvector recording the configuration of the fermionic particles as in our naive presentation. The creation and annihilation operators can be encoded in

[2]As an important theoretical remark, the *algebra* generated by these operators is invariant of any particular choice of presentation, despite the evident choices made in our signs, which depend on a choice of ordering of the states.

terms of qubit gates as follows:

$$Q_j^\dagger = |1\rangle_j \langle 0|_j = \frac{1}{2}(X_j - iY_j), \qquad a_j^\dagger = Q_j^\dagger Z_{<j},$$
$$Q_j = |0\rangle_j \langle 1|_j = \frac{1}{2}(X_j + iY_j), \qquad a_j = Q_j Z_{<j},$$

where G_j denotes the gate G applied to the j^{th} qubit and $G_{<j}$ denotes a sequence of G gates applied to *each* qubit of index less than j. The operators $Q_j^\dagger$ and Q_j perform the qubit-toggling[3], and the gates $Z_{<j}$ introduce the correct number of negative signs, according to how many qubits are toggled on in positions below j. In particular, parity counting is an intrinsic part of these operators, and so we are intrinsically interested in performing these parity counts efficiently.[4]

The encoding scheme above recounted above is not very access-efficient: the creation and annihilation operators both access $O(n)$ qubits in the average and worst cases, and more accesses means more stress on a quantum computer's stability and reliability. Bravyi and Kitaev observed that our earlier Fenwick tree encoding scheme translates directly into this situation, immediately reducing the access complexity to $O(\log n)$. For a prettier encoding, we switch to the *Majorana basis* of unitary operators:

$$c_j = a_j + a_j^\dagger, \qquad d_j = i(a_j^\dagger - a_j).$$

(In particular, this removes the obnoxious linear combination in the definitions of Q_j and $Q_j^\dagger$.) These two operators then translate into qubits by a Fenwick encoding as follows:

- The operator d_j is encoded by applying a Y–gate to the j^{th} qubit, an X–gate to each parent of the j^{th} qubit in the Fenwick tree, and applying a Z–gate to each immediate child of each parent of the j^{th} qubit, provided that child is labeled less than j. Respectively, the Y–gate comes from our change of basis, the X–gates toggle the occupancy counts, and the Z–gates introduce the relevant negative signs.
- The operator c_j is encoded by applying an X–gate to the j^{th} qubit, applying an X–gate to each parent of the j^{th} qubit in the Fenwick tree, and applying a Z–gate to each immediate child of each parent of the j^{th} qubit *including immediate children of j itself*, provided that child is labeled less than j. Their roles are identical to those described above.

Again, because of the observations about Fenwick tree width and depth, there are only $\approx 2\log_2 n$ gates involved in either operator.

Remark 3. This 1–dimensional case actually does have interesting applications: Seely, Richard, and Love used this encoding to give an efficient implementation of the second-order approximation to the Hamiltonian governing electron excitation in a(n ionized) hydrogen atom.

Remark 4. This also has the scent of something common in mathematics: the Koszul sign rule in homological algebra appears when applying an operator to different tensor factors in a large tensor product. The prototypical example is the Leibniz law on a derivation:

$$d(a_1 \otimes \cdots \otimes a_k) = \sum_{j=1}^{k} (-1)^{j-1} a_1 \otimes \cdots \otimes d(a_j) \otimes \cdots \otimes a_k.$$

In this sense, linear fermionic models are some kind of $\mathbb{Z}/2$–graded-commutative extension of qubits subject to the Koszul sign rule.

3. Qubit encoding of geometrically localized fermionic interaction

Based on the previous discussion, and especially the last remark, you might (rightly) expect that general fermionic simulation is computationally costly. In fact, motivated by the same sign-rule remark, you might also notice that you can re-encode classical qubit behavior in terms of linear fermionic sites by dedicating pairs of sites to represent any given qubit: by expanding each $|0\rangle$ and $|1\rangle$ into $|00\rangle$ and $|11\rangle$, one avoids collecting any alternating signs. In

[3]Note that these operators are *not* unitary.

[4]Exactly what region is tracked isn't important if we're just trying to enforce these relations. You can totally count parity upward if you want, which I hear is popular among quantum chemists.

fact, this indicates that linear fermionic models can serve as *constant time* simulators of qubits, so that they are, in a precise sense, more computationally expressive objects.

Troubled by this, Bravyi and Kitaev also noticed that most physically occuring fermionic systems have rather extreme locality properties: their Hamiltonians are governed by terms encoding adjacent site-to-site hopping. Accordingly, if we were going to try simulating physical systems of fermions, we need only provide a computational model that deals with these kinds of adjacent annihilation and creation oeprators, and furthermore only those operators that occur in *pairs*.[5] Working this out in the naive encoding above is encouraging: all but a few of the Z–gates cancel! Their prototypical system consists of a configuration of fermions occupying vertices on a(n undirected) graph, with the possibility to transition from one vertex to another along a single edge connecting them. The operators governing such a system are generated by the following two collections:

$$B_k = 1 - 2a_k^\dagger a_k, \qquad\qquad A_{(j,k)} = -i(a_j + a_j^\dagger)(a_k + a_k^\dagger),$$

indexed on vertices and edges respectively.[6] By placing a qubit on each edge in the graph, these operators admit the following encoding:

$$B_k = \prod_{\text{edges }(j,k)} Z_{(j,k)}, \qquad\qquad A_{(j,k)} = \varepsilon_{(j,k)} X_{(j,k)} \prod_{(l,k)<_k(j,k)} Z_{(l,k)} \prod_{(j,i)<_j(j,k)} Z_{(j,i)},$$

where $\varepsilon_{(j,k)} \in \{\pm 1\}$ is *any* choice of orientation of the edges of the graph satisfying $\varepsilon_{(j,k)} = -\varepsilon_{(k,j)}$ and $<$ is *any* choice of ordering on the edges on each vertex j. Additionally, a configuration is considered to be physical when it is *loop-stabilized* in the following sense: for any closed path $j_1 \cdots j_p$ we define a stabilizing operator

$$C_J = i^p A_{(j_1,j_2)} \cdots A_{(j_p,j_1)},$$

and a physical state is a simultaneous eigenvector for all the C_J operators.

Example 5. A prototypical example making use of this set-up is the Hubbard model: a family of spin-up and spin-down particles propagate through a graph according to the Hamiltonian

$$H = H_\uparrow + H_\downarrow + U \sum_{j\in V} (a_{j\uparrow}^\dagger a_{j\uparrow})(a_{j\downarrow}^\dagger a_{j\downarrow}),$$

consisting of two decoupled spin-up and spin-down propagation terms and a density-density interaction term, each specified by

$$H_\uparrow = -t \sum_{(j,k)} (a_{j\uparrow}^\dagger a_{k\uparrow} + a_{k\uparrow}^\dagger a_{j\uparrow}) + \varepsilon \sum_j a_{j\uparrow}^\dagger a_{j\uparrow}, \qquad H_\downarrow = -t \sum_{(j,k)} (a_{j\downarrow}^\dagger a_{k\downarrow} + a_{k\downarrow}^\dagger a_{j\downarrow}) + \varepsilon \sum_j a_{j\downarrow}^\dagger a_{j\downarrow},$$

where t, U, and ε are all physical parameters of the system governing ease of motion, typically occuring theoretically as path integrals.

Example 6. A specialization of this example was worked out by Halíček, Troyer, and Whitfield in the case of an evenly subdivided rectangle. This amounts to picking signs and working through the appearance of the X– and Z–gates in the encoding above. In particular, they observe that either one of the spin-up or spin-down partial Hamiltonians involves terms with only 7–gated qubit operators occuring, and the density-density interaction term involves terms with 8–gated qubit operators. It is not an accident that these values are *constant*: any upper bound on the incidence degree of a graph translates directly into an upper bound on the gate degree of the families of operators A and B.

Remark 7. For computational purposes, it is also useful to introduce a correctional *penalty term* that pushes the system back into the eigenspace of physical configurations. This is accomplished by selecting some large value $\Delta \gg t, \varepsilon, U$ and adding $-\Delta \sum_{\text{loops } J} C_J$ to the Hamiltonian above.

[5]Accordingly, what follows is *neither* a special case *nor* an extension of the discussion above. In some sense, this second topic exists to spite the first.

[6]This is an analogue of the Majorana basis mentioned earlier.

4. Extensions

Neither the Bravyi–Kitaev paper nor the Halíček–Troyer–Whitfield paper deal with more than adjacent site-to-site hopping terms. However, the major application of the earlier discussion of fermionic simulation, appearing in Seely–Richard–Love, involved hopping terms across greater differences, and it made serious use of the Fenwick segmentation scheme to gain control of access complexity. One potential avenue for exploration is a mixing of these two schemes: using Fenwick-like segmentation to gain some further operator locality for Hubbard-like Hamiltonians with higher-order terms.[7]

The transcoding of occupancy state into graph-edge state is not discussed in either of the two sources above. This is probably not an efficient operation, but it is *necessary* for this encoding scheme to be useful. One hint from Bravyi–Kitaev in this direction is that the fermionic ground state corresponds to the unique qubit state $|\xi\rangle$ satisfying the equations

$$B_k|\xi\rangle = |\xi\rangle.$$

I wrote a little pyQuil document that encodes a bunch of the basic operations in this set-up into Quil circuits. I do not understand how to encode the Hamiltonians, which is crucial for demonstrating the simulation capabilities of these schemes, essentially because I don't understand how to sum non-unitary operations in a reasonable (i.e., local) way. I imagine this is just ignorance, and that someone with more experience with quantum computing or with pyQuil would be able to implement this quickly.

The "superfast" in superfast geometrically-localized fermionic simulation refers to the constant gate size, so that the access complexity of applying a Hamiltonian amounts to $O(\ell \cdot w)$. It is conceivable that this can be optimized slightly further, within the same Landau class, by again aiming to minimize duplicated parity update. A more fine-grained analysis of this kind has not been treated.

[7]There are sub-questions here, like: what are the important properties of covering regimes in order to govern occupancy counting efficiently? Our preference for initial segments on a line was completely arbitrary, and it is much less convincing that a similar stab in the dark will work on even, say a rectangular 2–dimensional grid. Things will almost certainly become more complicated—in an interesting way—if the graph has an interesting topology. A 2–torus is probably a good example.