## D.4 Search problems

This lecture is based primarily on IQC.

### D.4.a Overview

¶1. Many problems can be formulated as search problems over a solution space $\mathcal{S}$. That is, find the $x \in \mathcal{S}$ such that some predicate $P(x)$ is true.

¶2. For example, hard problems such as the Hamiltonian paths problem and Boolean satisfiability can be formulated this way.

¶3. **Unstructured search problem:** a problem that makes no assumptions about the *structure* of the search space, or for which there is no known way to make use of it.
Also called a *needle in a haystack problem*.

¶4. That is, information about a particular value $P(x_0)$ does not give us usable information about another value $P(x_1)$.

¶5. **Structured search problem:** a problem in which the structure of the solution space can be used to guide the search.
Example: searching an alphabetized array.

¶6. **Cost:** In general, unstructured search takes $\mathcal{O}(M)$ evaluations, where $M = |\mathcal{S}|$ is the size of the solution space (which is often exponential in the size of the problem).
On the average it will be $M/2$ (think of searching an unordered array).

¶7. **Grover's algorithm:** We will see that Grover's algorithm can do unstructured search on a quantum computer with bounded probability in $\mathcal{O}(\sqrt{M})$ time, that is, quadratic speedup. This is provably more efficient than any algorithm on a classical computer.

¶8. **Optimal:** This is good (but not great). Unfortunately, it has been proved that Grover's algorithm is optimal for unstructured search.

¶9. Therefore, to do better requires exploiting the structure of the solution space. *Quantum computers do not exempt us from understanding the problems we are trying to solve!*

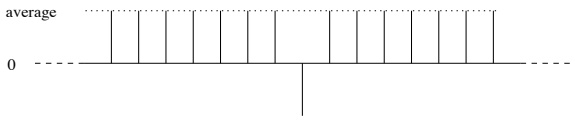


Figure III.28: Depiction of the result of phase rotation (changing the sign) of solutions in Grover's algorithm. [source: IQC]

---

¶10. Shor's algorithm is an excellent example of exploiting the structure of a problem domain.

¶11. Later we will take a look at *heuristic quantum search algorithms* that do make use of problem structure.

### D.4.b Grover

¶1. Pick $n$ such that $2^n \geq M$.
Let $N = 2^n$
and let $\mathbf{N} = \mathbf{2}^n = \{0, 1, \ldots, N-1\}$, the set of $n$-bit strings.

¶2. Suppose we have a quantum gate array $U_P$ that computes the predicate:

$$U_P|x, y\rangle = |x, y \oplus P(x)\rangle.$$

¶3. **Application:** As usual, we begin by applying the function to a superposition of all possible inputs $|\psi_0\rangle$:

$$U_P|\psi_0\rangle|0\rangle = U_P\left[\frac{1}{\sqrt{N}}\sum_{x\in\mathbf{N}}|x, 0\rangle\right] = \frac{1}{\sqrt{N}}\sum_{x\in\mathbf{N}}|x, P(x)\rangle.$$

¶4. Notice that the components we want, $|x, 1\rangle$, and the components we don't want, $|x, 0\rangle$, all have the same amplitude, $\frac{1}{\sqrt{N}}$.
So if we measure the state, the chances of getting a hit are very small, $\mathcal{O}(2^{-n})$.

¶5. The trick, therefore, is to amplify the components that we want at the expense of the ones we don't want; this is what Grover's algorithm accomplishes.

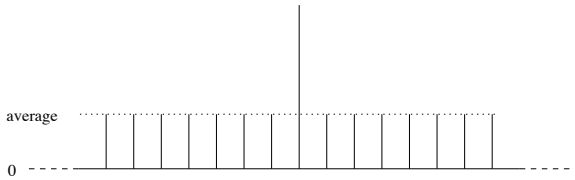


Figure III.29: Depiction of result of inversion about the mean in Grover's algorithm. [source: IQC]

---

¶6. To do this, first we change the sign of every solution (a phase rotation of $\pi$).
That is, if the state is $\sum_j a_j|x_j, P(x_j)\rangle$, then we want to change $a_j$ to $-a_j$ whenever $P(x_j) = 1$.
See Fig. III.28.
I'll get to the technique in a moment.

¶7. Next, we invert all the components around their mean amplitude (which is a little less than the amplitudes of the non-solutions).
The result is shown in Fig. III.29.
This amplifies the solutions.

¶8. **Iteration:** This *Grover iteration* (the sign change and inversion about the mean) is repeated $\frac{\pi\sqrt{N}}{4}$ times.
Thus the algorithm is $\mathcal{O}(\sqrt{N})$.

¶9. **Measurement:** Measurement yields an $x_0$ for which $P(x_0) = 1$ with high probability.

¶10. **Probability:** Specifically, if there is exactly one solution $x_0 \in \mathcal{S}$, then $\frac{\pi\sqrt{N}}{8}$ iterations will yield it with probability 1/2.
With $\frac{\pi\sqrt{N}}{4}$ iterations, the probability of failure drops to $1/N = 2^{-n}$.

¶11. Unlike with most classical algorithms, additional iterations will give a *worse* result.
This is because Grover iterations are unitary rotations, and so excessive rotations can rotate past the solution.

¶12. Therefore it is critical to know when to stop.

Fortunately there is a quantum technique (Brassard et al. 1998) for determining the optimal stopping point.

¶13. Grover's iteration can be used for a wide variety of problems as a part of other quantum algorithms.

¶14. **Changing the sign:** Now for the techniques for changing the sign and inversion about the mean. To change the sign, simply apply $U_P$ to $|\psi_k\rangle|-\rangle$.
To see the result, let $X_0 = \{x \mid P(x) = 0\}$ and $X_1 = \{x \mid P(x) = 1\}$, the solution set. Then:

$$
\begin{aligned}
& U_P|\psi_k\rangle|-\rangle \\
&= U_P\left[\sum_{x\in\mathbf{N}} a_x|x,-\rangle\right] \\
&= U_P\left[\frac{1}{\sqrt{2}}\sum_{x\in\mathbf{N}} a_x|x,0\rangle - a_x|x,1\rangle\right] \\
&= \frac{1}{\sqrt{2}}\,U_P\left[\sum_{x\in X_0} a_x|x,0\rangle + \sum_{x\in X_1} a_x|x,0\rangle - \sum_{x\in X_0} a_x|x,1\rangle - \sum_{x\in X_1} a_x|x,1\rangle\right] \\
&= \frac{1}{\sqrt{2}}\left[\sum_{x\in X_0} a_x U_P|x,0\rangle + \sum_{x\in X_1} a_x U_P|x,0\rangle\right. \\
&\qquad \left. - \sum_{x\in X_0} a_x U_P|x,1\rangle - \sum_{x\in X_1} a_x U_P|x,1\rangle\right] \\
&= \frac{1}{\sqrt{2}}\left[\sum_{x\in X_0} a_x|x,0\rangle + \sum_{x\in X_1} a_x|x,1\rangle\right. \\
&\qquad \left. - \sum_{x\in X_0} a_x|x,1\oplus 0\rangle - \sum_{x\in X_1} a_x|x,1\oplus 1\rangle\right] \\
&= \frac{1}{\sqrt{2}}\left[\sum_{x\in X_0} a_x|x\rangle|0\rangle + \sum_{x\in X_1} a_x|x\rangle|1\rangle - \sum_{x\in X_0} a_x|x\rangle|1\rangle - \sum_{x\in X_1} a_x|x\rangle|0\rangle\right] \\
&= \frac{1}{\sqrt{2}}\left(\sum_{x\in X_0} a_x|x\rangle - \sum_{x\in X_1} a_x|x\rangle\right)(|0\rangle - |1\rangle)
\end{aligned}
$$

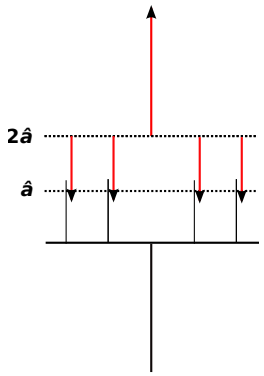


Figure III.30: Process of inversion about the mean in Grover's algorithm. The black lines represent the original amplitudes $a_j$. The red lines represent $2\bar{a} - a_j$, with the arrow heads indicating the new amplitudes $a'_j$.

---

$$= \left( \sum_{x \in X_0} a_x |x\rangle - \sum_{x \in X_1} a_x |x\rangle \right) |-\rangle.$$

Therefore the signs of the solutions have been reversed (they have been rotated by $\pi$).

¶15. Notice how $|-\rangle$ in the target register can be used to separate the 0 and 1 results by rotation. This is a useful idea!

¶16. **Inversion about the mean:** To perform inversion about the mean, let $\bar{a}$ be the average of the $a_j$.
See Fig. III.30.

¶17. Inversion about the mean is accomplished by the transformation

$$\sum_{j \in \mathbf{N}} a_j |x_j\rangle \mapsto \sum_{j \in \mathbf{N}} (2\bar{a} - a_j) |x_j\rangle.$$

To see this, suppose $a_j = \bar{a} \pm \delta$.
Then $2\bar{a} - a_j = 2\bar{a} - (\bar{a} \pm \delta) = \bar{a} \mp \delta$.
Therefore an amplitude $\delta$ below the mean will be transformed to $\delta$ above, and vice verse.
But an amplitude that is negative, and thus very far below the mean, will be transformed to an amplitude much above the mean.

¶18. **Diffusion transformation:** Inversion is accomplished by a "diffusion transformation" $D$.
To derive the matrix $D$, consider the new amplitude $a'_j$ as a function of all the others:

$$a'_j = 2\bar{a} - a_j = 2\left(\frac{1}{N}\sum_{k=0}^{N-1} a_k\right) - a_j = \sum_{k\neq j}\frac{2}{N}a_k + \left(\frac{2}{N} - 1\right)a_j.$$

¶19. This matrix has $\frac{2}{N} - 1$ on the main diagonal and $\frac{2}{N}$ in the off-diagonal elements:

$$D = \begin{pmatrix} \frac{2}{N}-1 & \frac{2}{N} & \cdots & \frac{2}{N} \\ \frac{2}{N} & \frac{2}{N}-1 & \cdots & \frac{2}{N} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{2}{N} & \frac{2}{N} & \cdots & \frac{2}{N}-1 \end{pmatrix}.$$

¶20. It is easy to confirm $DD^\dagger = I$ (exercise), so the matrix is unitary and therefore a possible quantum operation, but it remains to be seen if it can be implemented efficiently.

¶21. **Claim:** $D = W_n R W_n$, where $W_n = H^{\otimes n}$ is the $n$-qubit Walsh-Hadamard transform and

$$R \stackrel{\text{def}}{=} \begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & -1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & -1 \end{pmatrix}.$$

¶22. To see this, let

$$R' \stackrel{\text{def}}{=} R + I = \begin{pmatrix} 2 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{pmatrix}.$$

Then $WRW = W(R' - I)W = WR'W - WW = WR'W - I$.

¶23. It is easy to show (exercise) that:

$$WR'W = \begin{pmatrix} \frac{2}{N} & \frac{2}{N} & \cdots & \frac{2}{N} \\ \frac{2}{N} & \frac{2}{N} & \cdots & \frac{2}{N} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{2}{N} & \frac{2}{N} & \cdots & \frac{2}{N} \end{pmatrix}.$$

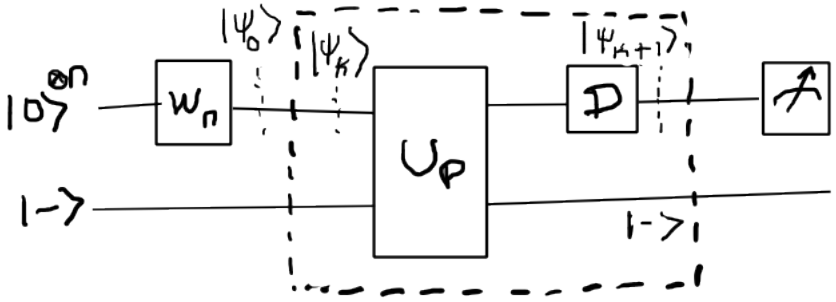


Figure III.31: Circuit for Grover's algorithm. The Grover iteration in the dashed box is repeated $\frac{\pi\sqrt{N}}{8}$ times.

---

It therefore follows that $D = WR'W - I = WRW$.

¶24. See Fig. III.31 for a diagram of Grover's algorithm.

### D.4.c Hogg's heuristic search algorithms

¶1. **Constraint satisfaction problems:** Many important problems can be formulated as *constraint satisfaction problems*, in which we try to find a set of assignments to variables that satisfy specified *constraints*.

¶2. More specifically let $V = \{v_1, \ldots, v_n\}$ is a set of variables,
and let $X = \{x_1, \ldots, x_n\}$ is a set of values that can be assigned to the variables,
and let $C_1, \ldots, C_l$ be the constraints.

¶3. The set of all possible assignments of values to variables is $V \times X$.
Subsets of this set correspond to full or partial assignments, including inconsistent assignments.
The set of all such assignments is $\mathcal{P}(V \times X)$.

¶4. **Lattice:** The sets of assignments form a *lattice* under the $\subseteq$ partial order (Fig. III.32).

¶5. **Binary encoding:** By assigning bits to the elements of $V \times X$, elements of $\mathcal{P}(V \times X)$ can be represented by $mn$-element bit strings (i.e.,

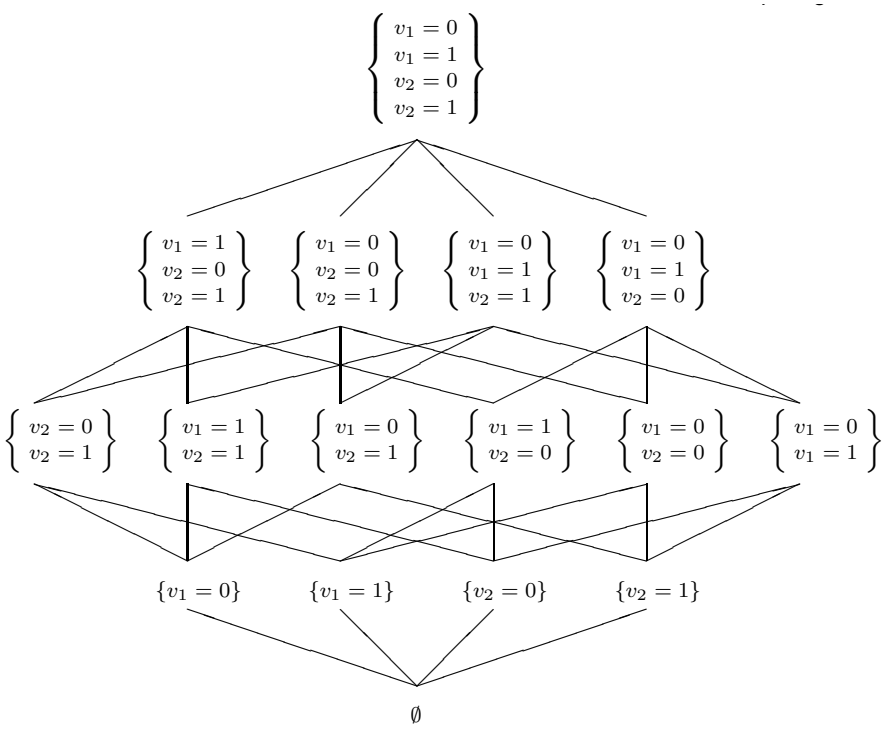


Figure III.32: Lattice of variable assignments. [source: IQC]

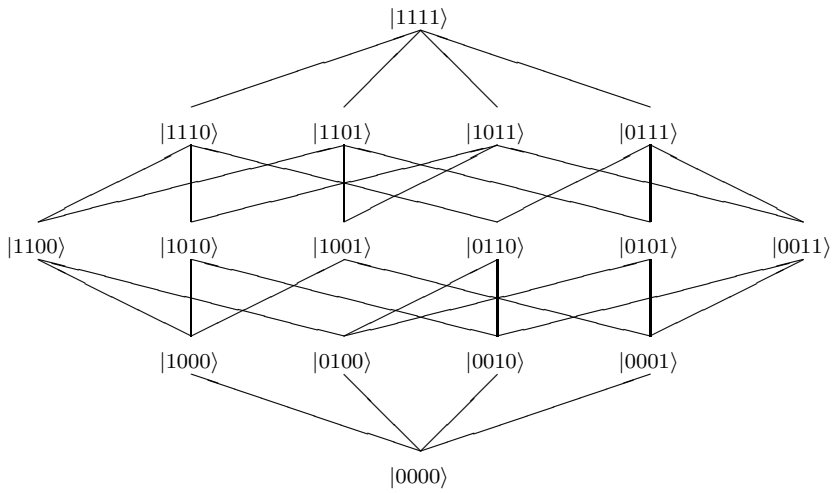


Figure III.33: Lattice of binary strings corresponding to all subsets of a 4-element set. [source: IQC]

---

integers in the set $\mathbf{MN} = \{0, \ldots, 2^{mn} - 1\}$).
See Fig. III.33.

¶6. **Idea:** Hogg's algorithms are based on the observation that if an assignment violates the constraints, then so do all those above it.

¶7. **Initialization:** The algorithm begins with all the amplitude concentrated in the bottom of the lattice, $|0 \cdots 0\rangle$ (i.e., the empty set of assignments).

¶8. **Movement:** The algorithm proceeds by moving amplitude up the lattice, while avoiding assignments that violate the constraints.
That is, we want to move amplitude from a set to its supersets.
For example, we want to redistribute the amplitude from $|1010\rangle$ to $|1110\rangle$ and $|1011\rangle$.
Hogg has developed several methods.

¶9. One method is based on the assumption that the transformation has the form $WDW$, where $W = H^{\otimes mn}$, the $mn$-dimensional Walsh-Hadamard transformation, and $D$ is diagonal.
The elements of $D$ depend on the size of the sets.

¶10. Recall (¶13, p. 112) that

$$W|x\rangle = \frac{1}{\sqrt{2^{mn}}} \sum_{z \in \mathbf{MN}} (-)^{x \cdot z} |z\rangle.$$

¶11. As shown in Sec. A.2.h (p. 53), we can derive a matrix representation for $W$:

$$\begin{aligned} W_{jk} &= \langle j \mid W \mid k \rangle \\ &= \langle j | \frac{1}{\sqrt{2^{mn}}} \sum_{z \in \mathbf{MN}} (-)^{k \cdot z} |z\rangle \\ &= \frac{1}{\sqrt{2^{mn}}} \sum_{z \in \mathbf{MN}} (-)^{k \cdot z} \langle j \mid z \rangle \\ &= \frac{1}{\sqrt{2^{mn}}} (-1)^{k \cdot j}. \end{aligned}$$

¶12. Note that $k \cdot j = |k \cap j|$, where on the right-hand side we interpret the bit strings as sets.

¶13. **Constraints:** The general approach is to try to stear amplitude away from sets that violate the constraints, but the best technique depends on the particular problem.

¶14. One technique is to invert the phase on bad subsets so that they tend to cancel the contribution of good subsets to supersets.
This could be done by a process like Grover's algorithm using a predicate that tests for violation of constraints.

¶15. Another approach is to assign random phases to bad sets.

¶16. **Efficiency:** It is difficult to analyze the probability that an iteration of a heuristic algorithm will produce a solution.
Therefore their efficiency is usually evaluated empirically.
This technique will be difficult to apply to quantum heuristic search until larger quantum computers are available.
Recall that classical computers require exponential time to simulate quantum systems.

¶17. **Hogg's algorithms:** Small simulations indicate that Hogg's algorithms may provide polynomial speedup over Grover's algorithm.