Quantum Computation **(CMU 18-859BB, Fall 2015)**

Lecture 10: Hidden Subgroup Problem

10/12/15

Lecturer: John Wright *Scribe: Lachlan Lancaster*

1 Review and Setting the Stage

We start by reviewing the problems considered in the last lecture, their similarities, and how they can be generalized to a more general problem (The Hidden Subgroup Problem) that is the topic of today's lecture. We then review some ideas of Group Theory before diving in.

1.1 A Single Problem, Leading a Double Life

We recall from last time the Period Finding Problem, which is critical in Shor's Algorithm, the problem can be given as:

Problem:
Given: $f : \mathbb{Z}_N \to$ "`colors`" with the promise that $\exists s \in \mathbb{Z}_N . s \neq 0$ such that $\forall x \in \mathbb{Z}_N$, $f(x) = f(x+s)$ OR all f-vaues are distinct.
Goal: Find s

We will refer to the above statement of the problem as the (A) statement of the problem. We then also have Simon's Problem, which can be stated very similarly to the above and we have also encountered before in the course. This problem can be stated in the following way:

Problem:
Given: $f : \mathbb{Z}_2^n \to$ "`colors`" with the promise that $\exists s \in \mathbb{Z}_2^n . s \neq 0$ such that $\forall x \in \mathbb{Z}_2^n$, $f(x) = f(x+s)$ OR all f-vaues are distinct.
Goal: Find s

We will refer to this definition of Simon's Problem as (1). Obviously the two above problem's look very similar, it's almost as though the scribe just copied the two definitions and simply changed a few details... The answer to this unification is the topic of today's lecture, but first we will go over some simple reminder topics from Group Theory.

1.2 Galois's Soliloquy

We want to restate the statements (A) and (1) of the problems at hand in terms of Group Theory. To do this let's first restrict to the case of the Period-Finding problem and enumerate what could be considered a 'solution' to the problem

$$H := \{0, s, 2s, 3s, \dots\}$$

We see that, according to statement (A), the function f takes the same value on every member of the set H, so in a sense if we know this set, then we have our solution, we can easily find s. We also notice that H is a *subgroup* of the larger group $G_p := (\mathbb{Z}_N, +)$ of the integers modulo N with addition (modulo N) as the group operation. What is we add one to everything in the above set? By the problem statement of the period finding problem, we should have another distinct set of things which are all assigned a different color, and the same thing would happen if we added two to everything in H. So we have:

$$\begin{aligned} 1+H &:= \{1+h \mid h \in H\} \\ 2+H &:= \{2+h \mid h \in H\} \\ &\vdots \\ x+H &:= \{x+h \mid h \in H\} \end{aligned}$$

We see easily by construction that each of these sets has a the same size and is assigned a different f-value. It is also quite easy to see that, as long as we don't repeat this process to many times and come back to H itself, each of these sets is going to be disjoint. This intuitively makes sense in when we note they have different f-values. Well, in the language of the Group Theory, the sets we generated above are called *cosets* of the subgroup H:

Definition 1.1. A **coset** of a subgroup H of a group G is the set $\{xh \mid h \in H\}$ for some $x \in G$. This set is usually denoted xH.

Definition 1.2. A group G is said to be **Abelian** or **commutative** if $\forall x, y \in G$ we have that $xy = yx$. That is to say that the group operation is commutative.

Specifically we should note that the above examples are the *left cosets* of H, the *right cosets* being the sets $H + x$, but since the present group G_p is Abelian this doesn't make a difference. We can take for example the case that $N = 2^n$ for some $n \in \mathbb{N}$ (which is usually convneient) and consider the following subgroup and its cosets:

$$\begin{aligned} H &:= \{0,4,8,\dots\} \to \text{Green} \\ 1+H &:= \{1,5,9,\dots\} \to \text{Red} \\ 2+H &:= \{2,6,10,\dots\} \to \text{Blue} \\ 3+H &:= \{3,7,11,\dots\} \to \text{Yellow} \end{aligned}$$

We see in the above example that the above exactly have the properties we specified for the above example with $s = 4$, we also note that only *one* of the above sets is a subgroup, obviously it is H.

We now have the machinery we wanted to restate the problems as we stated them above.

Problem:
Given: $\exists H$ which is a subgroup of $\mathbb{Z}_N$ with addition modulo N such that f assigns the same value to every member of H
Goal: Find H

We call the above set theoretic statement of the problem (B). We origianlly considered this to be a *slightly* more general problem because it allows you any subgroup H, but every subroup of $\mathbb{Z}_N$ is cyclic and therefore can be generated in teh sense of statement (A). We then have finally have the following statement of the Simon's Problem, which we will refer to as (2):

Problem:
Given: Let $H = \{0, s\} \subseteq \mathbb{Z}_2^n$ which is a subgroup of $\mathbb{Z}_2^n$ with vector addition modulo 2 (as $s + s = s \oplus s = 0$). There exists an $s \in \mathbb{Z}_2^n$ and equivalent unique subgroup H as definied above such that f assigns a unique color to each coset of H
Goal: Find H

We *do* in this case have that this statement is more general than the statement (1) due to the less trvial structure of the this group. In any case, we can clearly see the similarities between these problems, (B) and (2) look even more similear than (A) and (1)! We're on the edge of a big discovery...

2 Protagonist Revealed: The HSP and Applications

We now introduse the general **Hidden Subgroup Problem** of which both of the above cases can be reduced to.

2.1 Problem Statement

Problem:
Given: Let G be a group and f be a given function such that $f : G \rightarrow$``colors". We are promised that $\exists H \subseteq G$ which is a subroup of G such that f assigns a uniqu color to each coset of H.
Goal: Find H

We note that in some cases the subgroup H may be exponentially large, so the algorithm itself will output a generating set for the subroup H. We are guaranteed that there will always be a generating set for this subgroup of polynomial size as stated (without proof) in lecture. Further similar discussions may be found in [DW71, HHR07].

2.2 Reductions

We care about this because there are a *lot* of problems that can be reduced to this more general case. Here are some examples:

Problem	Group	
Simon's	$\mathbb{Z}_2^n$	Abelian Groups
Factoring/Period Finding	$\mathbb{Z}_N$	Abelian Groups
Discrete Log	$\mathbb{Z}_N \times \mathbb{Z}_N$	Abelian Groups
Pell's Equation/Princp. Ideal Problem	$\mathbb{R}$	Abelian Groups
Graph Isomorphism	S_n (Symmetric Group)	Non-Abelian Groups
Shortest Vector Problem	D_N (Dihedral Group)	Non-Abelian Groups

To make things clear we'll give a shor definition of the non-typical groups mentioned above:

Definition 2.1. The **Symmetric group** on N elements, or S_N is the group who's underlying set is the set of all bijections from $[N] = 1, 2, \dots N-1, N$ to itself. The group operation on this set is the composition of two of these functions, as these functions are bijections, a composition of two of them is clearly a bijection, each one has an inverse, and the identity transformation belongs to this set.

Definition 2.2. The **Dihedral group** D_N is the group who's underlying set ig the set of symmetries of an N-sided regular polygon. Again here the group operation is the composition of two symmetries. Clearly the identitiy transformation is a symmetry, the composition of two symmetries is also a symmetry, and every symmetry has an inverse symmetry.

In returning to trying to solve the problems above we fiind our selves in one of two general cases most of the time for algorithm's to solve these problems on a Quantum Computer:

Case 1: The underlying group G associated with the problem is Abelian, in which case we can solve the problem with polylog$|G|$ number of gates and calls to the provided oracle O_f. This is the case with the first four of the problems listed in the table above. Generally, these cases are not all that interesting.

Case 2: The underlying group G associated with the problem is non-Abelian, in which case we can have an algorithm that has polylog$|G|$ calls to the oracle function O_f, but it may need and exponenital number of gates to run succesfully.

Unfortunately, it seems as though we have generally disjoint sets of interesting problems on one side and tractably solvable problems on the other, but not generally. Additionally (cause we know you're all thinking it) many people believe that quantum computers cannot have algorithms that solve NP-complete problems in polynomial time.

2.3 How to Solve These

So how wuld we go about implenting and algorithm to solve this problem generally? We have the following set-up. Well, we know we have the function of $f : G \rightarrow$"`colors`" (or equivalently $\{0,1\}^n$). It is then convenient to basically "round up" so that the numbers that we are dealing with are nice powers of two, we do this by setting $n = \lceil \log_2 |G| \rceil$ number of bits. This is annoying, but neccessary.

We then have the, supposedly given, oracle function:

$$O_f : |g\rangle |x\rangle \to |g\rangle \otimes |x \oplus f(g)\rangle \tag{1}$$

Which is defined if $g \in G$ and undefined otherwise. Note we *will* hit the case where it is undefined because of the way we have expanded the number of bits for convenience. We then use this set-up in the next section.

3 The Standard Method

We present here the way in which "basically everyone" solves the Hidden Subgroup Problem [Lom04].

3.1 Presentation

Here's the algorithm:

Step 1: We begin by preparing a uniform superposition of $g \in G$ call it $|\psi\rangle$ (or more literally a linear superposition of $x \in 2^n$ where $n = \lceil \log_2 |G| \rceil$). We do this in the usual way by applying the Hadamard gate to each of n input qubits:

$$|\psi\rangle = \frac{1}{\sqrt{2^n}} \sum_{|x\rangle \in [N]} |x\rangle \approx \frac{1}{\sqrt{|G|}} \sum_{g \in G} |g\rangle$$

Step 2: We attach the necessary m ancillary bits $|0^m\rangle$

Step 3: We apply the Oracle O_f to get:

$$O_f\left(|\psi\rangle \otimes |0^m\rangle\right) = \frac{1}{\sqrt{|G|}} \sum_{g \in G} |g\rangle \otimes |f(g)\rangle$$

Step 4: We measure the second register, that is the bits who represent the ket $|f(g)\rangle$ in the above equation. We get Pr[observe color '$\texttt{c}$'] = fraction of f-values equal to $\texttt{c}$. Therefore we see a uniformly random color (as each coset is of the same size). This then causes the state of the system to collapse to collapse to the state that is consistent with seeing that color. We see we go from the generally stated version of the state on the right, to the state on the left after the measurement:

$$\frac{1}{\sqrt{|G|}} \sum_{\texttt{c} \in \texttt{colors}} \sum_{g \ \text{if} f(g) = \texttt{c}} |g\rangle \otimes |\texttt{c}\rangle \to \left[\frac{1}{\sqrt{|H|}} \sum_{g \ \text{if} f(g) = \texttt{c}} |g\rangle \right] \otimes |\texttt{c}\rangle$$

We then note that $f^{-1}(\mathtt{c}) = gH$ for some $g \in G$ and some subgroup H of G. We can then rewrite the above (dropping the tensor with the redundant $|c\rangle$) as:

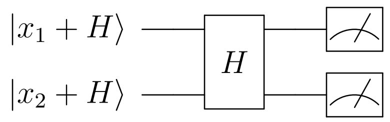
$$\frac{1}{\sqrt{|H|}} \sum_{h \in H} \tag{2}$$

The above equation (2) is by convention called the **coset state** and is usually represented by $|gH\rangle$. So, to summarize, what have we done? We have created an algorithm which calls the oracle gate once and outputs a *uniformly random* coset state $|gH\rangle$. We can see that this state is uniformly random as the $\mathtt{c}$ is uniformly random and we have a bijection between these properties.

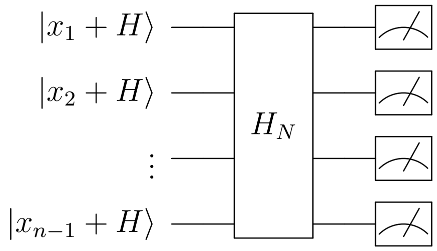
This is the exact smae distribution of picking a uniformly random $g \in G$ and outputting the coset state $|gH\rangle$.

3.2 We've Been Here Before...

We now note that if we look back carefully at our algorithms from the beginning of the lecture, that we've seen this before! In the Period Finding Problem we run the above algorithm twice by gernerating tow states, Hadamard transforming them and the performing some other classical operations. The intermediate quantum steps are shown below:



We saw a similar case for Simon's Problem, this time generating $n-1$ such states

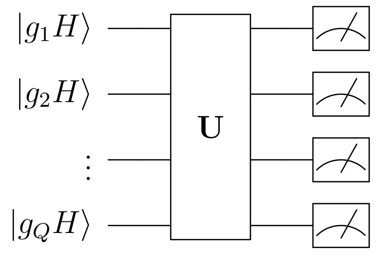


This process is the followed up by more classical computation.

3.3 The EHK Algorithm

We now review an algorithm given by Ettinger, Hoyer, and Knill [EHK04]. They showed by demonstarting and algorithm which did so, that the quantum query complexity of the Hidden Subgroup Problem is $Q = \text{poly}(n)$ where $n = \lceil \log_2 |G| \rceil$ for the associated group G.

This happens by first generateing Q cosets states using the general method outlined above. They then outline a very large and entangled unitary transformation U that must be applied to these states. This unitary transformation can be quite large and take $\mathcal{O}(\log |G|)$ number of gates to run for Abelian groups G. It then has the following picture:



The reader who is interested in the process of creating the (generally very complicated) transformation **U** is encourages to check the paper cited above. We note that the examples we gave above for the Period Finding and Simon's problem did not involve entangled operations, whereas the transformation **U** does.

4 The Final Act: Graph Isomorphism

We now present the algorithm for solving the Graph Isomorphism problem with the Hidden Subgroup Problem standard method. Of course, as noted above, the underlying group here is the Symmetric Group on n elements. We first state the problem for reference:

Problem:

Given: Two graphs $C_1 = (V_1, E_1)$ and $C_2 = (V_2, E_2)$ which are represented here as ordered pairs of a set of vertices V_i and a set of edges between these vertices (ordered pairs of vertices) E_i. We call the grpahs C_i instead of G_i in order to avoid confusion with the previous discussion using G to denote groups. We assume here that $V_1 = V_2 = \{1, 2, \dots, n\}$, that is that the set of vertices have the same size. We also assume for convenience that the sets are connected. We note that is this is not the case then we can separate these graphs into the cases of their disjoint components. We saw in homework 3 that we could decide this connectedness (returning a spanning tree) in $\mathcal{O}(\sqrt{n})$ time, so this is not a huge problem.

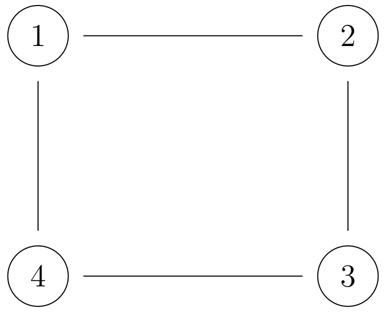
Goal: Output "Yes" if the Graphs C_1 & C_2 are isomorphic, output "No" otherwise.

Let's first give a formal definition of the isomorphism that we mean and the consider an example.

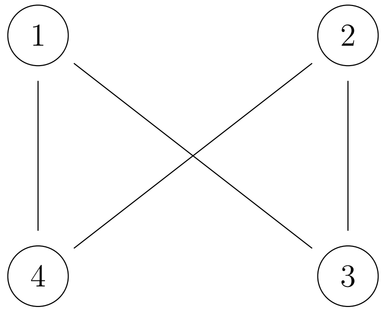
Definition 4.1. Two graphs C_1 and C_2 with sizes $|V_1| = |V_2| = n$ are said to be **isomorphic** if $\exists \pi : [n] \to [n]$ a bijection (permutation of $[n]$ such that π matches the edges of C_1 with the edges of C_2, or equivalently vice versa. That is to say that for $i, j \in V_1$ C_1 and c_2 are isomorphic if $\exists \pi$ such that $(i, j) \in E_1 \leftrightarrow (\pi(i), \pi(j)) \in E_2$.

We then consider the case of the rtwo following graphs who have vertx sets of size four. We will refer to the first graph as C_1 and the second graph as C_2

C_1



C_2



We then note that if we define the permutation π by $\pi : \{1, 2, 3, 4\} \to \{1, 3, 2, 4\}$ that we end up with the same edges between the nodes. So the above two graphs are isomorphic! This somehwat motivates a second definition of isopmorphism:

Definition 4.2. Given a graph $C = (V, E)$ and a permutation $\pi \in S_n$ where $|V| = n$ we may define the new graph $\pi(C)$ by $\pi(C) = (V, \pi(E))$ where $(i, j) \in E \leftrightarrow (\pi(i), \pi(j)) \in \pi(E)$.

We then can say that two graphs C_1 and C_2 are **isomorphic** is there exists a permutation π such that $\pi(C_1) = C_2$.

We will now give some short lemmas and definitions which will be esssential fo rthe main result.

Definition 4.3. The **automorphism group** of a graph C is the set of permutations of the graph nodes which leave it unchanged, denoted $\mathrm{Aut}(C)$. That is:

$$\mathrm{Aut}(C) = \{\pi \in S_n | \pi(C) = C\} \tag{3}$$

If we consider the two graphs above as one disconnected graph (as we will do below) we see that acting with π on C_2 and π^{-1} on C_1 gives us the same composite graph.

Lemma 4.4. *The Automorphism Group on a graph $C = (V, E)$ is a subgroup of the group S_n where $n = |V|$.*

Proof. We first note the it is clear the $\mathrm{Aut}(C) \subseteq S_n$, since all of it's elements are elements of S_n. We then need to show that it (i) contains the identity element, (ii) each of it's elements has an inverse in the set, and (iii) it is closed under the group operation.

(i) It is clear that the identity element is in $\mathrm{Aut}(C)$, as ths identity permutation I just gives back C so $I(C) = C$ trivially.

(ii) Suppose $\pi \in \mathrm{Aut}(C)$, then by definition $\pi(C) = C$. Applying π^{-1} to both sides of this equation we have $\pi^{-1}(\pi(C)) = \pi^{-1}(C)$ this leads simply by definition of the inverse to $C = \pi^{-1}(C)$, showing that $\pi^{-1} \in \mathrm{Aut}(C)$ by definition of the Automorphism group.

(iii) Suppose $\pi, \sigma \in \mathrm{Aut}(C)$, we want to show that their composition is also in this set. This is done easiuly as follows:

$$\begin{aligned} \pi(\sigma(C)) &= \pi(C) \texttt{ as } \sigma \in \texttt{ Aut}(C) \\ &= C \texttt{ as } \sigma \in \texttt{ Aut}(C) \end{aligned}$$

So that clearly $\pi\pi \in \mathrm{Aut}(C)$ and thus it is closed under the group operation.

We then have shown all the necessary properties to demonstrate that $\mathrm{Aut}(C)$ is a subgroup of S_n. □

Lemma 4.5. *Suppose that for a graph C we have a function $f : S_n \to \{\texttt{graphs on n vertices}\}$, defined such that $f(\pi) \to \pi(C)$ which we can treat like our "colors" in this problem. Then this function f 'hides' the Subgroup Aut(C) of C. That is that f assigns a unique value to each coset of* $\mathrm{Aut}(C)$.

Proof. We need to show that f assigns a unique value to each coset of the automorphism group.

Part 1: To do this we state a general coset of the automorphism group as $\sigma\mathrm{Aut}(C)$. We then suppose $\pi_1, \pi_2 \in \sigma\mathrm{Aut}(C)$. Then we may state $\pi_1 = \sigma\omega_1$ and $\pi_2 = \sigma\omega_2$ for $\omega_1, \omega_2 \in \mathrm{Aut}(C)$. We then show that $f(\pi_1) = f(\pi_2)$.

$$\begin{aligned} f(\pi_1) &= \pi_1(C) \texttt{ def of } f \\ &= \sigma\omega_1(C) \texttt{ def of } \pi_1 \\ &= \sigma(\omega_1(C)) \\ &= \sigma(C) \texttt{ def of } \mathrm{Aut}(C) \\ &= \sigma(\omega_2(C)) \texttt{ def of } \mathrm{Aut}(C) \\ &= \pi_2(C) \texttt{ def of } \pi_2 \\ &= f(\pi_2) \texttt{ def of } f \end{aligned}$$

Thus $f(\pi_1) = f(\pi_2)$

Part 2: We then want to show that $f(\pi_1) = f(\pi_2)$ for $\pi_1, \pi_2 \in S_n$ implies that $\pi_1 \in \pi_2\mathrm{Aut}(C)$ or equivalently $\pi_2 \in \pi_1\mathrm{Aut}(C)$, this comes easily from the following:

$$\begin{aligned}
f(\pi_1) = f(\pi_2) &\leftrightarrow \pi_1(C) = \pi_2(C) \texttt{ def of } f \\
&\leftrightarrow (\pi_2^{-1}\pi_1)(C) = C \texttt{ def of inverses} \\
&\leftrightarrow \pi_2^{-1}\pi_1 \in \mathrm{Aut}(C) \texttt{ def of } \mathrm{Aut}(C) \\
&\leftrightarrow \pi_1 = \pi_2\pi_2^{-1}\pi_1 \in \pi_2\mathrm{Aut}(C)
\end{aligned}$$

This is exactly what we desired to show, so that f does indeed hide the automorphism group of C. □

With these lemmas and definitions in hand we will see that the main result will be wuite easy.

Theorem 4.6. *Graph Isomorphism between two graphs $C_1 = (V_1, E_1)$ and $C_2 = (V_2, E_2)$ reduces to HSP on the group S_{2n} where $n = |V_1| = |V_2|$.*

Proof. Given these two graphs we form the composite graph $C_U := C_1 \cap C_2$ defined as the disjoint union of two separate graphs. This is equivalent to taking the two grpahs separately and then simply considering them as a single 'structure.'

We then define the funciton f mapping S_{2n} to the set of graphs on $2n$ vertices in a similar way to that given in Lemma 4.5 by saying $f(\pi) = \pi(C_U)$.

In this case, treating the problem as the HSP problem we see we want to find the automorphism group of C_U. Running the standard method on this problem will give a description of this automorphism group. We may then examine this group and note that if any $\sigma \in (C_U)$ exchanges members of the graphs C_1 and C_2 then we may output "Yes", otherwise, we output "No."

To see why this works we note that if the two graphs are isomorphic then the automorphism group will necessarily have a member that will exchange members of the graphs on the overall graph. Such an example of performing a permutation π that satisfies the defintion of isomorphism to C_1 and then π^{-1} to C_2 so that the overall permutation acts on C_U and leaves the graph unchanged. This clearly cannot be the case if they are not isomorphic.

We additionally note that, since the algorithm should only return a generating set for the the automorphism group, that this doesn't cause us to lose any generality. The generating set should have the same properties as discussed above. □

So we have shown that Graph Isomorphism is simply a special case of the Hidden Subgroup Problem!

CLOSE CURTAIN

References

[DW71] I. M. S. Dey and James Wiegold. Generators for alternating and symmetric groups. *Journal of the Australian Mathematical Society*, 12:63–68, 2 1971.

[EHK04] M. Ettinger, P. Hoyer, and E. Knill. The quantum query complexity of the hidden subgroup problem is polynomial. *eprint arXiv:quant-ph/0401083*, January 2004.

[HHR07] Lorenz Halbeisen, Martin Hamilton, and Pavel Ruzicka. Minimal generating sets of groups, rings, and fields. *Quaestiones Mathematicae*, 30(3):355–363, 2007.

[Lom04] C. Lomont. The Hidden Subgroup Problem - Review and Open Problems. *eprint arXiv:quant-ph/0411037*, November 2004.