The Factorization Problem In Computational Group Theory

Spencer T. Parkin

March 2, 2017

The Problem And Solution

For any given element g of a finite permutation group G generated by a set of permutation S, we wish to factor g^{-1} in terms of S. Since $g \in G = \langle S \rangle$, it is clear that for some positive integer k, there exists a sequence of k elements $\{s_i\}_{i=1}^{k}$ taken from S such that

$$g = s_1 s_2 \dots s_k,$$

and therefore,

$$g^{-1} = s_k^{-1} s_{k-1}^{-1} \dots s_1^{-1}.$$

Indeed, there may be many such sequences. How we might come up with such a sequence, however, is not immediately clear. Our goal is to come up with a systematic way of producing such a sequence, even if it is not of minimal length.

Associated with our group G is a set Ω over which each $g \in G$ is defined. That each g is a permutation of this set Ω is to say that g is a one-to-one and onto mapping from Ω to Ω. For a given $\omega \in \Omega$, we define the stabilizer subgroup of G with respect to ω as

$$\text{stab}_\omega(G) = \{g \in G | g(\omega) = \omega\}.$$

This is the set of all permutation of g that stabilize ω. It is not hard to show that this is a subgroup of G. Furthermore, it is a normal subgroup of G. For

any $h \in \text{stab}_\omega(G)$, and any $g \in G$, we have

$$g(h(g^{-1}(\omega))) = g(g^{-1}(\omega)) = \omega.$$

Being normal, we can consider the factor group

$$G/H = \{gH | g \in G\},$$

where $H = \text{stab}_\omega(G)$. We now make the observation that since S generates G, we must have $SH = \{sH | s \in S\}$ generating G/H. To see this, we write the typical element of G/H as

$$gH = \left(\prod_{i=1}^{k} s_i \right) H = \prod_{i=1}^{k} (s_i H).$$

We now let R be a set of coset representatives for G/H, and $[\cdot] : G \to R$ a function mapping any element of g to the coset representative in R representing the coset in G/H that contains g. That is, $|R| = |G|/|H| = |G/H|$, and $G/H = \{rH | r \in R\}$, and for all $g \in G$, $gH = [g]H$ with $[g] \in R$. Computationally, coming up with R is really just the same problem as generating G/H as we would simply use coset representatives to represent each coset of G/H anyway. An implementation of $[g]$ might, for each $r \in R$, check whether $g^{-1}r \in H$.

Armed with all this, and letting g be any element of G, we're now going to begin to address our original question by finding an element q in terms of the generators of S such that $gq \in H$. Clearly, a procedure for such a thing is recursively applicable, for H is simply a permutation group associated with $\Omega - \{\omega\}$, bringing us one step closer to the trivial group $\{e\}$. The trick is that at each step, we must keep track of all new generators in terms of the original generators in S.

So how do we go about finding q? Well, if $g \in H$ (i.e., $g(\omega) = \omega$), then $q = e = ss^{-1}$ for any $s \in S$, and we're done. If not, consider $q = [g]^{-1}$. Clearly $g[g]^{-1} \in H$, since $gH = [g]H$, and furthermore, we know the factorization of $[g]^{-1}$ in terms of the generators in S, because we know such a factorization for $[g] \in R$ by virtue of how we generated R from S.

We are now almost there! All that remains is our ability to continue this process in H with $gq \in H$ and a stabilizer subgroup of H with respect to some element in $\Omega - \{\omega\}$. For that, we need a generating set for H, and this

is where Schreier's Lemma comes into play. According to Schreier, this set of generators is given by

$$\{rs[rs]^{-1} | r \in R, s \in S\},$$

provided $e \in R$. Note that computationally, while we might boil each generator down to its permutation $g \in G$, we would need to keep its expression (or "word") in terms of our original generators in S.

So there we have it! In closing, it might be worth considering why the algorithm thus described does better than the most obvious, naive and impractical approach to solving the factorization problem, which is to simply generate all of G from S, then search for g^{-1} in G. To begin, $|G|$ may be extremely large, making its complete generation take too long and require too much memory. That being said, the stabilizer-chain approach may have its own, similar problems. For a group G of large order, how big is the order of a factor group likely to get? How big is a set of generators likely to get? I'll have to give it a try. I believe it can also be shown that this method does not produce solution sequences anywhere near minimal length.

Generating Groups

For a finite group $G = \langle S \rangle$, how do we go about generating all elements of G using that of S? Letting $G = \{e\}$, we proceed until S is empty. Remove s from S and then, for each $g \in G$, add gs or sg to S if they're not found in G or S, then add s to G.

Clearly this algorithm will terminate, but is it correct? As elements are added to G, keep them in order so that in the end, we have G as an array of elements showing the order that they were added. Then for any $g \in G$, we know that G also contains rg and gr for every $r \in G$ left of g. We also know that this is true for all $r \in G$ right of g. In fine, we should have closure in G. Now since G is a finite subset of a larger group (since the original S was a subset of that larger group), it must be a subgroup of that group. Lastly, can we claim that G is the smallest group containing all original elements of S? Well, since we never added any elements to G unnecessarily, I would have to think so.

This line of reasoning is not terribly rigorous, but it is the best I can come up with for now.