multiple times, we select a different vertex from $\cup_i N(M_i)$ and again explore both paths. The height of this search tree can be bounded by $2q$ because at most q characters can mutate multiple times. The path of height $2q$ in the search tree is an interleaving of q characters that mutate once and q characters that mutate multiple times. Therefore, the size of the search tree is bounded by 4^q.

Combining the two results, the algorithm can be derandomized by solving at most 8^q different instances of Step 3 while traversing the while loop 8^q times for a total running time of $O(144^q + 8^q nm^2)$. This is, however, an over-estimate. Consider any iteration of the while loop when M_j is replaced with $M0$ and $M1$. If a state in character c is unfixed and therefore guessed, we know that there are two associated mutations of character c in both $M0$ and $M1$. Therefore at iteration i, if q'_i states are unfixed, then $\texttt{penalty}(M0) + \texttt{penalty}(M1) \leq \texttt{penalty}(M_j) - q'_i$. At the end of the iteration we can reduce the value of q used in Step 2 by q'_i, since the penalty has reduced by q'_i. Intuitively this implies that if we perform a total of q' guesses (or enumerations) at Step 2c, then at Step 3 we only need to solve Steiner trees on $q - q'$ characters. The additional cost $2^{q'}$ that we incur results in reducing the running time of Step 3 to $O(18^{q-q'} + qnm^2)$. Therefore the total running time is $O(72^q + 8^q nm^2)$. ■

D. Improving the Run Time Bounds

In Lemma 4.9, we showed that the guesses performed at Step 2c of the pseudo-code in Figure 4 do not affect the overall running time. We can also establish a trade-off along similar lines for Step 2a that can reduce the theoretical run-time bounds. We now analyze the details of such a trade-off in the following lemma:

Lemma 4.10: The algorithm presented above runs in time $O(21^q + 8^q nm^2)$.

Proof:

For the sake of this analysis, we can declare each character to be in either a 'marked' state or an 'unmarked' state. At the beginning of the algorithm, all the characters are 'unmarked'. As the algorithm proceeds, we will mark characters to indicate that the algorithm has identified them as mutating more than once in T^*

We will then examine two parameters, ρ and γ, which specify the progress made by the derandomized algorithm in either identifying multiply mutating characters or reducing the problem to sub-problems of lower total penalty. Consider the set of characters S such that for all $c \in S$,