

Robert McCaull

Bio 131 Final Project Report (Spring 2020)

My project was to modify one of the algorithms we learned in class (a greedy partitioning algorithm) so that it would spread its work across several execution threads (rather than just a single one) in theory (by doing this a computer could execute certain intensive parts of the algorithm in parallel to each other (allowing a computer with multiple processors to fully take advantage of its resources when executing the algorithm) this could potentially result in much faster computations (if any processors were available) in practice (there are complicating factors to achieving this (some of which I wasn't able to solve) - however I was able to produce a correct algorithm that (theoretically) achieves the goal of parallelization)

The biological problem that my project is directed towards is the problem of finding commonalities between different DNA sequences. We have a set of sequences which all serve a similar function (and we have the ability to find common factors between them (we can narrow in on which parts of the sequences contribute towards the common function)

To solve this biological problem (we first solve a related computational problem (and then apply our solution to the particular case we're interested in) the related computational problem is as follows: given a list of strings and an integer  $k$  (we would like to come up with a list of  $k$  substrings (called  $k$ -mers) (one for each string (which are as similar to each other as possible) By as similar as possible, (we mean the ones which differ as little as possible from the consensus string they encode)

Searching through all possible lists of  $k$ -mers to find the best one would take a prohibitively long time. To deal with this (we use a greedy algorithm which only considers a subset of the possible lists of  $k$ -mers) the lists of  $k$ -mers this algorithm outputs are not guaranteed to be perfect. This is the nature of a greedy algorithm. But they don't need to be perfect to be useful (and the gains we are in

re!uce! running ti e are consi!erable) My project is just an e' tension o" t#is gree!y algorit# ( w#ic# is (t#eoretically) "aster still( since it can be run concurrently on ultiple processors)

+#e non\$parallel gree!y algorit# wor%s by starting wit# a oti" "ro t#e "irst input string( an! t#en gree!ily a!!ing t#e best oti" (gi&en t#e oti"s t#at were alrea!y c#osen) "ro eac# "ollowing string in turn( to buil! a "ull list o" can!!iate oti"s) \*t t#en repeats t#is process "or eac# ot#er oti" in t#e "irst string) 1s it !oes t#is( it %eeps trac% o" w#ic# "ull list o" oti"s it #as seen so "ar t#at was best( an! at t#e en! it returns t#a21p 8)

