In this paper, Dr. Rivest outlines an algorithm for the selection of the next node to expand in a game tree. Typically, when searching a game tree, it is infeasable to expand every node in the tree to find the optimal next move. Therefore, we define a heuristic which attempts to find the best available move on the board without searching to end-game. The difficulty that remains is deciding how far into the tree we should search.

Searching to a fixed depth on every turn has a big disadvantage known as the "horizon affect." A move can look like the optimal move for four or five turns, but then something can happen that drastically changes the outcome of the game. For example, in chess, using your queen to capture an opponents piece could look like a good strategy for a few moves in the future, but it could expose your queen to potential capture. This could mean the difference between winning and loosing. Even iterative deepening can suffer from the horizon affect if the algorithm is not given enough time.

The next idea is to choose which node to expand next based on the volatility of the leaf nodes. That is, we should expand the node that is mostly likely to have the greatest affect on the choice of move. In his paper, Dr. Rivest does three things to choose the best node to expand next. First, he uses generalized mean values to approximate the min and max functions. These functions are better for testing sensitivity because their partial derivatives are easier to analyze. Second, he argues that if x is the parent of y in the game tree, then analyzing the sensitivity of node x to the variations at node y can be accomplished by analyzing the partial derivatives of the generalized mean values (for large enough exponent). Therefore, the node that we should expand next is the one that maximizes his sensitivity function.

Unfortunately, there is one downside to this approach, and it affects the results of the Connect Four experiments. The problem is that dealing with all the extras (p^{th} powers, p^{th} roots, logarithms, and simply needing to keep track of more data in the game tree) causes a tremendous slowdown. For example, if time was the limiting factor (like we have in this project), then this new method was inferior to a standard implementation of minimax with alpha-beta pruning and iterative deepening. However, if "move" calls are the limiting factor, then this new method worked very well compared to the standard approached. The advantage comes from the fact that this new approach expanded nodes in an order based on volatility, while the minimax algorithm with alpha-beta pruning and iterative deepening expanded nodes in a naive order and could have potentially expanded every node in a level before being able to prune.

Overall the results of this paper are very interesting to me. Throughout this project I wanted a way to expand nodes in the game tree in an order that made more sense than simply taking them in the order they are presented. If the overhead of these extra computations can be handled in a more efficient way, then this new method has lots of potential.