I used MTD(F). per Wikipedia

“MTD(f) calls the zero-window searches from the root of the tree. Implementations of the MTD(f) algorithm have been shown to be more efficient (search fewer nodes) in practice than other search algorithms (e.g. NegaScout) in games such as chess [1], checkers, and Othello.”

I believe MTD(F) or (Memory-enhanced Test Driver with node n and value f), is an effective alternative to alpha-beta pruning alone because it only zero-window alpha-beta searches, with a "good" bound (variable beta). To find the minimax value, MTD(f) calls alpha-beta a number of times, converging towards the exact value. It uses a transposition table to store and retrieve the previously searched portions of the tree reducing the overhead of re-exploring already explored parts of the search tree.

Looking at the table:

Algorithm	100 Games
	MTD(F)
MinMax	52%
Random	90%
Greedy	63%

We can see that MTD(F) beat random by 90% and even minmax 52% of the time. This is a significant improvement, but the improvement comes in search time, rather than accuracy.

Per Wikipedia:
Zero-window searches hit a cut-off sooner than wide-window searches. They are therefore more efficient, but, in some sense, also less forgiving, than wide-window searches. Because MTD(f) only uses zero-window searches, while Alpha-Beta and NegaScout also use wide window searches, MTD(f) is more efficient.

This helps with the efficiency that I am looking for using MTD(F).
When running tests I chose not to change the fair_matches flag. I feel because I am starting the first point randomly on the board(rather then the recommended best guess) it essentially moves us forward 1 step negating any benefit of fair_matches.