TEAM CIRAM

Federico Domeniconi

Andrea Gavagna

Federico Stella

FIRST APPROACH

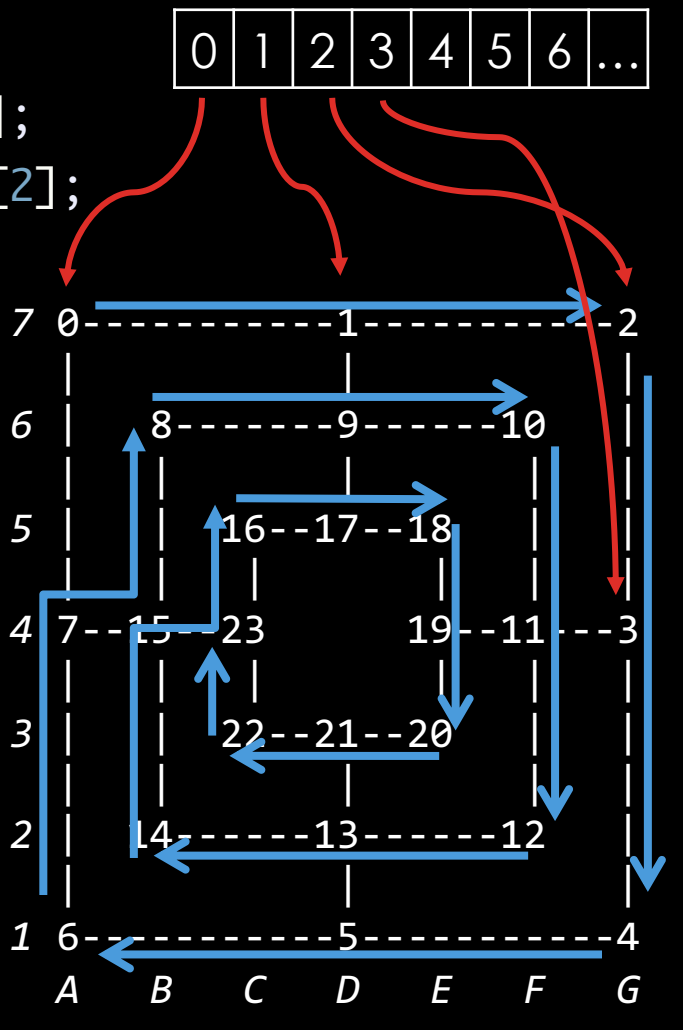
- Quick development
- Chesani's state and action representation
- Minimax (with Alpha-Beta pruning and Killer Heuristic)

- ~280K nodes/second

  w/ i7-4790K @4.4GHz

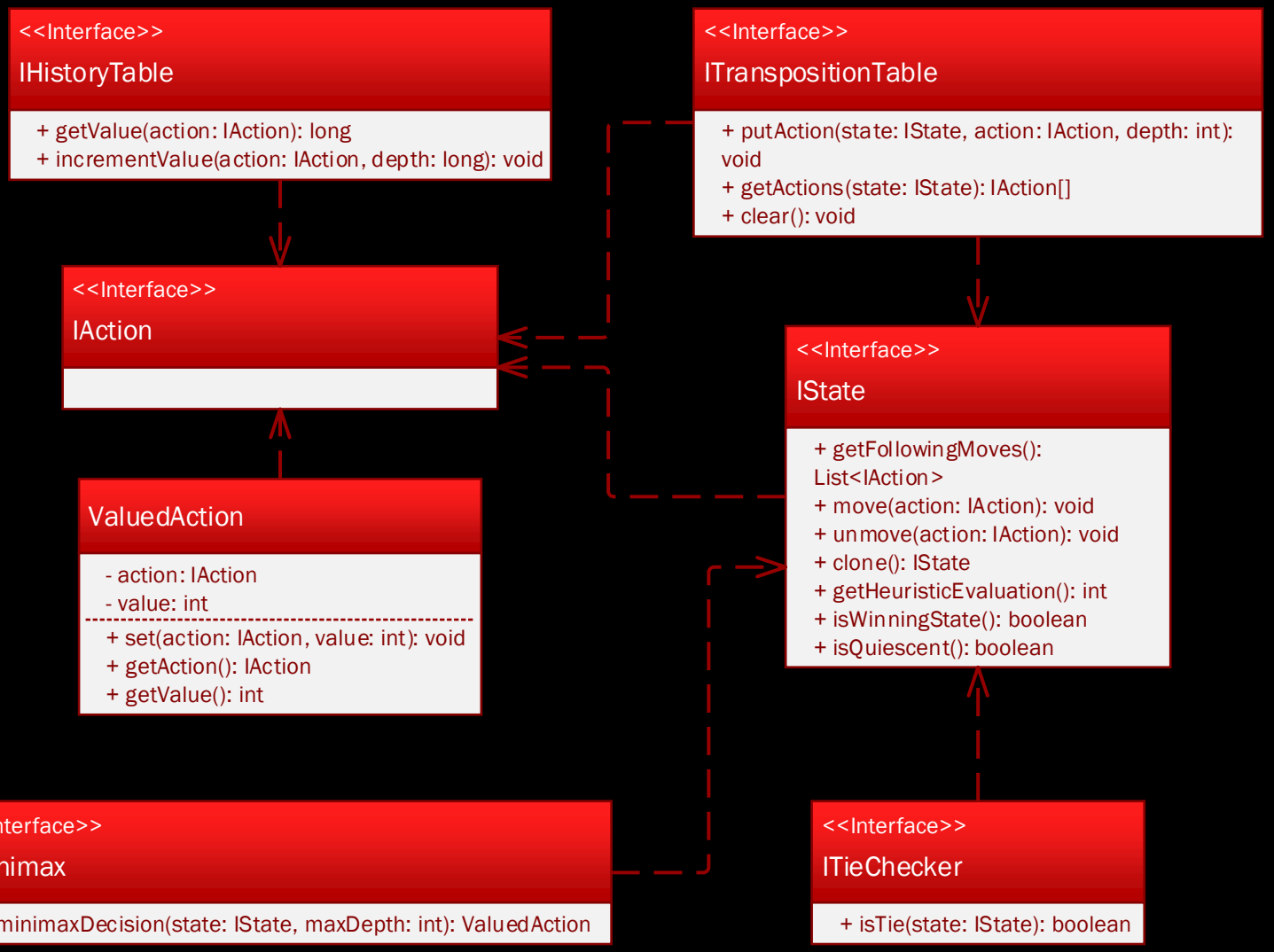
A BETTER APPROACH

```
public class BitBoardState implements IState {
    private int[] board = new int[2];
    private byte[] checkersToPut = new byte[2];
    private byte[] checkersOnBoard = new byte[2];
    public byte playerToMove;
    private byte gamePhase;
}
```



- ~5M nodes/second
  w/ i7-4790K @4.4GHz
- ~x20 quicker

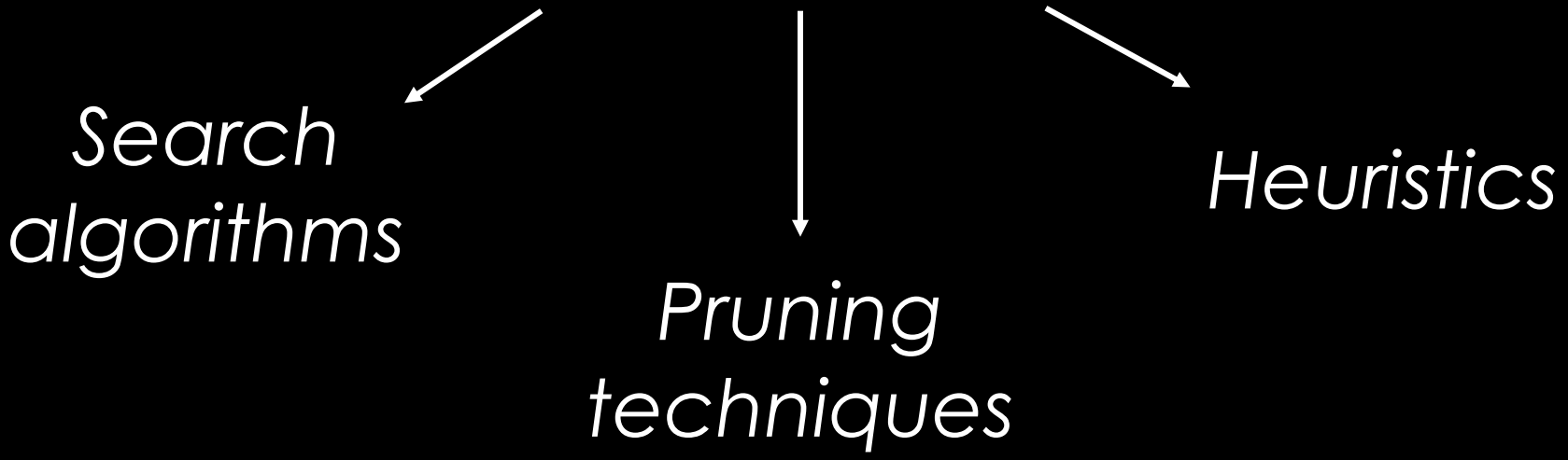
BASIC APPLICATION ARCHITECTURE



DONE

- ✓ *Efficient State and Action representation*

NEXT IN LINE



SEARCH ALGORITHMS

- Minimax
  - AlphaBeta
    - NegaScout
      - MTDf
        - MTDf Variant

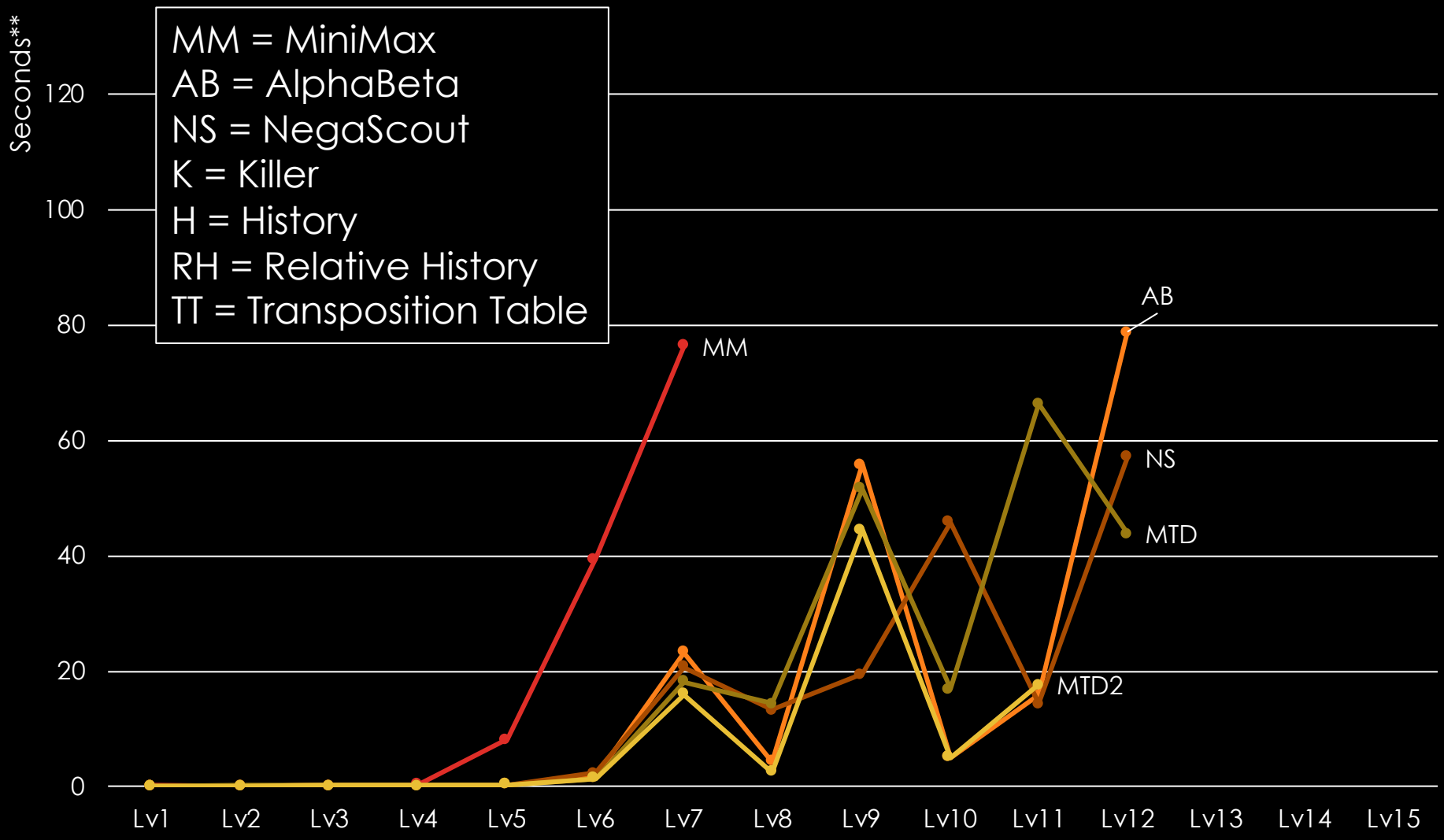
PRUNING TECHNIQUES

- Killer Heuristic
  - History Heuristic
    - Relative History Heuristic
      - Transposition Tables (<2GB)

HEURISTICS

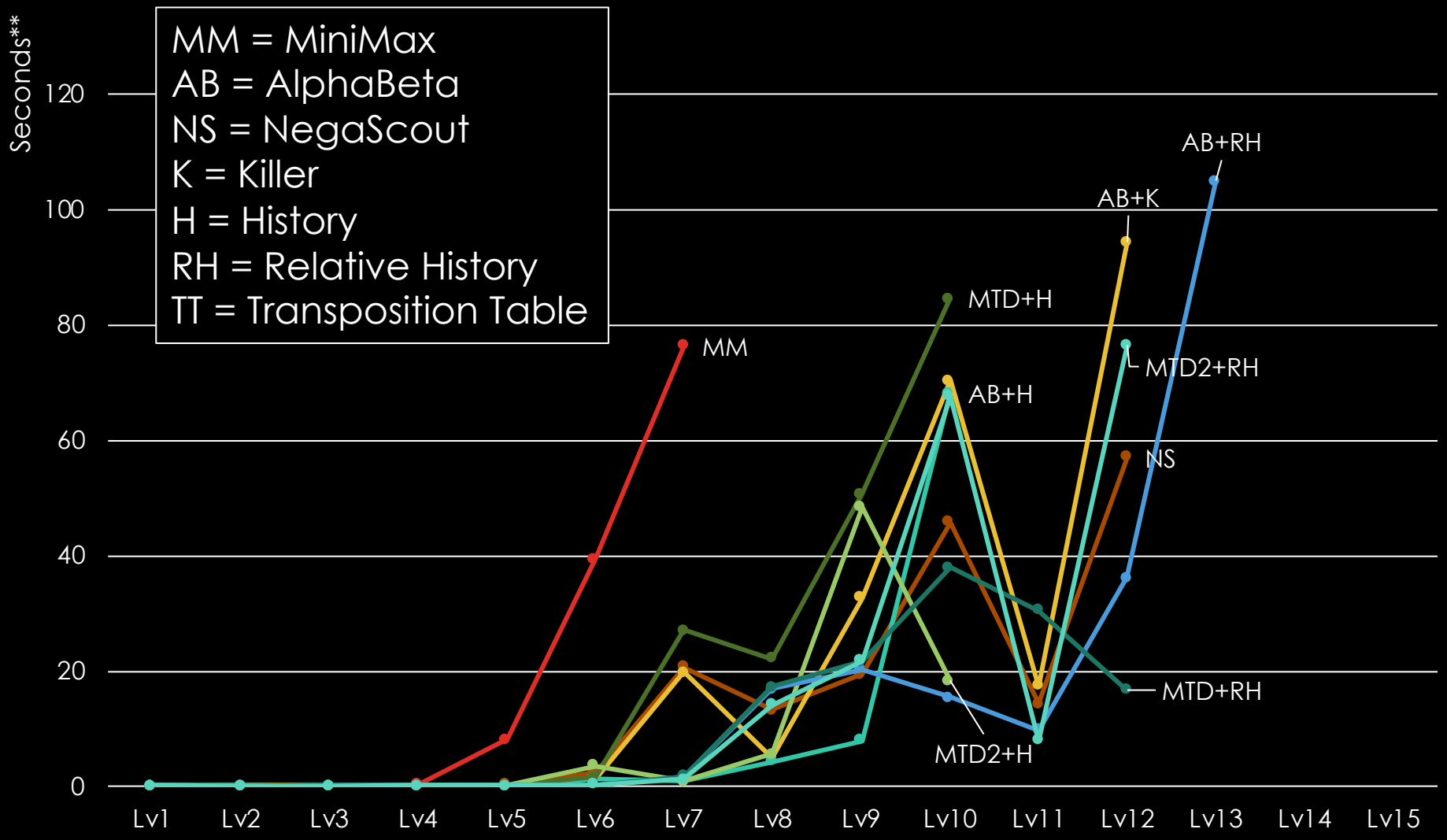
- Petcu-Holban
  - Belasius (Biagio Miceli)
    - Modified Petcu-Holban

STRATEGIES COMPARISON*



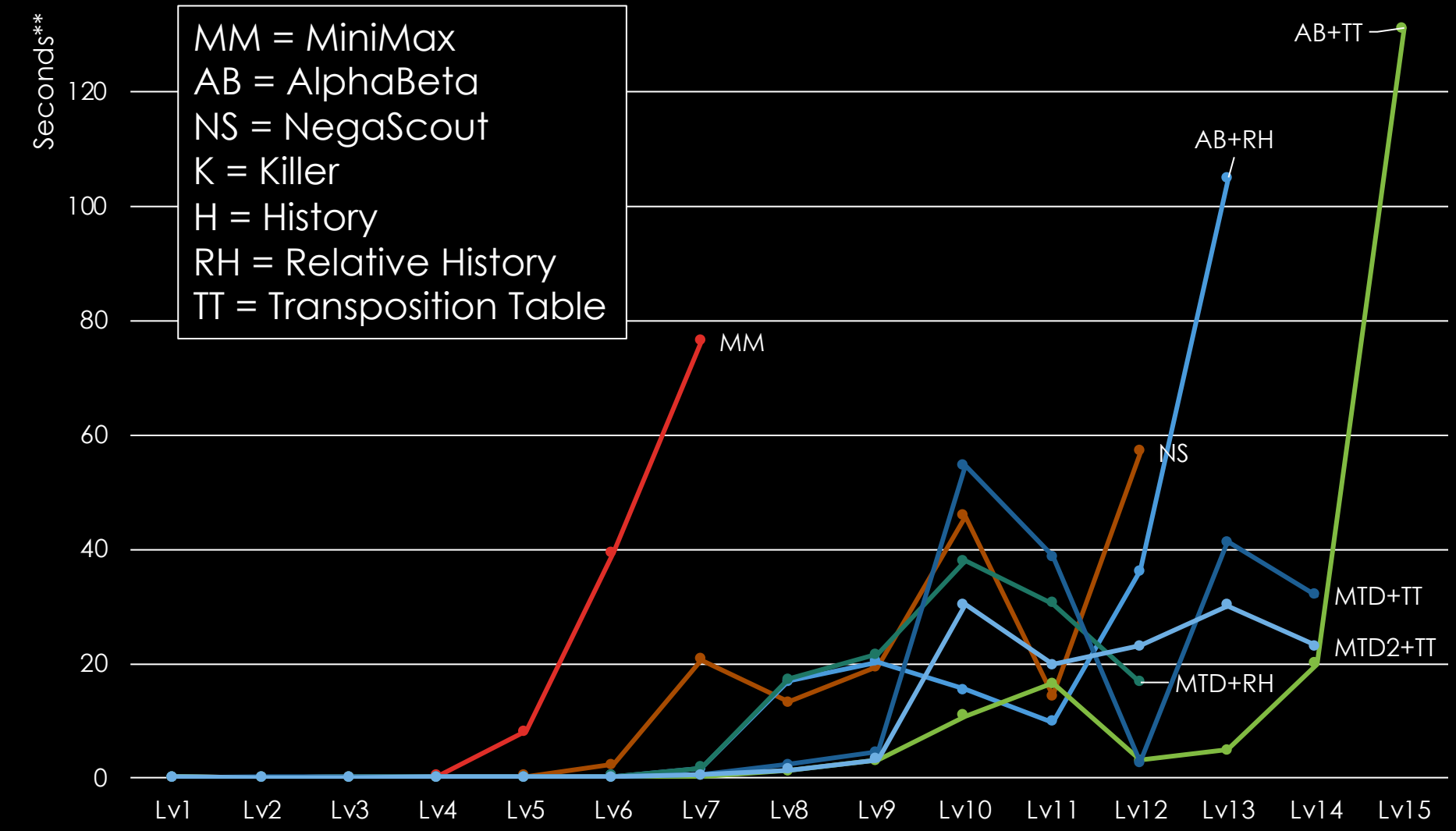
*Limited tests on 5 states. K, H and RH perform much better in real games.
**w/ i7-7500U @3.5GHz

STRATEGIES COMPARISON*



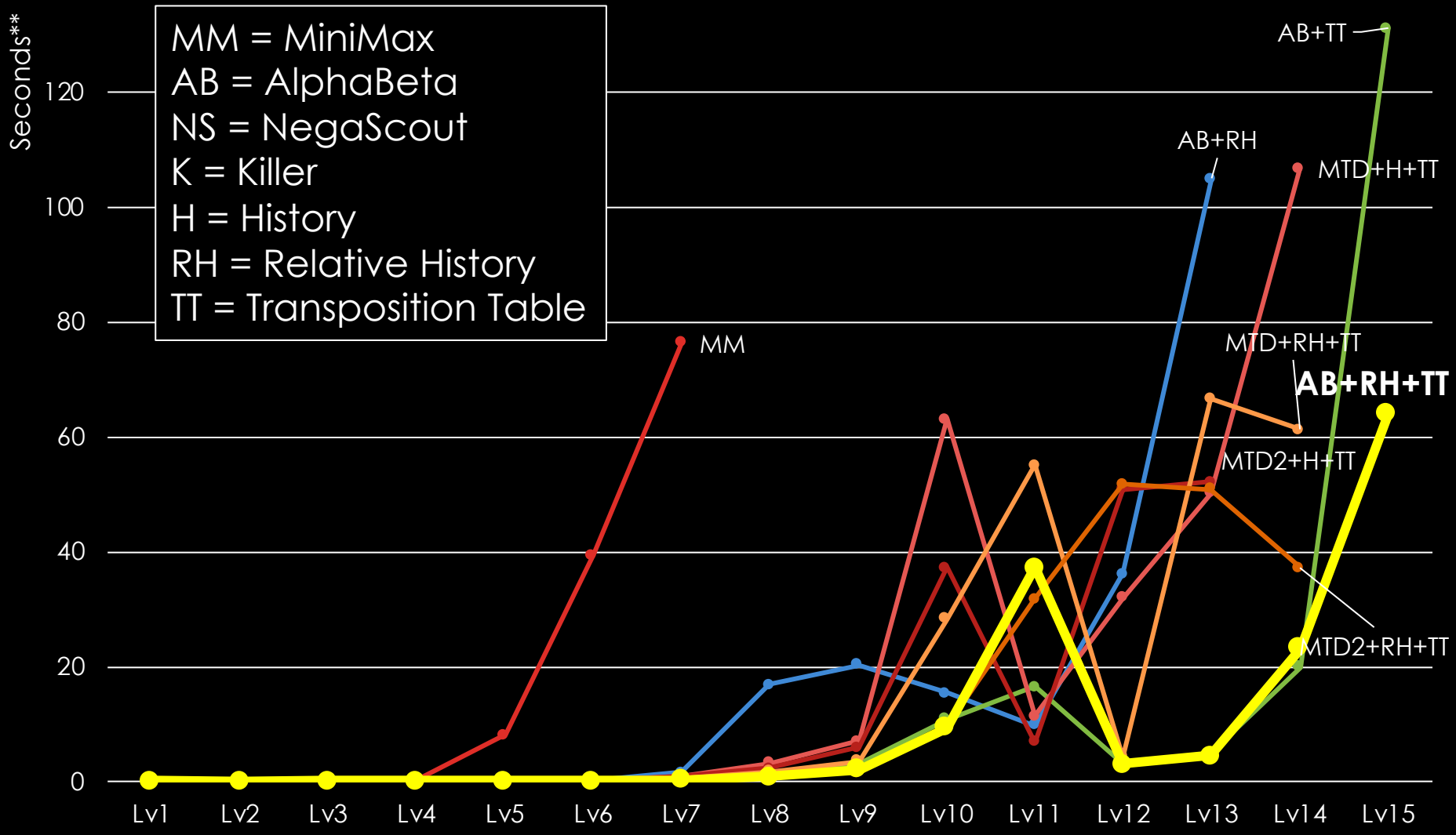
*Limited tests on 5 states. K, H and RH perform much better in real games.
**w/ i7-7500U @3.5GHz

STRATEGIES COMPARISON*



*Limited tests on 5 states. K, H and RH perform much better in real games.
**w/ i7-7500U @3.5GHz

STRATEGIES COMPARISON*



*Limited tests on 5 states. K, H and RH perform much better in real games.
**w/ i7-7500U @3.5GHz

BEST STRATEGY

AlphaBeta + TT + RelativeHistory

(+ Modified Pectu-Holban)

HISTORY HEURISTIC

Used to sort possible moves with a most-pruning-first strategy

a1a4	a1b1	a4a7	a4a1	a4a1g7	…
0	0	0	0	**$0+n^2$**	0

If "a4a1g7" cuts the tree with a remaining depth of n, then sum n^2 or 2^n

This prioritizes good moves that prune a lot, but also average moves that prune just a little but happen to be used very often.
Not all moves occur with the same frequency.
⇒ Add a table that has information about the occurring frequency of every move.
⇒ This is called Butterfly Heuristic, updated when a move is used.

The resulting value for each action is calculated as a relative value: HistoryValue/ButterflyValue.
This architecture is called Relative History Heuristic

TRANSPOSITION TABLE

Used to store the 2 best-pruning moves for each state.
The state space is NOT a true graph => no bounds in table

```
private LRUMap<Long, BitBoardEntry> transpositionTable;


class BitBoardEntry {
    private BitBoardAction[] actions = new BitBoardAction[2];
    private byte depth;
}
```

- Hash → Full 57 bit state representation, no collisions
- Update strategy → Depth + Always
- Symmetries → 4 + Color symmetry (hash is computed as minimum hash of all 32 possible symmetrical states)

TO SUM UP

- Transposition Table → Best 2 moves *per state*
- Relative History Heuristic → Ordering strategy for the rest of the moves

Complementary strategies!

ADDITIONAL IMPLEMENTED STRATEGIES

- Quiescence (considered irrelevant, not used)
- Opening move-set for both colors
- Parallel search (slower than single-threaded search, not used)
- Idle-time search (only to handicap opponent)

TEAM
CIRAM

THANK YOU!