AI coding agents are increasingly capable of optimizing code performance, though their effectiveness depends on whether they are guided by human intuition or integrated into automated benchmarking environments.

AI agents use a combination of pattern matching, static reasoning, and iterative feedback to improve code execution speed and resource usage. Agents can identify suboptimal algorithms and suggest more efficient alternatives, such as replacing a $O(n^2)$ nested loop with a $O(n\log n)$ hash map-based approach. Specialized profiling agents can analyse code to find memory leaks, redundant database queries, or inefficient API calls that slow down applications. Agents often suggest improvements for memory usage, data structures, and the elimination of "dead code" or unnecessary checks.

Research shows that AI agents largely use the same optimization patterns as humans (e.g., caching, unrolling loops). While humans typically use benchmarks to prove an optimization works, AI agents often rely on static reasoning (predicting it should be faster) rather than empirical evidence. This can lead to "hallucinated" speedup claims. AI agents may lack awareness of broader system constraints or specific hardware optimizations, which can lead to changes that break functionality in edge cases.