Literature Review: Chess Programming Techniques

Comprehensive Citation Analysis

Citation	Key Contributions	Relationship to Other Works	Role in Review
Shannon (1950) *Programming a Computer for Playing Chess*	• Frames chess as computational problem • Identifies 10^{120} position space impossibility • Proposes evaluation function $f(P)$ • Introduces Type A/B search strategies • Recognizes horizon effect • Suggests material, mobility, pawn structure, king safety as evaluation factors	• Foundation for all subsequent work • Directly influences Knuth & Moore's formalization • Evaluation principles adopted by all HCE engines • Type B strategy leads to selective extensions • Basically is the foundational paper that drived all of chess programming	Establishes theoretical foundations; defines core problems (intractable search space, need for evaluation, horizon effect) that motivate decades of innovation
Knuth & Moore (1975) *Analysis of Alpha-Beta Pruning*	• Formalizes minimax as $F(p), G(p)$ • Proves alpha-beta optimality • Demonstrates deep cutoffs advantage • Shows F_3 improvements impossible • Best case: $W^{\frac{D}{2}} + W^{\frac{D}{2}} - 1$ nodes	• Builds on Shannon's minimax concept • Validates alpha-beta vs. all alternatives • Referenced by Marsland, Brange, all modern pruning work • Later challenged by AlphaZero's MCTS success	Provides mathematical proof of alpha-beta's theoretical optimality within minimax framework; establishes performance bounds guiding implementation decisions
Björnsson & Marsland (2000) *Review of Game-Tree Pruning*	• Comprehensive pruning taxonomy • TPT essential for pruning & move ordering • Null-move, futility pruning techniques • Quiescence search importance • Move ordering heuristics (killer, history)	• Synthesizes Knuth & Moore theory with practice • Connects to Shannon's Type B strategy • Foundation for Brange, Tesseract implementations • Complements Zobrist's hashing	Bridges theory and practice; comprehensive survey of pruning landscape from foundational to advanced techniques
Zobrist (1970) *New Hashing Method*	• Incremental hashing via XOR • 2-4 XOR operations per update • 0.000003% collision rate with 1B positions • Enables efficient transposition tables	• Enables Marsland's TPT recommendations • Universal adoption (Stockfish, LC0, all engines) • Complements magic bitboards for efficiency • Used with all search techniques • Now is the standard for hashing	Solves position hashing problem; enables memory-aided search techniques critical to modern engine performance

Citation	Key Contributions	Relationship to Other Works	Role in Review
Nasu (2018) *NNUE: Efficiently Updatable Neural Networks*	• Introduces NNUE architecture • Incremental accumulator mechanism • HalfKP feature representation • CPU-optimized neural evaluation • Quantized integer arithmetic (8-16 bit)	• Bridges HCE limitations with neural learning • Alternative to AlphaZero's GPU approach • Integrated into Stockfish (2020) • Complements alpha-beta search	Enables neural evaluation in traditional engines; represents hybrid paradigm combining alpha-beta efficiency with learned patterns
Silver et al. (2017) *Mastering Chess and Shogi (AlphaZero)*	• MCTS + deep neural networks • Self-play reinforcement learning • 80K pos/s vs Stockfish's 70M pos/s • Defeats Stockfish 28-0-72 despite 1000× fewer nodes • Policy + value network architecture	• Challenges Knuth & Moore's alpha-beta dominance • Contrasts with Shannon's HCE approach • Influences Leela Chess Zero development • Motivates NNUE as CPU alternative	Demonstrates paradigm shift from exhaustive to selective search; proves learned evaluation can surpass handcrafted heuristics
Fiekas (2018) *Finding Hash Functions for Bitboard Move Generation*	• Magic bitboard hash functions • Multiplicative hashing technique • Brute-force magic number discovery • Constant-time move generation • PEXT comparison shows only 2.3% speedup	• Extends bitboard work (1970s origins) • Compared with BMI2 PEXT instructions • Used by Stockfish, all modern engines • Dives deeper into magic bitboards	Solves sliding piece move generation; provides constant-time lookup essential for competitive performance
de Groot (1965) *Thought and Choice in Chess*	• Coins "progressive deepening" term • Studies chess cognition • Iterative search concept • Human pattern recognition insights	• Theoretical foundation for iterative deepening • Influences Marsland's search enhancements • Applied by all modern engines • Cognitive basis for search strategies	Introduces iterative deepening concept; provides cognitive basis for human-inspired search strategies
Bijl & Tiet (2021) *Exploring Modern Chess Engine Architectures*	• Empirical architecture comparison • 0x88 nearly matches bitboards in Perft (46.5 vs 48.8 Mn/s) • Bitboards dominate in evaluation, not move generation • Move generation only 10% of engine time	• Challenges conventional bitboard justification • Validates magic bitboards vs PEXT findings • Tests techniques from Marsland, Fiekas • Supports NNUE need for learned evaluation	Provides empirical validation of architectural choices; reveals nuanced performance tradeoffs challenging conventional assumptions

Citation	Key Contributions	Relationship to Other Works	Role in Review
	• Parameter tuning improves 15% win rate		
Brange (2021) *Evaluating Heuristic and Algorithmic Improvements (KLAS)*	• KLAS engine case study • MVV-LVA: 68.5% execution time reduction • Iterative deepening + PV: 28.7% faster • Lazy SMP: 33.1% speedup (4 cores) • Empirical technique validation	• Implements Marsland's techniques • Validates Zobrist, magic bitboards • Compares YBWC vs Lazy SMP • Supports Tesseract findings	Provides quantitative performance impacts; demonstrates real-world effectiveness of theoretical techniques
Vrzina (2023) *Piece By Piece Building a Strong Engine (Tesseract)*	• Tesseract engine development journey • Quiescence Search: 62.5% faster, +1206 eval score • PSTs: biggest eval impact (5640→8255) • 16-bit move encoding: 50% faster generation • Lazy SMP: 40% speedup • LMR challenges (score drop 8584→8124)	• Validates techniques from Marsland, Brange • Confirms bitboard + mailbox hybrid benefits • LMR challenges align with AlphaDeepChess • Supports Shannon's evaluation factors	Documents practical implementation journey; reveals implementation challenges (LMR sensitivity) and empirical performance gains
Columbia et al. (2023) *Chess Move Generation Using Bitboards*	• Bitboard move generation analysis • Legal vs pseudo-legal comparison • Lookup table techniques • Modern engine preferences documented	• Extends 1970s bitboard origins • Complements Fiekas's magic bitboards • Aligns with Stockfish implementation choices	Clarifies move generation approaches; explains pseudo-legal preference in modern competitive engines
Rasmussen (2004) *Parallel Chess Searching and Bitboards*	• Parallel search fundamental challenges • Young Brothers Wait Concept (YBWC) algorithm • Lazy SMP concept • 9.2× speedup on 22 processors (not 22×) • Synchronization overhead analysis • Tree size growth roughly linear	• Theoretical foundation for Brange, Vrzina work • Explains Stockfish's YBWC→Lazy SMP switch • Connects to alpha-beta's sequential nature • Identifies parallelization limits	Identifies fundamental parallelization challenges; explains sublinear scaling and inherent tradeoffs in parallel search
Girón & Wang (2025) *AlphaDeepChess: Alpha-Beta Engine*	• AlphaDeepChess engine development • LMR implementation challenges and failures • YBWC performance degradation	• Validates Rasmussen's YBWC concerns • Confirms LMR sensitivity (also in Tesseract) • Contrasts with Stockfish's successful implementation	Highlights implementation difficulty gap; shows advanced techniques require sophisticated supporting systems to succeed

Citation	Key Contributions	Relationship to Other Works	Role in Review
	• Synchronization overhead issues • Move ordering criticality demonstrated	• Highlights implementation difficulty	
Świechowski et al. (2022) *Monte Carlo Tree Search: A Review*	• MCTS comprehensive review • UCB1 selection strategy • MCTS variants and modifications • Application across domains • Pre-neural MCTS performance	• Contextualizes AlphaZero's MCTS approach • Documents historical MCTS weakness vs alpha-beta • Explains neural networks as game-changer • Provides theoretical background	Provides MCTS theoretical background; explains why neural guidance was necessary for MCTS superiority over alpha-beta
Stockfish Team (2025) *NNUE Documentation*	• NNUE implementation in Stockfish • HalfKAv2 architecture details • Accumulator mechanism explained • Integration with alpha-beta search • 100+ Elo gain documented	• Implements Nasu's NNUE architecture • Alternative to AlphaZero's GPU approach • Combines with traditional alpha-beta techniques • Represents current state-of-the-art	Documents successful hybrid approach at highest competitive level; proves neural evaluation compatible with classical search
Plaat (1997) *A Minimax Algorithm Faster than NegaScout*	• MTD(f) algorithm (Memory-enhanced Test Driver) • Multiple minimal window searches • Convergence to minimax value • Strong transposition table dependency • Alternative to PVS/ NegaScout	• Alternative to PVS/ NegaScout approach • Builds on Knuth & Moore's alpha-beta bounds • Less widely adopted than PVS due to TPT dependency • Requires strong hashing infrastructure	Illustrates continued algorithmic refinement; shows innovation continues despite theoretical optimality proofs of alpha-beta

Table 1: Analysis of Sources based on their contributions