The primary goal of my research is to develop computational techniques for competent and timely decision-making in complex domains. My interests span multiple areas of Artificial Intelligence (AI) including single-agent and adversarial planning, combinatorial reasoning, and machine learning. My thesis work focuses on understanding and improving algorithms for game-playing. Games such as Chess and Go have occupied the human intellect for millennia and have proven to be an irresistible lure for AI researchers down the years. The appeal of games as research testbeds is twofold. On the one hand, the properties and rules of games can be described and represented succinctly. At the same time, optimal decision-making in most games is exceedingly difficult (PSPACE-complete or worse). Moreover, many problems such as logistical planning, formal verification and hardware debugging have natural formulations as games between competing agents. Thus, progress in algorithms for game-playing has the potential to yield significant payoffs in several real-world domains.

## Background

It is estimated that the Chess game tree has about $10^{120}$ nodes. While computing the optimal move to play by comprehensively searching this tree is impossible given our current technology, a Chess playing program still needs to arrive at some *reasonable* decision. For several decades, the standard approach to taming this complexity has been the *Minimax* search algorithm, which builds a search tree representing every possible sequence of moves and counter-moves of some pre-determined length[1]. A heuristic function is used to evaluate the leaf nodes of this tree, and these values are propagated up the tree to select the best action at the root node. This brute-force approach has been highly successful in a number of domains including Othello, Checkers and Chess, where the standard of computer play now surpasses that of humans. However, the method suffers from two major problems:

> **Poor Scaling Behavior:** Domains with large branching factors limit the extent to which Minimax-style approaches can "look ahead" into the future of the game, thereby diminishing their efficacy.
>
> **Over-Reliance on Domain-Specific Knowledge:** The heuristic function, which typically requires heavy input from a human designer, plays a critical role in the success of Minimax-based agents.

The game of Go is a domain where Minimax approaches collide against both these limitations. In a typical Go position, an agent is faced with the task of choosing from a hundred or more possible moves. Moreover, effective heuristics have been difficult to hand-design (or learn using machine learning techniques) for Go. Thus, for a long time, the standard of Go playing programs remained stagnant at the level of a weak amateur. This has changed dramatically in the past few years with the development of a new stochastic search algorithm termed Upper Confidence bounds for Trees (UCT). Unlike Minimax-based approaches, UCT expands the search tree in an opportunistic manner. It focuses on areas of the search space that are deemed most promising, and grows highly asymmetric trees that probe deeper into interesting parts of the game (as illustrated in figure 1). Moreover, UCT employs *random playouts*, i.e., random completions of games, to estimate the strength of a position. This obviates the need for extensive engineering of domain-specific heuristics. UCT's prowess in Go was first demonstrated by the success of programs like CrazyStone and MoGo, with the latter becoming the first computer program to attain master-level play in $9 \times 9$ Go. UCT

[1] Modulo enhancements, such as alpha-beta pruning

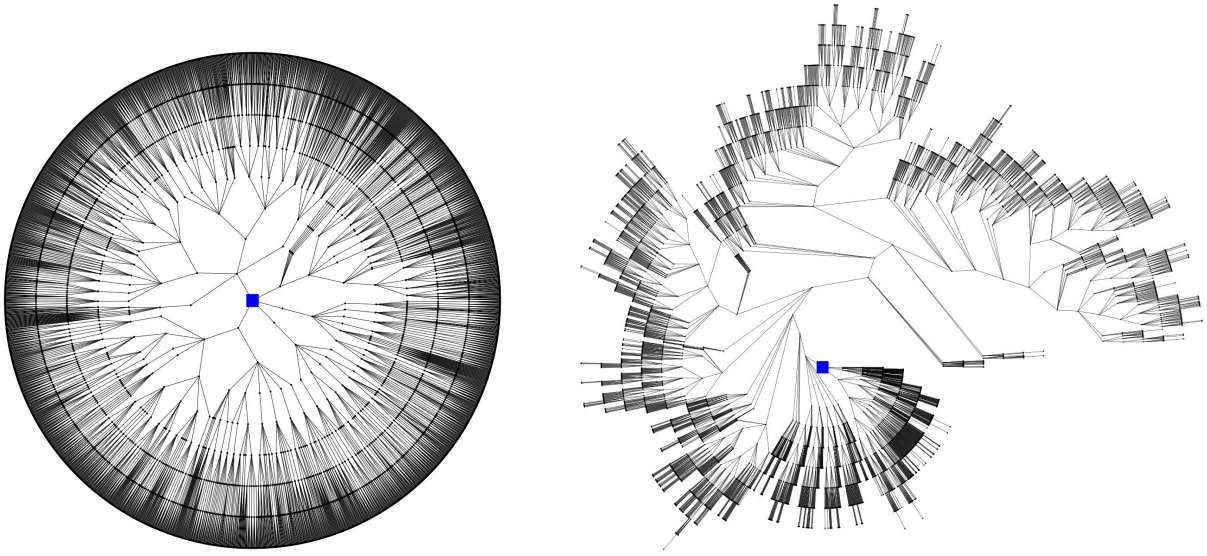

Figure 1: On the left is an example of a search tree expanded by Minimax search (with alpha-beta pruning), and on the right is a tree expanded by a UCT search. The blue square is the root node. Nodes at the same depth from the root are placed at the same radius.

has since been successfully employed in other challenging domains such as planning in real-time, stochastic and partially-observable domains, and general game-playing.

## Past Work [2]

### On the Impact of Trap States

Despite the impressive accomplishments of UCT, there are significant gaps in our understanding of the behavior of the algorithm. While UCT guarantees perfect decision-making in the limit, it is known that this convergence can take super-exponential time. This negative result makes UCT's successes appear quite puzzling. Moreover, anecdotal evidence has suggested that UCT does not fare well at Chess, where Minimax and its variants have been the "gold standard" for decades. Our work offers an explanation for this phenomenon [3]. We showed that *shallow trap states* — positions from which a game is irretrievably lost against an opponent searching moderately deep — are sprinkled throughout the search space of Chess. In Go, such positions are not encountered until the end-game phase. We experimentally demonstrated that sampling-based approaches to tree search such as UCT frequently misevaluate the utility of such trap states and thus have difficulty differentiating good moves from bad.

### UCT for Chess

Given that a prevalence of shallow traps has an adverse effect on the success of UCT, a natural question to ask is whether the basic algorithm can be modified to outperform Minimax-based approaches at Chess. We pursued this question in [4]. Here, we showed that UCT *can* outperform Minimax if both algorithms are made to operate in a completely knowledge-free manner, with

[2]The work described here was carried out in collaboration with Alessandro Previti, Ashish Sabharwal, Marco Schaerf and Bart Selman.

heuristic guidance coming in the form of random playouts. This raises a natural follow-up question — what is the actual information content of random playouts? Can the outcomes of games played by two random players really produce useful evaluations of the strengths of positions? It is highly unintuitive that such an estimation scheme could be valuable, since games between random players tend to follow a very different trajectory as compared to games between strong players. Surprisingly, we discovered that the outcomes of random playouts *do* contain some useful, albeit limited, information. Finally, to gain a more analytical understanding of UCT's behavior, we studied its performance on simple abstracted game tree models, with implanted winning strategies for one player. We presented results that demonstrated how UCT's excessive optimism could lead it to "freeze" onto sub-optimal lines of play and thereby have an adverse effect on its performance.

### Hybrid Search Strategies for Game-Playing

To understand why UCT is so successful in practice, we carried out a study that compared its behavior to the relatively well-understood Minimax algorithm [6]. Initially, this effort was hindered by the fact that in most domains, either UCT is too weak to attain a reasonable standard of play (like in Chess), or Minimax is too weak to compete with UCT (as in Go). We circumvented this problem by situating our study in the game of Mancala, which as we demonstrated, is the first known domain where both approaches produce a reasonable standard of play with virtually *no enhancements* to the basic algorithms. We carried out a dissection of UCT in this domain and showed that:

1. selective search methods *can* outperform complete search methods in some adversarial domains, contradicting decades-old views to the contrary,
2. given a fixed computational budget, it is more prudent to build large trees with just a few random playouts per leaf node, and
3. when a good heuristic function is available, combining UCT with a Minimaxing backup operator can offer a significant performance boost over the traditional implementation of the algorithm.

Based on these insights, we introduced a new hybrid game-playing algorithm that outperforms both vanilla UCT and Minimax in the game of Mancala. **This work was recognized with an honorable mention for the best student paper at the $21^{st}$ International Conference on Automated Planning and Scheduling.**

## Ongoing and Future Work

### Understanding UCT's Behavior in Synthetic Game Trees

In the short-term, my research interests lie in characterizing the types of search spaces where UCT outperforms Minimax. In ongoing work [5], we are developing synthetic games that mirror the complexities of real games but are still amenable to rigorous analysis. The ultimate goal of this work is to identify a set of features of a search space that can be used to predict *a priori* whether UCT is likely to succeed in a given domain. In addition to contributing to a more fundamental understanding of the reasons for UCT's practical successes, this work could also pave the way for ensemble approaches to game-playing and planning.

### Beyond Game Playing

While UCT's most high-profile successes have been in game-playing applications, it also has the potential to advance the state-of-the-art in other challenging combinatorial domains. For example, we have demonstrated that UCT can exploit problem structure to outperform complete search methods (like the DPLL algorithm) on some families of Boolean Satisfiability (SAT) instances [1, 2]. In ongoing work, we are exploring the possibility of using UCT-based approaches for solving MaxSAT (the optimization version of the SAT problem) instances and Quantified Boolean Formulas (QBFs). Since current solvers for these problems scale poorly with increasing input size, there is potential for UCT-inspired approaches to make significant contributions here.

## Supervising Undergraduate Research

Based on my experiences doing research as an undergraduate and mentoring a team of students building an intelligent poker-playing agent, I have come to appreciate the benefits of exposing students to Computer Science (CS) research. When restricted to the confines of the typical CS curriculum, most students only ever work on problems with well-defined answers. Participating in a research project allows them to supplement this experience by pursuing open-ended questions, thus giving them a taste of "real science." Research work also helps students cultivate and hone useful professional skills. For example, writing up and presenting results to external audiences polishes a student's communication skills, while doing a literature review teaches a student how to be an independent learner. Research experiences are also particularly useful for students grappling with the question of whether they should continue onward to graduate school. Thus, I believe that my role as an educator does not simply end with the walls of the traditional classroom. I intend to fully encourage and support undergraduate research work to produce more accomplished and complete computer scientists. I believe that the natural appeal of AI, and game-playing in particular, will help me attract students to my research projects, and I look forward to supervising student projects and theses.

## References

[1] Alessandro Previti, Raghuram Ramanujan, Marco Schaerf, and Bart Selman. Applying UCT to boolean satisfiability. In *14th SAT*, Ann Arbor, MI, June 2011.

[2] Alessandro Previti, Raghuram Ramanujan, Marco Schaerf, and Bart Selman. Monte-Carlo style UCT search for boolean satisfiability. In Roberto Pirrone and Filippo Sorbello, editors, *AI*IA*, volume 6934 of *Lecture Notes in Computer Science*, pages 177–188. Springer, 2011.

[3] Raghuram Ramanujan, Ashish Sabharwal, and Bart Selman. On adversarial search spaces and sampling-based planning. In *20th ICAPS*, pages 242–245, Toronto, Canada, May 2010.

[4] Raghuram Ramanujan, Ashish Sabharwal, and Bart Selman. Understanding sampling style adversarial search methods. In *26th UAI*, Catalina Island, CA, July 2010.

[5] Raghuram Ramanujan, Ashish Sabharwal, and Bart Selman. On the behavior of UCT in synthetic search spaces. In *Workshop on Monte Carlo Tree Search Methods: Theory and Applications*, Freiburg, Germany, June 2011.

[6] Raghuram Ramanujan and Bart Selman. Trade-offs in sampling-based adversarial search. In *21st ICAPS*, Freiburg, Germany, June 2011.