# An Automated Technique for Drafting Territories in the Board Game Risk

**Richard Gibson** and **Neesha Desai** and **Richard Zhao**
Department of Computing Science, University of Alberta
Edmonton, Alberta, T6G 2E8, Canada
{rggibson | neesha | rxzhao}@cs.ualberta.ca

## Abstract

In the standard rules of the board game Risk, players take turns selecting or "drafting" the 42 territories on the board until all territories are owned. We present a technique for drafting territories in Risk that combines the Monte Carlo tree search algorithm UCT with an automated evaluation function. Created through supervised machine learning, this function scores outcomes of drafts in order to shorten the length of a UCT simulation. Using this approach, we augment an existing bot for the computer game Lux Delux, a clone of Risk. Our drafting technique is shown to greatly improve performance against the strongest opponents supplied with Lux Delux. The evidence provided indicates that territory drafting is important to overall success in Risk.

## Introduction

For the past few decades, computer games have been a popular platform for artificial intelligence research. Many successful computer programs (bots) play at expert levels in games such as checkers (Schaeffer et al. 1996), Othello (Buro 1997), and chess (Campbell, Hoane Jr., and Hsu 2002). These are popular two-player games where both players share complete information about the game states. Such games are handled well by classical minimax search with alpha-beta pruning and some evaluation function (often hand-tuned) for intermediate states.

However, games involving more than two players (multi-player) are generally less understood. In this paper, we describe how to create stronger bots for playing the multi-player game Risk by Hasbro Inc. In the standard rules of Risk, the game begins with the players drafting the territories on the board until every territory has an owner. Our work here focuses on improving overall performance by developing an intelligent system for playing the opening territory draft phase of Risk.

The rest of the paper is organized as follows. First, we provide an outline for the rules of Risk and a computer version of the board game, Lux Delux. Second, we discuss related work, including traditional minimax search, its extension to multi-player games, max$^n$, and the Monte-Carlo tree search algorithm UCT. We then describe how our system drafts territories in Risk. Our approach combines UCT with a machine-learned evaluation function to search the game tree up to the end of the drafting phase, where an estimate of the merits of the draft outcomes are propagated back. Next, using a computer version of Risk, we create a bot that uses our drafting technique and pit it against a number of other programs. The empirical results show that our bot outperforms all of its opponents, even those using post-draft strategies identical to our own. Finally, we summarize the results and conclude with some future works.



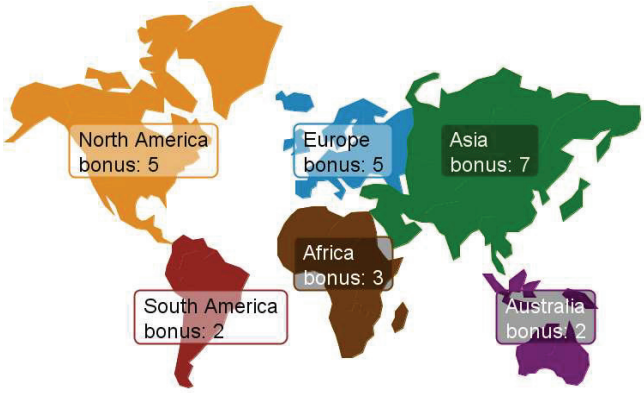



**Figure 1:** The layout of the Risk board and the continent reinforcement bonuses.

## Risk and Lux Delux

Risk is a strategy board game for two to six players where the objective is to occupy all 42 territories on the board. The territories are divided into 6 continents as labeled in Figure 1. In the standard rules, players first take turns selecting the territories until all have been chosen. After players place their initial armies among their selected territories, players then take turns until only one player remains. On a turn, in sequence a player may place reinforcements, conduct attacks on opposing territories, and tactically maneuver armies amongst owned territories. Attacks are resolved by dice rolls, where the number of dice is dependent on the number of attacking and defending armies. A player reduced to zero armies on the board is eliminated from the game. Players receive one reinforcement army for every three territories owned, and receive bonus reinforcements for owning entire continents (as depicted in Figure 1). Full rules can be found

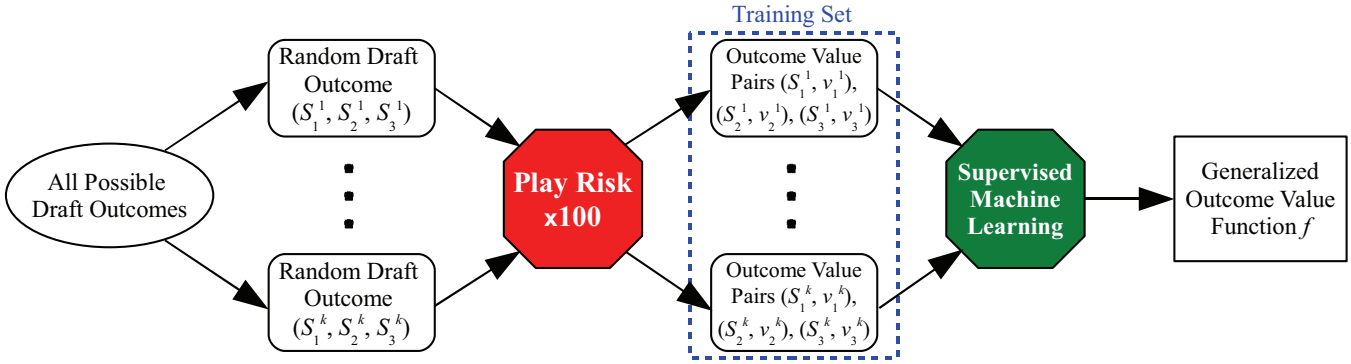



**Figure 2:** The process described for obtaining a general function $f$ for estimating the merit of each feature set in Risk draft outcomes (adapted from (Lee 2004, Figure 5.1)).

on-line (Parker Brothers 1993).

The Lux Delux (Sillysoft 2010) game is a commercialized computer version of Risk, providing several rule-based bots with source code to play against. Each of the included bots has an associated difficulty and a general playing style. For example, "Quo" is a difficult bot that tries to form a cluster of adjacent territories and methodically expand that cluster. We use the Lux Delux environment in all of our computations and experiments.

In this paper, we are only concerned with strategies for playing the opening territory selection phase of Risk. In addition, we focus only on three-player Risk, as this incorporates the multi-player aspect and is arguably more strategically demanding compared to playing with more players. With fewer players, each player must make more selections in the opening draft. Since we are concerned with improving territory selection in Risk, we chose to use the Quo bot's rules for our post-draft play as it appeared to be the strongest bot provided with Lux Delux during preliminary testing (though more thorough tests in our empirical evaluation indicate otherwise).

## Related Work

A widely known approach to game playing is the minimax adversarial search algorithm (Russell and Norvig 2003). To reduce computation time, internal nodes of the game tree are typically evaluated by a heuristic function, which returns a value estimating the merit of the state to the player. These values are then backed up the tree to the root, where the player then chooses the action leading to the child with the maximum propagated value.

While minimax search is traditionally employed in many two-player games, it is not applicable to games with $n > 2$ players. A generalization of traditional minimax search to more players is the max$^n$ algorithm (Luckhart and Irani 1986). At the leaf nodes of the game tree, a heuristic function now estimates a vector of $n$ merits $(v_1, ..., v_n)$, one for each player. At nodes belonging to player $i$, the vector with the highest merit $v_i$ is propagated up to the parent. Thus, players are assumed to be maximizing their own individual payoffs throughout the remainder of the game. An alternative to max$^n$ is the Paranoid algorithm (Sturtevant and Korf 2000), where the active player assumes that the other players are out to minimize his or her payoff with complete disregard to their own benefits. The Paranoid algorithm is essentially equivalent to the minimax algorithm, where the other players are represented as one meta-player (Min) attempting to minimize the individual heuristic value of the active player (Max). Finally, the max$^n$ and Paranoid propagation strategies can be dynamically selected according to the game situation. This is done in the MP-Mix algorithm (Zuckerman, Felner, and Kraus 2009), along with a third strategy called Offensive. On the whole, these algorithms perform best when the branching factor (i.e. action space) is small and when there exists a good evaluation function for approximating the merits of non-terminal states. Unfortunately, when drafting in Risk the first player has 42 actions to choose from initially, and we know of no such evaluation function to use for territory selection.

An algorithm more suited for large actions spaces and requiring no evaluation function is UCT (Kocsis and Szepesvari 2006), a Monte Carlo planning algorithm that has been used effectively in Computer Go (Gelly and Silver 2008) and general game playing, for example in CadiaPlayer (Björnsson and Finnsson 2009). UCT tries to intelligently bias the game tree by simulating games that focus on appealing branches of the tree. At each step, UCT builds upon the sparse game tree from the previous simulations by selecting an action $a$ at state $s$ to expand by

$$a = \text{argmax}_{a'} \left( Q_i(s, a') + c\sqrt{\frac{\log n(s)}{n(s, a')}} \right),$$

where $Q_i(s, a)$ is the estimated value for player $i$ of taking $a$ at $s$, $i$ is the active player at $s$, $c$ is an exploration constant, $n(s)$ is the number of times $s$ has been visited, and $n(s, a)$ is the number of times $a$ has been taken in $s$. The estimate $Q_i(s, a)$ is equal to the average final game value player $i$ received on previous simulations where $s$ was visited and $a$ was taken at $s$. This final game value is typically 1 for a win and 0 for a loss in win-lose type games. On each simulation, UCT adds one new node to the game tree and then randomly plays out the rest of the game. After running numerous simulations from the current state $s$, UCT takes the action $\text{argmax}_a Q_i(s, a)$.

We now describe some previous work related to drafting

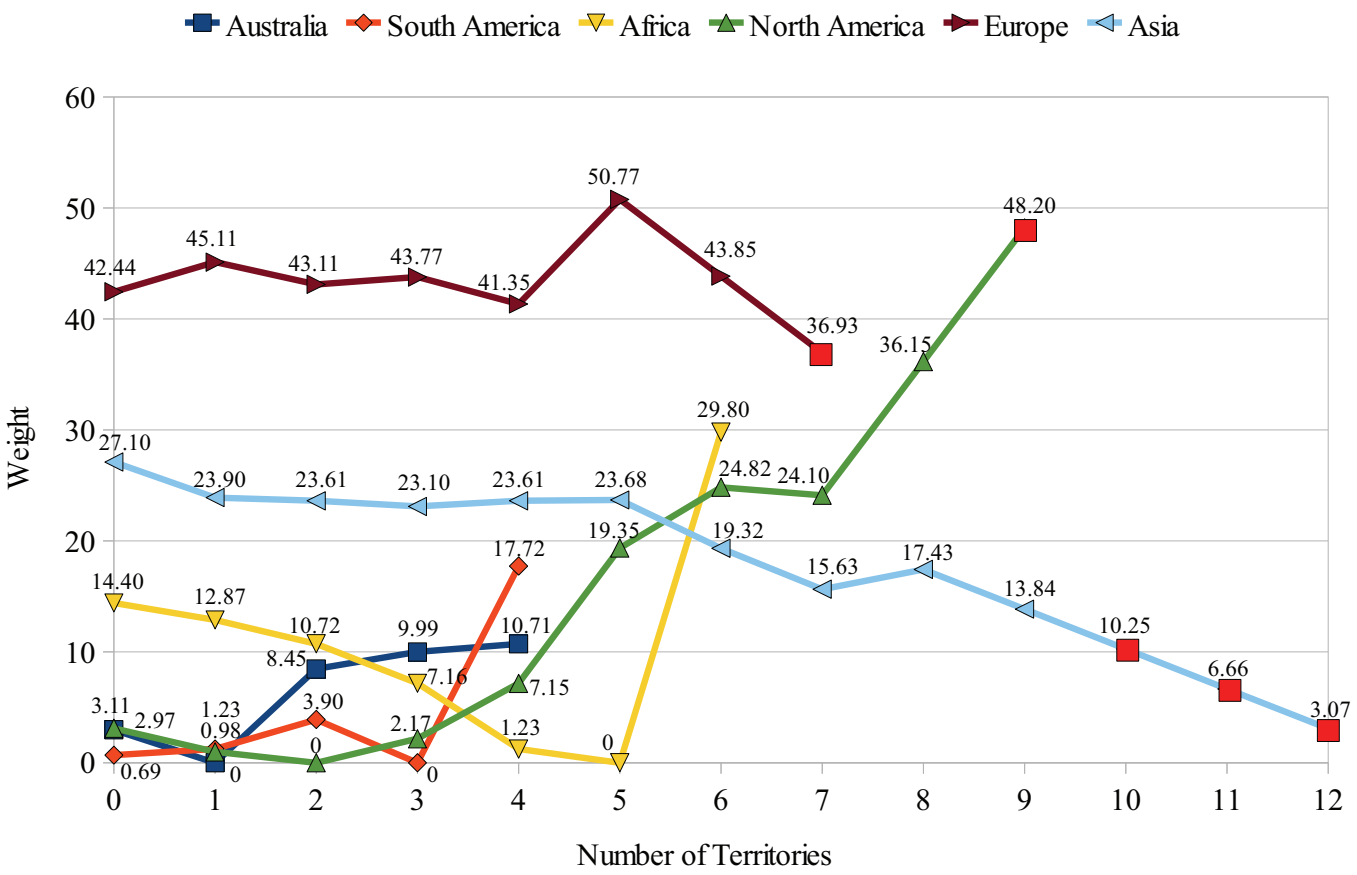



**Figure 3:** The weights for features of type (i), as computed in Weka. Data points with a red square are derived through linear extrapolation.

territories in Risk. One related area is the problem of drafting athletes in sports leagues, to which some progress has been made. Many professional sports use a drafting procedure where teams sequentially select entry-level players based on the teams' current needs and scouting reports of the players' skills. Brams and Straffin (Brams and Straffin Jr. 1979) examine very small drafts where each team has its own order of preference of the entering players. They use a max$^n$-type approach to find optimal drafting strategies, but hint that the approach is ill-advised for larger drafts due to a prohibitively large branching factor. In addition, Fry et. al (Fry, Lundberg, and Ohlmann 2007) solve a deterministic dynamic programming problem through linear programming to determine a team's best drafting strategy and apply it to a fantasy football draft. However, a key ingredient in their model is that each team has positional (quarterback, running back, etc.) needs and must limit its picks to a fixed number of players for each position. For Risk drafting, we are not concerned with restricting territories to competitors in any way.

Finally, playing Risk in particular has garnered some attention from the AI community. Firstly, the program MARS (Johansson and Olsson 2006) is a multi-agent system for Risk that deploys an agent in each territory on the board. The system's actions for placing armies, conducting attacks, and fortifying territories are determined through "bids" submitted by the territory agents. These bids evaluate the territories via hand-tuned features and parameter values. Secondly, Zuckerman, Felner, and Kraus used Risk as a testbed for their MP-Mix algorithm (Zuckerman, Felner, and Kraus 2009). To do so, they restricted the branching factor to only 3 promising moves, where a move represented an entire sequence of territories to conquer. However, both of these approaches use a Risk variant where territories are randomly assigned to begin the game rather than selected by the players. We are not aware of any previous work regarding the drafting phase of Risk.

## Our Approach

We use UCT as our main algorithm for drafting territories in Risk, since max$^n$ and other traditional search algorithms are not practical when there are many possible actions and no known evaluation functions. One issue with applying UCT simulations, however, is that Risk is not well-suited to numerous runs of the entire game as games typically last many turns and dice rolls introduce a lot of randomness in the results. Furthermore, if all players play randomly in Risk, there is no guarantee that the game will even end. Since UCT simulates games randomly beyond previously stored states in its search tree, the algorithm is unlikely to complete more than a small number of simulations, leading to poor estimates of the possible actions. To increase the rate at which

**Table 1:** The weights for features (ii), (iii), and (iv) as computed in Weka.

| Feature | Weight |
|---|---|
| First to play | 13.38 |
| Second to play | 5.35 |
| Each unique enemy neighbor | -0.07 |
| Each pair of friendly neighbors | 0.96 |

simulations can be performed, we instead simulate only the initial drafting portion of the game, which has a fixed length (the number of territories remaining to be selected) regardless of random play. This is done by evaluating each draft outcome with a numerical score that estimates our probability of winning the entire game of Risk from that draft outcome. These scores are generated off-line using supervised machine learning, which we now describe.

Our learning technique is closely related to Lee's work (Lee 2004) of combining single-agent heuristic search with a machine-learned fitness function to pick a set of actions from a large library. Lee's objective is to find a small set of actions which allow the agent to behave as close to optimal as possible in a Markov decision process, relative to having access to the entire library of actions. Our approach here can be seen as an extension of Lee's work from a single-agent problem to a competitive multi-agent problem, where we replace single-agent search with an adversarial method, UCT.

We manually identified a number of features that are tactically important in draft outcomes of Risk. Many of these features are inspired by the calculation of highest "bids" (Johansson and Olsson 2006). For each player, our features are described by: (i) for each continent, the number of territories owned in that continent; (ii) when the player plays in the turn order; (iii) the number of distinct territories owned by other players which border owned territories (enemy neighbors); and (iv) the number of pairs of owned territories which are adjacent (friendly neighbors). Our process for estimating the merit of a feature set is depicted in Figure 2. First, we collect many draft outcomes, each obtained by randomly assigning all 42 territories evenly among the 3 players. Then, for each draft outcome, 100 games are played to completion where each player follows the post-draft strategy of the Quo bot provided with the Lux Delux software package. Each player $i$, $i = 1, 2, 3$, has a set of features $S_i$ associated with each outcome. Each feature set is assigned a value $v_i \in \{0, 1, ..., 100\}$ equal to the number of games that player $i$ eventually won from the associated draft outcome. This provides a collection of feature set value pairs $\{(S_i, v_i)\}$ of size equal to three times the number of draft outcomes collected. Next, supervised machine learning is applied to obtain a general function

$$f : S \mapsto v \in \mathbf{R}$$

that estimates the merit of the feature set $S$ when following Quo's post-draft strategy. Finally, for a draft outcome $Z = (A_1, A_2, A_3)$ with $A_i$ being the set of player $i$'s territory selections, we calculate $v_i = f(S_i)$ where $S_i$ is the feature set associated with $A_i$, and define the value of the draft outcome $Z$ for player $i$ to be

$$V_i(Z) = v_i^+ / \left(v_1^+ + v_2^+ + v_3^+\right),$$

where $v_i^+ = \max(0, v_i)$. Thus, player $i$ attempts to maximize the estimated merit $v_i$ of his or her selections while minimizing those of the opponents.

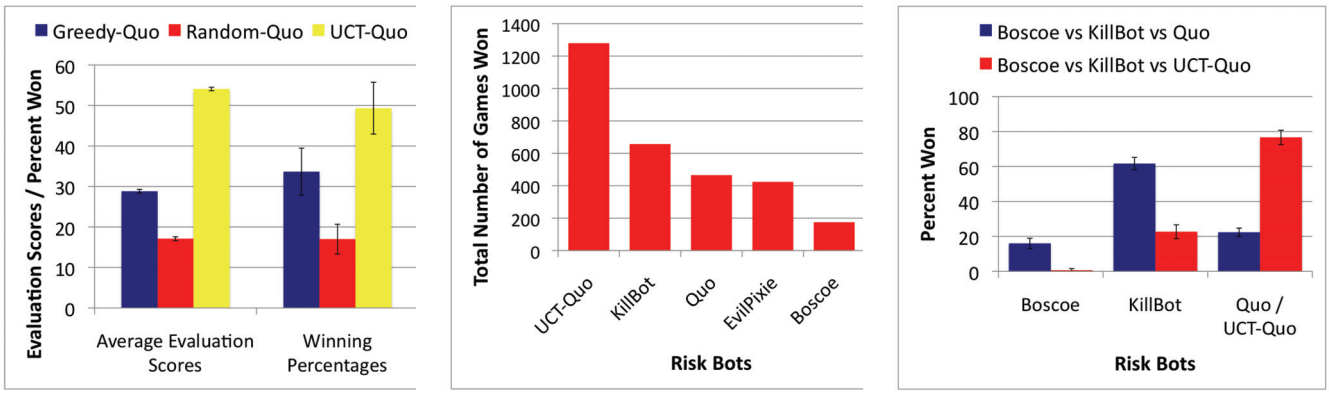
**Figure 4:** An example of a draft outcome. The territories picked by players 1, 2, and 3 are blue, green, and red respectively. Territories connected by a line are also considered adjacent or bordering. Modified from Lux Delux.

Our general function $f$ was computed off-line from 7364 random draft outcomes for a total of 22092 $(S, v)$ pairs. We used Weka (Hall et al. 2009) to weight each individual feature using Weka's linear regression classifier (with no attribute selection), where features (i) and (ii) were represented as nominal features and features (iii) and (iv) were numeric. Thus, $f(S)$ can be calculated by simply summing the weights, displayed in Figure 3 and Table 1, of the features present in $S$. However, there are some features of type (i) which did not appear in any of the 22092 feature sets because of their unlikeliness of occurring through random drafting; for instance, there were no cases where one player owned all 9 territories of North America. The weights of these features were calculated through linear extrapolation of the closest two continent counts for the associated continent.

We now show how to use Figure 3 and Table 1 to compute $V_i(Z)$ for the draft outcome given in Figure 4. Player 1 (blue) has 4 territories in Australia, 0 in South America, 2 in Africa, 1 in North America, 3 in Europe, and 4 in Asia. In addition, player 1 has 18 distinct enemy neighbors and 11 pairs of friendly neighbors. Assuming player 1 is first to play, $f(S_1)$ is computed as

$$\begin{aligned} f(S_1) &= [10.71 + 0.69 + 10.72 + 0.98 + 43.77 \\ &\quad + 23.61] + [13.38 + 18(-0.07) + 11(0.96)] \\ &= 113.12, \end{aligned}$$

where the values in the first set of brackets are from Figure 3 and the second set of brackets are from Table 1. This gives $f(S_1) = 113.12$. We can similarly compute the values of the feature sets for player 2 (green) and player 3 (red) as $f(S_2) = 89.47$ and $f(S_3) = 119.65$ respectively, where player 2 is second to play. Finally, $V_i(Z)$ is computed




(a) (b) (c)

**Figure 5:** Risk bot performances. Error bars indicate 95% confidence intervals.

via $V_i(Z) = f(S_i)/\left(f(S_1) + f(S_2) + f(S_3)\right)$, giving us $\vec{V}(Z) = (0.35, 0.28, 0.37)$.

## Empirical Evaluation

We created a "UCT-Quo" bot by replacing the drafting rules of the Quo bot provided in Lux Delux with our approach of combining UCT and the machine-learned evaluation function. The Quo bot is one of the four strongest bots provided by the game, according to the game designers. For comparison, we also created a "Random-Quo" and a "Greedy-Quo" bot that replace Quo's drafting rules with a random drafting strategy and a "greedy" strategy, respectively. The greedy strategy works as follows: At any state $\hat{s}$ in the draft (i.e. a partial assignment of territories to players), we can obtain temporary feature sets $\hat{S}_1, \hat{S}_2, \hat{S}_3$ for each player of the current selections made, using the same features as described in the previous section. We can then calculate $\hat{V}_i(\hat{s}) = f(\hat{S}_i)/(f(\hat{S}_1) + f(\hat{S}_2) + f(\hat{S}_3))$ of the state $\hat{s}$, where $f$ is our machine-learned evaluation function. Greedy picks the territory which leads to the state $\hat{s}$ with the greatest $\hat{V}_i(\hat{s})$ value, breaking ties by selecting randomly among the territories with the fewest number of unoccupied neighbors. For example, on an empty board, we can conclude from Figure 3 and Table 1 that Greedy will always open by picking a territory in Europe with exactly three neighbors (the fewest of all territories in Europe) because Europe has the biggest increase in score from 0 to 1 territory. Greedy can be seen as a 1-ply lookahead search that uses $\hat{V}_i(\hat{s})$ as an evaluation function, despite $f$ being designed to only evaluate draft outcomes.

We ran several preliminary matches consisting of only UCT-Quo bots with different UCT exploration constants and found that $c = 0.01$ provided the best results. Thus, UCT-Quo used $c = 0.01$ in all later experiments. For each pick, UCT-Quo ran 3000 simulations from the root decision point, taking less than one second per pick on an Intel Core 2 Duo CPU at 2.00 GHz with 4GB of RAM. We also pitted UCT-Quo against the four difficult bots in the game Lux Delux: Quo, EvilPixie, KillBot, and Boscoe.

We present the experimental results in Figure 5. For each combination of 3 bots that were tested, 50 rounds of games were played to completion where 1 round consisted of 6 games, one for each of the 3! turn orderings of the bots. All games were run with the Lux Delux settings "selected countries" and "placed armies" turned on, and "cards" turned off.

Firstly, Figure 5(a) shows that UCT-Quo came first place against Greedy-Quo and Random-Quo in terms of both the average draft evaluation score from our machine-learned outcome evaluator (leftmost bars), and in number of actual games won (rightmost bars). This suggests that UCT is a good choice for drafting Risk territories. Furthermore, since Random-Quo wins only a small percentage of games despite having an identical post-draft strategy as its opponents, we have evidence that the drafting phase in Risk is crucial to success. Finally, note that the similarity in shape between the leftmost and rightmost bars suggests that our draft outcome evaluator does a good job at approximating Quo's chances of winning from an outcome.

Next, a five-bot round robin tournament consisting of UCT-Quo and the four difficult Lux Delux bots was played and the total number of wins for each bot in the tournament is shown in Figure 5(b). The five bots were competing in three-player matches where each bot competed against all combinations of two other bots for a total of 10 combinations. The results clearly show that UCT-Quo is superior to the other four bots as it wins nearly double and nearly triple the number of games that the next closest competitors KillBot and Quo win respectively. Full results of the tournament are provided in Table 2.

Finally, we compare two particular results from the round robin tournament in Figure 5(c). The blue bars show that KillBot was the strongest among Boscoe, KillBot, and Quo, having won roughly 60% of the games against these two opponents. However, when the hand-coded drafting strategy of Quo was replaced by our UCT approach, we see that UCT-Quo did significantly better than Quo against the same two opponents, having won over 75% of the games (the red bars). This result again demonstrates that drafting territories is an important stage of Risk, and that combining UCT with an automated evaluation function is an effective computational approach to this problem.

**Table 2:** Results of the five-bot round robin tournament. Error values give 95% confidence intervals of win totals for each bot per matchup.

| Matchup | No. of Wins | Matchup | No. of Wins |
|---|---|---|---|
| UCT-Quo | 190 ± 14.00 | UCT-Quo | 234 ± 13.56 |
| Quo | 54 ± 12.47 | Quo | 53 ± 12.65 |
| EvilPixie | 56 ± 13.03 | Boscoe | 13 ± 7.31 |
| UCT-Quo | 181 ± 12.80 | UCT-Quo | 190 ± 16.08 |
| Quo | 60 ± 14.82 | EvilPixie | 51 ± 12.98 |
| KillBot | 59 ± 12.10 | KillBot | 59 ± 11.44 |
| UCT-Quo | 254 ± 12.47 | UCT-Quo | 230 ± 12.20 |
| EvilPixie | 36 ± 11.56 | KillBot | 68 ± 11.80 |
| Boscoe | 10 ± 6.26 | Boscoe | 2 ± 2.74 |
| Quo | 105 ± 10.19 | Quo | 126 ± 9.39 |
| EvilPixie | 90 ± 10.84 | EvilPixie | 128 ± 8.00 |
| KillBot | 105 ± 8.04 | Boscoe | 46 ± 10.42 |
| Quo | 67 ± 7.20 | EvilPixie | 63 ± 7.31 |
| KillBot | 185 ± 10.57 | KillBot | 181 ± 10.80 |
| Boscoe | 48 ± 8.84 | Boscoe | 56 ± 9.95 |

UCT-Quo's typical strategy is to claim many neighboring territories in North America and prevent a player from claiming all of South America or Africa by selecting the last available territory in those continents when necessary. This behavior can be understood from Figure 3 as the weights for having many territories in North America are much larger than the weights for having few to no territories there. In addition, there are large increases in weight from having all but one territory to having every territory in South America and Africa. Since UCT-Quo uses $V_i(Z)$ to evaluate draft outcomes, it believes its chances of winning are better if it stops its opponents from beginning the game with all of South America or Africa.

## Conclusions

We presented a technique for drafting territories in the board game Risk. Our approach combines the UCT algorithm with a machine-learned draft outcome evaluator to trim the length of each UCT simulation. The augmented UCT-Quo bot outperformed all of the difficult bots tested within Lux Delux, including the Quo bot itself. Our experiments indicate that the drafting stage is an important part of Risk, and combining UCT with an evaluation function is an effective means of drafting territories. As mentioned, our drafting technique is quite fast (less than a second per pick) and could be appealing to anyone making a bot for a future commercial version of Risk.

We see potential future work stemming from using UCT and a machine-learned evaluator in other types of games, particularly in other drafting-type scenarios such as in fantasy sports leagues. Furthermore, many sports video games include a rookie or fantasy draft component and a UCT approach could provide a more intelligent opponent than current implementations. These and other drafting "games" are convenient because the number of turns each player gets is typically fixed, guaranteeing that UCT simulations will terminate after a predictable amount of time (by possibly using a draft outcome evaluator). Finally, we suspect that a more sophisticated supervised learning technique than simple linear regression may improve the performance of our Risk bot.

## Acknowledgments

We would like to thank Vadim Bulitko for his helpful guidance throughout this project. This research was supported by NSERC, iCORE, and Alberta Ingenuity, now part of Alberta Innovates – Technology Futures.

## References

Björnsson, Y., and Finnsson, H. 2009. Cadiaplayer: A simulation-based general game player. *Computational Intelligence and AI in Games, IEEE Transactions on* **1**:4–15.

Brams, S., and Straffin Jr., P. 1979. Prisoners' dilemma and professional sports drafts. *Am. Math. Mon.* **86**:80–88.

Buro, M. 1997. The Othello match of the year: Takeshi Murakami vs. Logistello. *International Computer Chess Association Journal* **20**:189–193.

Campbell, M.; Hoane Jr., A.; and Hsu, F. 2002. Deep blue. *Artificial Intelligence* **134**:57–83.

Fry, M.; Lundberg, A.; and Ohlmann, J. 2007. A player selection heuristic for a sports league draft. *Journal of Quantitative Analysis in Sports* **3**.

Gelly, S., and Silver, D. 2008. Achieving master level play in 9 × 9 Computer Go. In Fox, D., and Gomes, C. P., eds., *AAAI*, 1537–1540. AAAI Press.

Hall, M.; Frank, E.; Holmes, G.; Pfahringer, B.; Reutemann, P.; and Witten, I. 2009. The WEKA data mining software: An update. *SIGKDD Explorations* **11**.

Johansson, S., and Olsson, F. 2006. Using multi-agent system technology in risk bots. In Laird, J., and Schaeffer, J., eds., *AIIDE*, 42–47. AAAI Press.

Kocsis, L., and Szepesvari, C. 2006. Bandit based Monte-Carlo planning. In *15th European Conference on Machine Learning*, 282–293.

Lee, G. 2004. Automated action set selection in Markov decision processes. Master's thesis, University of Alberta.

Luckhart, C., and Irani, K. 1986. An algorithmic solution of n-person games. In *AAAI-86*, 158–162.

Parker Brothers. 1993. `http://www.hasbro.com/common/instruct/Risk.pdf`. Accessed 23-Apr-2010.

Russell, S., and Norvig, P. 2003. *Artificial Intelligence: A Modern Approach*. Upper Saddle River, New Jersey: Prentice Hall, second edition.

Schaeffer, J.; Lake, R.; Lu, P.; and Bryant, M. 1996. Chinook: The world Man-Machine Checkers Champion. *AI Magazine* **17**:21–29.

Sillysoft. Lux Delux - The best Risk game there is. `http://sillysoft.net/lux/`. Accessed 23-Apr-2010.

Sturtevant, N., and Korf, R. 2000. On pruning techniques for multi-player games. In *AAAI-2000*, 201–207.

Zuckerman, I.; Felner, A.; and Kraus, S. 2009. Mixing search strategies for multi-player games. In *IJCAI*, 646–651.