Monte-Carlo Tree Search for the Game of Scotland Yard[1]

J. (Pim) A. M. Nijssen Mark H.M. Winands

Games and AI Group, Department of Knowledge Engineering
Maastricht University, Maastricht, The Netherlands

This abstract describes how *Monte-Carlo Tree Search* (MCTS) [1, 4] can be applied to play the hide-and-seek game *Scotland Yard*. This game is played by 6 players: 5 seekers and 1 hider. The seekers work together to capture the hider by moving one of their pawns to the location occupied by the hider. The game is played on a map consisting of 199 locations, connected by 4 different transportation types. The hider's location is announced every 5 turns. The seekers always know which transportation type the hider uses.

The basic MCTS algorithm is designed for two-player games with perfect information. When using MCTS in a hide-and seek game, which is a game with imperfect information, the algorithm has to be altered slightly. When we use MCTS in Scotland Yard, the seekers can guess the location of the hider at each iteration of the algorithm and place him on any of the empty locations on the board. This group of locations can be limited by removing the locations where the hider cannot be located, based on the old list of possible locations, the current locations of the seekers, and the type of transportation used by the hider. The list of possible locations is updated every move.

Some of the possible locations are more probable than others. The performance of the seekers could be improved by biasing the possible locations of the hider. This is done by categorizing the possible locations. These categories are numbered from 1 to L, where L is the number of categories. This technique is called *Location Categorization*. The type of categorization is game-dependent. For Scotland Yard, we use a categorization based on the distance of the possible location to the nearest seeker. After the hider performs a move, the possible locations are divided into the different categories. There are two ways to store the information about the possible categories and the category of the location of the hider. In the *general* table, we store for each category the number of times one or more possible locations belonged to the category, n, and the number of times the actual location of the hider belonged to the category, a. This way of storing and using information is similar to the transition probabilities used in Realization Probability Search [7]. In the *detailed* table, for each possible combination of categories, we store how many times the actual location of the hider belonged to each category. There are two different ways to gather the information for these tables: offline and online. When using *offline* information gathering, first a large number of games is played and the information is stored in a file. This information can later be used by the seekers. When using *online* information gathering, the seekers start without any information. At the end of each game, the seekers update the information with the statistics gathered from the last game. The seekers use a vector with length L to select a location for the hider at the start of each MCTS iteration. These values represent the weights of the categories. When using the general table, this vector consists of the values $[\frac{a_1}{n_1}, \frac{a_2}{n_2}, \cdots, \frac{a_L}{n_L}]$. When using the detailed table, this vector is directly taken from the table, by extracting the vector corresponding to the combination of categories. To select a possible location, roulette-wheel selection is used. The size of each possible location on the wheel is corresponding to the value of its category in the vector.

Scotland Yard is a cooperative multi-player game. Therefore, the seekers can be considered as one player, making the game essentially a 2-player game. If in a playout one seeker captures the hider, the playout is considered a win for all seekers and the result is backpropagated accordingly. However, when using this backpropagation rule, we observed that seekers sometimes rely too much on the other seekers and do not make any efforts to capture the hider. For solving this problem, we propose *Coalition Reduction*. If the seeker who is the root player captures the hider, a score of 1 is returned. If another seeker captures the hider, a smaller score, $1 - r$, is returned, where $r \in [0, 1]$.

[1]The full version of this paper is published in: *2011 IEEE Conference on Computational Intelligence and Games*, pp. 158–165.

Beside these two enhancements, we also incorporated some rudimentary domain knowledge [2] by implementing ϵ-greedy playouts [5, 6] for both the hider and the seekers. Move filtering is implemented for the hider, which prunes bad moves from the root node, based on the domain knowledge.

In the experiments we use different types of players. The *Greedy* player uses a straightforward heuristic to select a move. The *Basic-MCTS* player uses MCTS with Upper Confidence Bounds applied to Trees (UCT) [4]. It uses uniform random playouts. The *Greedy-MCTS* player uses ϵ-greedy playouts. It uses the same heuristics as the Greedy player. The MCTS-based players use 10,000 playouts to select a move. In all experiments, 2,500 games are played. The results are given with a confidence level of 95%.

In the first set of experiments we tested the performance of Location Categorization. We let different seekers with and without Location Categorization play against different hiders. The results are given in Table 1. For each setting, Location Categorization improved the performance of the seekers significantly.

Seekers	Hider	With LC	Without LC
Basic-MCTS	Basic-MCTS	54.1% $\pm$ 2.0 (detailed, offline)	47.7% $\pm$ 2.0
Basic-MCTS	Greedy-MCTS	40.2% $\pm$ 1.9 (detailed, online)	33.8% $\pm$ 1.9
Basic-MCTS	Greedy	85.4% $\pm$ 1.4 (detailed, online)	82.4% $\pm$ 1.5
Greedy-MCTS	Greedy-MCTS	77.4% $\pm$ 1.7 (general, online)	72.1% $\pm$ 1.8

Table 1: Win rates of the seekers with and without Location Categorization (LC).

To test the performance of seekers with Coalition Reduction, we let different types of seekers with and without Coalition Reduction play against different hiders. The results are given in Table 2. Except for the Greedy-MCTS player, Coalition Reduction increased the performance of the seekers significantly.

Seekers	Hider	With CR	Without CR
Basic-MCTS	Basic-MCTS	54.0% $\pm$ 2.0 (r = 0.375)	47.7% $\pm$ 2.0
Greedy-MCTS	Greedy-MCTS	73.6% $\pm$ 1.7 (r = 0.125)	72.1% $\pm$ 1.8
Greedy-MCTS + LC (general, offline)	Greedy-MCTS	80.6% $\pm$ 1.6 (r = 0.250)	75.3% $\pm$ 1.7

Table 2: Win rates of the seekers with and without Coalition Reduction (CR).

Finally, to test the performance of the MCTS program, it was matched against a Scotland Yard program on the Nintendo DS. The AI of this program is rather strong [3]. For our seekers, we used ϵ-greedy playouts, Location Categorization (general, offline), and Coalition Reduction (r = 0.25). For our hider, ϵ-greedy playouts and move filtering were used. A total of 50 games were played, where each program played 25 times as the seekers and 25 times as the hider. Out of these 50 games, 33 games were won by our program; 19 games were won as the seekers and 14 as the hider. The Nintendo DS program won 17 games, of which 11 as the seekers and 6 as the hider.

From the results we may conclude that Location Categorization is a robust enhancement for the seekers, which can be used against any type of opponent. Coalition Reduction is a significant improvement for the seekers as well. By using MCTS we are able to create a strong program for playing Scotland Yard. It plays significantly stronger than the commercial program on the Nintendo DS.

References

[1] R. Coulom. Efficient Selectivity and Backup Operators in Monte-Carlo Tree Search. In H. J. van den Herik, P. Ciancarini, and H. H. L. M. Donkers, editors, *Computers and Games (CG 2006)*, volume 4630 of *LNCS*, pages 72–83, Berlin, Germany, 2007. Springer-Verlag.

[2] E.-E. Doberkat, W. Hasselbring, and C. Pahl. Investigating Strategies for Cooperative Planning of Independent Agents through Prototype Evaluation. In P. Ciancarini and C. Hankin, editors, *Coordination Models and Languages (COORDINATION '96)*, volume 1061 of *LNCS*, pages 416–419, Berlin, Germany, 1996. Springer-Verlag.

[3] A. Frackowski. [NDS Review] Scotland Yard DS - Hold.Start.Select, 2011. http://holdstartselect.com/nds-review-scotland-yard-ds, retrieved September 2011.

[4] L. Kocsis and C. Szepesvári. Bandit Based Monte-Carlo Planning. In J. Fürnkranz, T. Scheffer, and M. Spiliopoulou, editors, *Machine Learning: ECML 2006*, volume 4212 of *LNCS*, pages 282–293, Berlin, Germany, 2006. Springer-Verlag.

[5] N. R. Sturtevant. An Analysis of UCT in Multi-player Games. *ICGA Journal*, 31(4):195–208, 2008.

[6] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, USA, 1998.

[7] Y. Tsuruoka, D. Yokoyama, and T. Chikayama. Game-Tree Search Algorithm Based on Realization Probability. *ICGA Journal*, 25(3):132–144, 2002.