MÔ HÌNH BÀI TOÁN ĐÁNH CỜ CÓ ĐỘ PHÂN NHÁNH CAO

Đặng Công Quốc[1] , Nguyễn Đăng Bình[1] , Nguyễn Quốc Huy[2]

[1]Khoa Công nghệ thông tin, Trường Đại học Khoa học, Đại học Huế

[2]Khoa Công nghệ thông tin, Trường Đại học Sài Gòn

Email: dangcongquoc1968@gmail.com, nguyendangbinh@gmail.com, nqhuy@sgu.edu.vn

Ngày nhận bài: 11/10/2017; ngày hoàn thành phản biện: 13/12/2017; ngày duyệt đăng: 8/01/2018

TÓM TẮT

Đánh cờ là một chuỗi lặp đi lặp lại việc chọn lựa nước đi giữa hai người chơi, trạng thái bàn cờ thay đổi khi một nước đi mới được thực hiện. Nói cách khác, đây là bài toán tìm kiếm giải pháp tối ưu trên một trạng thái của bàn cờ. Để chương trình tìm được một giải pháp tối ưu thì tất cả các thành phần chính trong chương trình cần phải tối ưu. Các thành phần chính trong chương trình gồm: cây tìm kiếm đối kháng, hàm lượng giá, đặc trưng của hàm lượng giá, phương pháp lựa chọn đặc trưng. Tùy theo mỗi loại cờ có những đặc điểm riêng và hình thành nên độ khó cũng như độ thú vị cho loại cờ đó, cho nên mỗi bài toán đánh cờ đều là một bài thử và đánh giá về mức độ thành công của việc nghiên cứu trí tuệ nhân tạo. Bài báo trình bày phương pháp xác định độ khó của các loại cờ và các thành phần chính để xây dựng một chương trình đánh cờ.

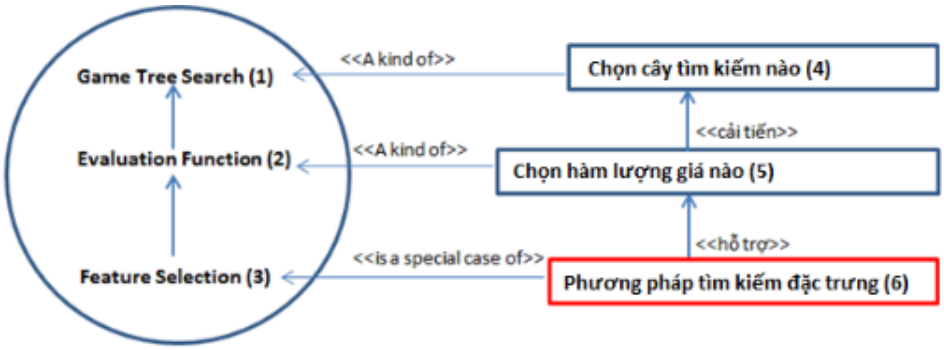
Từ khóa: Chọn lựa đặc trưng, Hàm lượng giá, Học tăng cường, Trò chơi bàn cờ.

1. GIỚI THIỆU

Đánh cờ là một chuỗi lặp đi lặp lại việc chọn lựa nước đi giữa hai người chơi, trạng thái bàn cờ thay đổi khi một nước đi mới được thực hiện. Nói cách khác, đây là bài toán tìm kiếm giải pháp tối ưu trên một trạng thái của bàn cờ. Mức độ tối ưu của việc chọn lựa giải pháp thể hiện tính thông minh của chương trình. Để kiểm tra mức độ thông minh của trí tuệ nhân tạo so với con người, các loại cờ đối kháng thường được dùng để kiểm tra tính thông minh đó. Năm 1997 đánh dấu một cột mốc quan trọng trong ngành trí tuệ nhân tạo khi chương trình Deep Blue của IBM lần đầu tiên chiến thắng nhà vô địch cờ Vua Kasparov, năm 2015 chứng kiến cột mốc quan trọng khác khi chương trình AlphaGo lần đầu tiên đánh thắng nhà vô địch cờ Vây Lee Sedol. Mức độ phức tạp của cờ Vây là 10^{172} trạng thái có thể có của bàn cờ so với 10^{46} trạng thái trong cờ Vua. Các loại cờ đối kháng khác như cờ Tướng, cờ Caro (Go-Moku), cờ

Connect-6 cũng giống như vậy nhưng chỉ khác nhau về độ phức tạp. Một ván cờ sẽ hình thành các trạng thái bàn cờ khác nhau có thể biểu diễn thành một cây tìm kiếm (hay còn gọi là cây trò chơi), xem hình 2.

Như chúng ta đã biết, chương trình đánh cờ đều phải dựa trên cây tìm kiếm. Mỗi trò chơi là một cây tìm kiếm có độ lớn khác nhau, khác nhau về độ sâu của cây và độ phân nhánh. Vì vậy, không có một máy tính nào có thể vét cạn được những cây tìm kiếm vừa và lớn. Cho nên, cây tìm kiếm thì cần phải có hàm lượng giá hỗ trợ, chi phí xây dựng hàm lượng giá thì rất cao, và chủ yếu là chi phí tìm ra các đặc trưng. Như vậy chương trình đánh cờ mạnh yếu khác nhau tùy thuộc vào phương pháp xây dựng hàm lượng giá và phương pháp tìm kiếm đặc trưng sao cho tối ưu. Hình 1 là một mô hình được đúc kết qua nhiều chương trình đánh cờ Deep Blue, Crazy Stone, AlphaGo.



Hình 1. Các thành phần chính trong bài toán đánh cờ

Như vậy chương trình đánh cờ là sự chọn lựa giữa các thành phần: mô hình chuyển đổi trạng thái, loại hàm lượng giá, và cách thức xây dựng loại hàm lượng giá. Phần còn lại của bài báo có cấu trúc như sau: Phần hai mô tả cách xác định độ phức tạp của từng loại cờ, phần ba mô tả tầm quan trọng của đặc trưng và cách thức rút trích, phần bốn mô tả bài toán và các loại hàm lượng giá, phần năm mô tả các cây tìm kiếm Monte Carlo là một cây trò chơi phù hợp cho các trò chơi có độ phân nhánh cao, cuối cùng là kết luận.

2. CÂY TRÒ CHƠI VÀ ĐỘ PHỨC TẠP

Một cây trò chơi bao gồm tất cả các nước đi có thể có của hai người chơi và mỗi nút của cây thể hiện một trạng thái bàn cờ sau khi nhận một nước đi từ người chơi. Từ một nút (trạng thái) hiện tại có thể có nhiều lựa chọn cho nước đi tiếp theo đó, số nước có thể chọn được gọi là hệ số phân nhánh. Độ sâu của cây trò chơi là số lần thay đổi lượt đi của hai người chơi. Hình 1 minh họa cây trò chơi của trò chơi đối kháng Tic-Tac-Toe, trò chơi này cực kì đơn giản vì chơi trên không gian 3x3 = 9 ô, 2 người chơi là X và O. Từ trạng thái ban đầu, bàn cờ chưa có quân cờ nào và do tính chất đối xứng của bàn cờ lượt đi đầu tiên dành cho người chơi O chỉ có thể đặt quân cờ đầu tiên vào 3

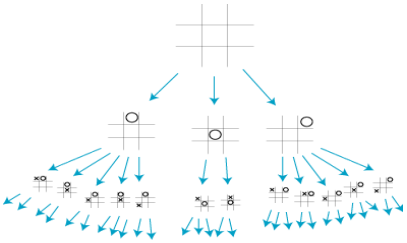
vị trí và hình thành nên 3 nhánh. Lượt đi tiếp theo dành cho người chơi X, tại nhánh thứ nhất có 5 khả năng đặt quân cờ, nhánh thứ 2 có 2 khả năng đặt quân cờ, nhánh thứ 3 có 5 khả năng đặt quân cờ. Tương tự cho các lượt đi tiếp theo, số lượt đi tối đa của trò chơi này cho cả 2 người chơi là 9. Đối với trò chơi Tic-Tac-Toe, mỗi ô có tối đa 3 trạng thái (O, X, trống), số ô của bàn cờ là 9, nên không gian trạng thái bàn cờ của trò chơi Tic-Tac-Toe là 3^9 = 19,683, lấy $\log_{10}$ = 4. Về số lượng cây có thể tính như sau: có 9 vị trí có thể đặt quân cho lượt đi đầu tiên nếu không quan tâm đến tính chất đối xứng của bàn cờ, 8 vị trí có thể đặt quân cho lượt đi thứ 2, 7 vị trí có thể đặt quân cho lượt đi thứ 3, tương tự như vậy, số lượng cây là 9! = 362,880, lấy $\log_{10}$ = 5. Sau đây là bảng so sánh độ phức tạp giữa các trò chơi đối kháng phổ biến, xem bảng 1.

Bảng 1. So sánh độ phức tạp giữa các trò chơi đối kháng

Trò chơi đối kháng	**Kích thước**	**Số trạng thái (theo $\log_{10}$)**	**Số cây (theo $\log_{10}$)**	**Số nước đi**	**Khả năng có thể đi trong 1 trạng thái**
Tic-tac-toe	9	4	5	9	4
Connect-4	42	14	21	36	4
Riversi (Othello)	64	28	58	58	10
Gomoku (Caro)	225	105	70	30	210
Chess	64	46	123	70	35
Chinese Chess	90	40	150	95	38
Connect-6	361	172	140	30	46000
Shogi	81	71	226	115	92
Go (19 x 19)	361	172	360	150	250

Trong bảng 1, chúng ta có thể thấy được độ phức tạp của các trò chơi. Đối với máy tính hiện đại thì những trò chơi có không gian tìm kiếm nhỏ như trò chơi Tic-Tac-Toe thì máy tính có thể vét cạn, và lúc đó chương trình đánh cờ chỉ từ hòa đến thắng vì biết được nước đi tốt nhất theo cách đi của đối phương. Các trò chơi có không gian tìm kiếm trung bình như Connect-4, Riversi, Chess, Chinese Chess, và Shogi thì máy tính không đủ khả năng để vét cạn. Lúc đó máy tính có thể tính trước một số bước nào đó rồi ước lượng, chương trình máy tính mạnh hay yếu nhờ vào khả năng ước lượng. Để chương trình có khả năng ước lượng thì phải xây dựng hàm lượng giá trạng thái. Đối với các trò chơi có hệ số phân nhánh phức tạp và không gian tìm kiếm quá lớn như Gomoku, Connect-6, Go thì việc xây dựng hàm lượng giá trạng thái tốt là gần như không thể.

Mô hình bài toán đánh cờ có độ phân nhánh cao

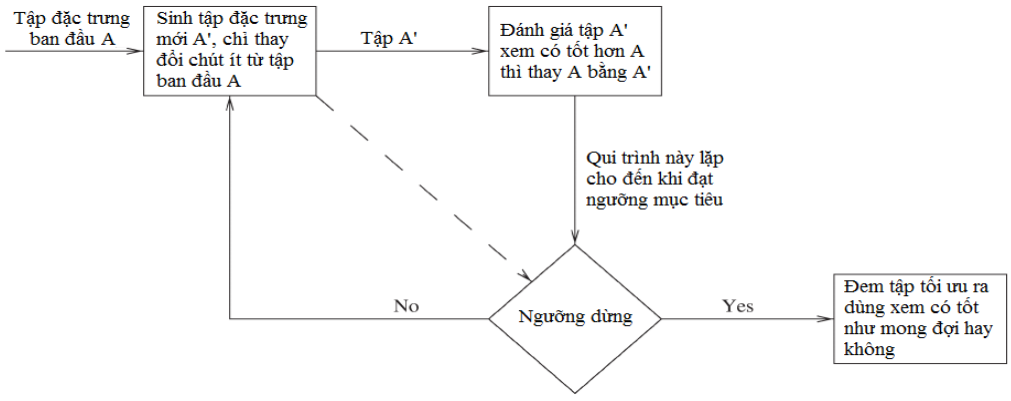


Hình 2. Cây trò chơi Tic-Tac-Toe

3. ĐẶC TRƯNG – TRI THỨC ƯỚC LƯỢNG NƯỚC ĐI

Trong phương pháp học máy, thay vì phải học hết tập dữ liệu huấn luyện lớn với chi phí cao và không hiệu quả do dữ liệu có những yếu tố dư thừa, nhiễu. Để kết quả huấn luyện cao thì thông thường học qua các đặc trưng thay vì học nguyên tập dữ liệu huấn luyện. Số lượng đặc trưng (features) càng nhiều thì độ chính xác càng cao, nhưng ngược lại, lượng đặc trưng quá nhiều sẽ khiến cho quá trình huấn luyện, quá trình phân loại mất nhiều thời gian hơn. Ngoài ra, nó còn khiến chương trình chiếm nhiều dung lượng bộ nhớ và đĩa cứng nhiều hơn. Vì vậy phải có phương pháp lựa chọn đặc trưng tối ưu, không nhất thiết phải chọn hết tất cả đặc trưng.

Bài toán đặt ra trong phương pháp máy học là phải lựa chọn từ tập các đặc trưng ra 1 tập con nhỏ hơn mà vẫn đảm bảo độ chính xác của quá trình phân loại. Việc lựa chọn đó, gọi là lựa chọn đặc trưng (feature selection, hay còn các tên khác như variable selection, feature reduction, attribute selection hay variable subset selection). Đối với từng phương pháp học máy, sẽ có những phương pháp tương ứng hiệu quả riêng với nó. Nói cách khác, không có phương pháp nào là tốt nhất. Nhưng phương pháp tìm tập đặc trưng phổ biến nhất được mô tả như trong hình 2.



Hình 3. Qui trình lựa chọn đặc trưng [5]

Theo qui trình lựa chọn đặc trưng như mô tả thì các phương pháp tối ưu ngẫu nhiên như Leo đồi, Luyện thép, Di truyền thường được dùng để thiết kế mô hình chọn

lựa đặc trưng. Công việc lớn nhất trong phần này là xây dựng một hàm mục tiêu phù hợp cho các phương pháp tối ưu ngẫu nhiên và phương pháp đánh giá kết quả.

Phương pháp đánh giá đặc trưng khá hiệu quả được đề cập nhiều trong 10 năm trở lại đây, từ năm 2007 do Remi Coulom giới thiệu [12, 14] là phương pháp Bradley Terry Minorization Maximization (BTMM). Phương pháp BTMM dùng trong việc tìm kiếm đặc trưng dựa trên tập dữ liệu các ván cờ, và độ đo Mean-Log Evidence được sử dụng rộng rãi để đánh giá xác suất được chọn của các nước đi trong các ván cờ, thông thường các ván cờ ở đây được dùng làm dữ liệu để học. Phương pháp này sẽ được giải thích dựa trên hai công thức (1) và (2):

$$prob(m) = \frac{strength(m)}{E} \quad (1)$$

$$MLE = \frac{1}{N}\sum_{j=1}^{N} \log(\, prob(m_j)) \quad (2)$$

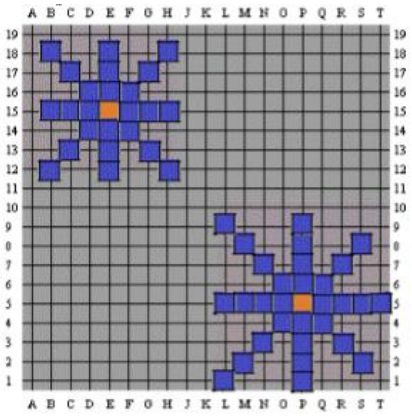
$$strength(m) = \prod_{i \in m} \gamma_i \quad (3)$$

$$\gamma_i = \frac{W_i}{\sum_{j=1}^{N} \frac{C_{ij}}{E_j}} \quad (4)$$

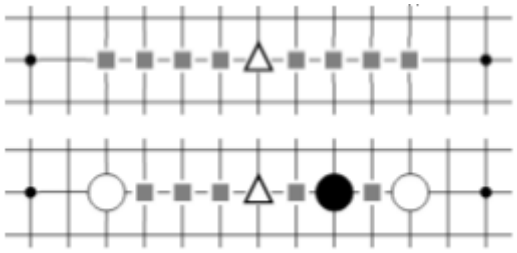
Trong công thức (1), *m* là nước đi nào đó, *strength*(m) là hàm tính độ mạnh của nước đi *m*, E là tổng độ mạnh của các nước đi hợp lệ trong một trạng thái của bàn cờ. Chúng ta hiểu trong một trạng thái của bàn cờ có nhiều nước đi hợp lệ có thể đi được, nhưng người chơi chỉ chọn một nước đi tốt nhất theo suy tính của người chơi. Nhưng đối với chương trình máy tính thì phải dựa trên hàm *strength*(m). Hàm này cũng chính là hàm lượng giá hành động được trình bày ở phần trên.

Trong công thức (2), *N* là số trạng thái có được trong tập dữ liệu các ván cờ dùng để học, m_j là nước đi được chọn trong trạng thái thứ *j*. Giả sử có 1,000,000 ván cờ dùng để làm dữ liệu học, mỗi ván cờ có trung bình xấp xỉ 70 nước đi. Như vậy giá trị *N* trong lúc này xấp xỉ 70,000,000 trạng thái. Quá trình học hay còn gọi là quá trình lựa chọn nghiệm tối ưu như trong hình 5 là quá trình lặp đi lặp lại cho đến khi không tìm ra nghiệm nào tốt hơn nữa thì dừng. Trong quá trình này chỉ số MLE thường được dùng để đánh giá độ tốt của nghiệm hiện hành.

Công thức (3) định nghĩa độ mạnh của một nước đi m, là tích các trọng số của các đặc trưng xuất hiện trong nước đi đó. Trọng số của mỗi đặc trưng được tính theo công thức (4), với W_i là số lần xuất hiện của đặc trưng i trong toàn bộ nước đi của tất cả ván cờ, C_{ij} là tích tất cả trọng số của các đặc trưng cùng xuất hiện với đặc trưng i trên trạng thái bàn cờ j, và E_j là tổng độ mạnh của tất cả các nước đi hợp lệ trên bàn cờ j.



Hình 4. Các mẫu chứa đặc trưng



Hình 5. Mẫu và đặc trưng

Trong hình 4 mô tả các mẫu xung quanh nước đi E15 và P5. Trong nước đi E15 có 4 mẫu có độ dài 6 mà người chơi cần chú ý đó là {B18, C17, D16, X, F14, G13, H12}, {B12, C13, D14, X, F16, G17, H18}, {B15, C15, D15, X, F15, G15, H15}, và {E18, E17, E16, X, E14, E13, E12}. Trong nước đi P5 có 2 mẫu có độ dài 8 là {L5, M5, N5, Q, X, Q5, R5, S5, T5}, {P9, P8, P7, P6, X, P4, P3, P2, P1} và 2 mẫu có độ dài 7 là {L9, M8, N7, O6, X, Q4, R3, S2}, {L1, M2, N3, O4, X, Q6, R7, S8}. Mẫu có độ dài n tại vị trí X thì chứa 2 x 3^n đặc trưng (xem hình 5). Ví dụ, mẫu có độ dài 6 thì chứa 1458 đặc trưng (729 đặc trưng cho quân X = màu Trắng, và 729 đặc trưng cho quân X = màu Đen). Công thức (4) tính trọng số cho một đặc trưng dựa trên các ván cờ có sẵn. Công thức số (3) tính độ mạnh cho một nước đi nào đó, ví dụ nước đi Ẹ như trong hình 5 có 4 đặc trưng nằm trong 4 mẫu như mô tả thì độ mạnh của nước đi E15 sẽ là tích các trọng số của 4 đặc trưng liên quan. Một vị trí nước đi có rất nhiều đặc trưng liên quan, để xác định đặc trưng nào là tốt nhất cho vị trí đó thì độ đo MLE trong công thức (3) được sử dụng.

4. BÀI TOÁN TRẠNG THÁI – HÀNH ĐỘNG VÀ HÀM LƯỢNG GIÁ

Phần này mô tả bài toán đánh cờ là một chuỗi trạng thái – hành động sao cho chuỗi đó mang về kết quả tối ưu cho người chơi. Bên cạnh đó còn mô tả hai loại hàm lượng giá quan trọng.

4.1. Bài toán trạng thái – hành động

Khi đối mặt với các cây trò chơi phức tạp, máy tính phải xét nhiều trạng thái và cần nhiều thời gian tính toán. Đây là bài toán chuyển đổi trạng thái liên tục từ trạng thái bắt đầu s_0 và kết thúc tại trạng thái s_T. Mô hình chuyển đổi trạng thái $T(s, a, s')$ được mô tả cụ thể như sau:

$$S = \{s_0, \ldots, s_T\}; s = s_i, s' = s_{i+1}$$

$$T(s, a, s') = \{\alpha\beta(s' | s, a), MCTS(s' | s, a)\}$$

$$A = \{a_1, \ldots, a_n\}, a \in A(s)$$

$$R = \{s* = \pi(s, a), a* = \pi(s, a)\}$$

S là tập các trạng thái của ván cờ, trạng thái này được hình thành do người chơi chọn nước đi, s_0 là trạng thái bắt đầu, và s_T là trạng thái kết thúc. Từ trạng thái kết thúc có thể biết được kết quả của ván cờ: thắng, thua, hòa. Trong các trạng thái có được thì s' là trạng thái tiếp theo của s khi đi một nước $a \in A(s)$, gọi là tập A chứa các nước có thể đi được tại trạng thái s của bàn cờ. Các thuật toán phổ biến để chuyển đổi trạng thái từ s sang s' hiện nay là $\alpha\beta$ và MCTS, những thuật toán này cần có hàm lượng giá để ước lượng nước nào là nước được chọn trong tập A. R là tập các hàm lượng giá có thể xây dựng, hàm lượng giá được chia làm hai loại: hàm lượng giá trạng thái và hàm lượng giá hành động. Hàm lượng giá trạng thái cho biết trạng thái s' nào là trạng thái tốt nhất tiếp theo sau s, hàm lượng giá hành động cho biết hành động $a \in A(s)$ nào là tốt nhất dù s' được tạo từ a chưa chắc là tốt nhất.

Tuy nhiên, nếu độ sâu trung bình của cây tìm kiếm lớn thì còn có thể kiểm soát bằng hàm lượng giá trạng thái, nhưng hệ số phân nhánh lớn thì nên dùng cách khác. Trong các trò chơi, nếu có nhiều nước đi "có khả năng, nhưng không hứa hẹn", thì nên tránh những nước đó càng nhiều càng tốt, để giảm tổng chi phí tìm kiếm trên cây. Phương pháp tìm kiếm truyền thống Minimax với tỉa Alpha-Beta thường được dùng để tỉa các nước đi không cần thiết theo cơ chế nhánh cận. Để phương pháp tỉa Alpha-Beta hiệu quả, thì thứ tự các khả năng đi được là khá quan trọng, thường người ta dùng "hàm lượng giá hành động" để sắp thứ tự. Đôi khi các hàm lượng giá hành động còn được dùng để giới hạn số nước tìm kiếm. Lấy ví dụ, chương trình cờ Vây (Nomitan) [7] của Ikeda và Viennot chỉ tìm kiếm 20 nước đi trên khoảng 361 khả năng có thể đi được.

4.2. Hàm lượng giá

Trong phần giới thiệu có đề cập hai loại hàm lượng giá: hàm lượng giá trạng thái và hàm lượng giá hành động. Trong một số cờ như cờ Vua, cờ Tướng, một nước đi $a \in A(s)$ đủ khả năng ảnh hưởng đến trạng thái của một bàn cờ cho nên đầu ra của hàm lượng giá trạng thái $s* = \pi(s, a)$ là ước lượng tất cả các trạng thái có thể, từ đó

chọn ra trạng thái tốt nhất sau khi người chơi đi nước đi *a* trên trạng thái *s*. Một số trò chơi như vờ Vây, Connect6, một nước đi $a \in A(s)$ không đủ khả năng ảnh hưởng đến trạng thái bàn cờ, nhưng những loại cờ này thường có hệ số phân nhánh quá cao, nghĩa là tồn tại nhiều nước đi "có khả năng, nhưng không hứa hẹn". Vấn đề ở đây là cần loại bỏ những nước đi không có nhiều hứa hẹn, lúc đó hàm lượng giá hành động $a* = \pi(s,a)$ phát huy vai trò. Hàm lượng giá hành động sẽ đo tất cả các nước đi để có căn cứ loại bỏ những nước không hứa hẹn. Chi phí xây dựng hàm lượng giá trạng thái thường mắc hơn rất nhiều so với chi phí xây dựng một hàm lượng giá hành động.

Sau đây là một hàm lượng giá trạng thái phổ biến cho cờ Vua. Hàm đánh giá trạng thái đánh giá tuyến tính tổng các đặc trưng có được của một đối thủ, vì lượt đi tiếp theo là của đối thủ.

$$Eval(s) = w_1 f_1(s) + w_2 f_2(s) + \ldots + w_n f_n(s) \quad (5)$$

Trong đó, w_i: trọng số gán cho quân thứ i, f_i: số quân thứ i của người chơi hiện hành so với đối phương. Ví dụ, trong cờ Vua quân Hậu có trọng số là 9, quân Xe có trọng số 5, quân Tượng và quân Mã có trọng số 3, quân Chốt có trọng số 1. Ngoài ra còn có các trọng số khác ngoài quân cờ như: số lượng các nước có khả năng đi của đối phương, Xe cặp, Mã cặp, Tốt cặp, sự an toàn của Vua.

Sau đây là hàm lượng giá hành động phổ biến cho cờ Vây, và Connect6. Giả sử tích tất cả các đặc trưng của nước đi *a* là $f(a)$, thì sức mạnh của nước đi *a* trong so với tất cả các nước đi hợp lệ trong *A* được tính theo công thức (1). Giả sử trạng thái bàn cờ có 3 nước có thể chọn, nước thứ nhất có liên quan đặc trưng 2 và 4, nước thứ hai có liên quan đến các đặc trưng 1, 2, 5, nước thứ ba có liên quan đến các đặc trưng 1, 3, 6, 7. Khi đó sức mạnh của nước đi một được tính như sau:

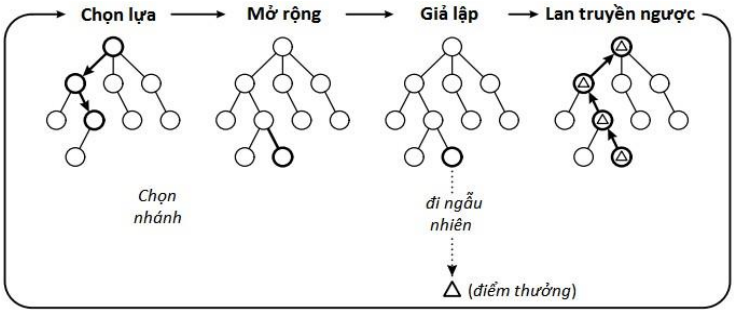
$$P(2\_4 \text{ against } 1\_2\_5 \text{ and } 1\_3\_6\_7) = \frac{\gamma_2\gamma_4}{\gamma_2\gamma_4 + \gamma_1\gamma_2\gamma_5 + \gamma_1\gamma_3\gamma_6\gamma_7}$$

Thông thường tri thức của người chơi được dùng trong việc thiết kế hàm lượng giá để tăng chất lượng chương trình giảm không gian tìm kiếm [6, 7]. Dùng tri thức bổ sung để xây dựng hàm lượng giá, tri thức bổ sung thường là dựa vào kinh nghiệm người chơi để lượng giá đặc trưng, hoặc dựa vào các ván cờ đã có và dùng các phương pháp chọn lựa đặc trưng để rút trích tự động từ các ván cờ đó. Hiện nay chọn lựa đặc trưng là một phần không thể thiếu để tìm ra các đặc trưng không phổ biến mà người chơi không thể thấy được, hoặc xác định trọng số chính xác hơn người chơi chỉ dựa vào kinh nghiệm, nhằm nâng cao chất lượng hàm lượng giá.

5. CÂY TÌM KIẾM MONTE CARLO

Các phương pháp Monte-Carlo hiện nay đang rất hiệu quả cho các chương trình cờ Vây. Năm 2006, Kocsis và Szepesvári [1, 3, 4] đề xuất thuật toán UCT (Upper Confidence Bounds for Tree) áp dụng cho cờ Vây. Hiện nay, thuật toán UCT và các biến thể của nó là các thuật toán chủ đạo cho các chương trình có sử dụng cây tìm kiếm Monte Carlo (MCTS). MCTS là phương pháp tìm kiếm dựa theo lấy mẫu dùng giả lập ngẫu nhiên để ước lượng tỷ lệ thắng thua của một trạng thái bàn cờ nhằm tìm ra nước đi tốt nhất và để cân bằng giữa việc khám phá và khai thác của tất cả các nước đi. Điểm mấu chốt của MCTS so với các phương pháp tìm kiếm truyền thống như Alpha-Beta và A* là nó không phụ thuộc vào tri thức đặc trưng của trò chơi, nói cách khác là không phụ thuộc vào hàm lượng giá trạng thái. Khi đó, MCTS có thể áp dụng vào nhiều trò chơi dạng không may rủi và thông tin trạng thái trò chơi rõ ràng sau mỗi lượt đi. Đối với các trò chơi mà khó xây dựng hàm lượng giá trạng thái tốt như cờ Vây thì việc áp dụng MCTS rất hiệu quả.

MCTS là một quá trình lặp đi lặp lại bốn bước trong một khoảng thời gian hữu hạn (xem hình 6). ***Chọn lựa***, từ một nút gốc (trạng thái bàn cờ hiện hành) cho đến nút lá, vì vậy sẽ có nhiều hướng đi được mang ra đánh giá. ***Mở rộng***, thêm một nút con vào nút lá của hướng được chọn trong bước chọn lựa, việc mở rộng không thực hiện trừ khi kết thúc ván cờ tại nút lá. ***Giả lập***, một ván cờ giả lập được chơi từ nút mở rộng, sau đó kết quả thắng thua của ván cờ giả lập sẽ được xác định. ***Lan truyền ngược***, kết quả thắng thua sẽ được cập nhật cho tất cả các nút của hướng được chọn theo cách lan truyền ngược. Cây trò chơi tăng trưởng sau mỗi lần lặp của MCTS, tăng trưởng rộng hơn và sâu hơn. Nước đi hứa hẹn là nút con nào có tỷ lệ thắng thua cao hơn được chọn trong giai đoạn chọn lựa, và rồi các cây con cũng tăng trưởng ngày càng rộng hơn và sâu hơn, và việc lấy mẫu ngày càng chính xác hơn. Sau khi kết thúc thời gian tìm kiếm, nút con nào được thăm nhiều nhất tại nút gốc sẽ được chọn để đi.



Hình 6. Cây tìm kiếm Monte Carlo

Thuật toán MCTS công thức sau cân bằng giữa việc lấy mẫu theo cách đào sâu một vùng nếu vùng này đang mang lại kết quả khả quan hay theo cách thăm dò vùng mới.

$$UCB_i = \frac{w_i}{n_i} + C\sqrt{\frac{lnN}{n_i}} \quad (6)$$

Trong đó tỷ số w_i/n_i là tỷ lệ thắng của nút con i, w_i là số lần thắng ván đấu giả lập có đi qua nút này, n_i là số lần nút này được đi qua, N là số lần nút cha được đi qua, và C là tham số có thể điều chỉnh. Tại nút gốc, nút con được chọn là nút có giá trị UCB cao nhất, rồi các nút con của nút hiện hành được so sánh với nhau bởi giá trị UCB và tiếp tục chọn nút có giá trị cao nhất. Vế thứ nhất trong công thức (6) có giá trị càng cao thì khả năng khai thác càng cao vì nút con có tỷ lệ thắng thua cao sẽ được chọn. Vế thứ hai của công thức (3) thể hiện sự tăng cường khám phá vì nút có số lần đi qua càng thấp thì càng có khả năng được chọn. Nếu hằng số C thấp, công thức sẽ nghiêng về khai thác, ngược lại nếu C cao, công thức sẽ nghiêng về khám phá.

6. KẾT LUẬN

Phương pháp tìm kiếm đặc trưng hiện nay là học có giám sát từ những dữ liệu ván cờ có sẵn, việc học này được lặp đi lặp lại nhiều lần cho đến khi tìm được những đặc trưng tối ưu đủ để xây dựng một hàm lượng giá tốt. Có hai phương pháp tìm kiếm tự động phổ biến là tìm kiếm tối ưu ngẫu nhiên dựa trên Bradley-Terry Minirization-Maximization, và Deep Learning. Để kiểm chứng tính hiệu quả của các đặc trưng và hàm lượng giá, phương pháp lấy trung bình logarit của các minh chứng (MLE-Mean Log Evidence) thường được sử dụng. Sau khi có hàm lượng giá tốt thì chương trình Monte Carlo mới phát huy hiệu quả thông qua việc giả lập (giai đoạn 3), và xác định vùng chọn mẫu (trong giai đoạn 1). Trong trò chơi có độ phân nhánh cao thì hàm lượng giá hành động là chọn lựa tốt nhất.

TÀI LIỆU THAM KHẢO

[1]. Browne, C., Powley, Whitehouse, Lucas, Cowling, Tavener, Perez, Samothrakis, Colton (2012). A survey of Monte Carlo tree search methods, *IEEE* transactions on computational intelligence and AI in games 4, pp. 1 – 43.

[2]. Buro, M. (2003). The evolution of strong othello programs, *In: The International Federation for Information Processing*, Volume 112. pp. 81 – 88.

[3]. Chaslot, G., Fiter, C., Hoock, J.-B., Rimmel, A., and Teytaud, O. (2009). Adding Expert Knowledge and Exploration in Monte-Carlo Tree Search, *Proceedings of the Twelfth International Advances in Computer Games Conference*, pp. 1-13, Pamplona, Spain.

[4]. Chaslot, G., Winands, M., Bouzy, B., Uiterwijk, J. W. H. M., and Herik, H. J. van den (2007). Progressive Strategies for Monte-Carlo Tree Search, *Proceedings of the 10th Joint Conference on Information Sciences* (ed.P. Wang), pp. 655–661, Salt Lake City, USA.

[5]. M. Dash, H. Liu (1997). Feature Selection for Classification, *Elservier*, Intelligent Data Analysis 1 (1997) 131–156.

[6]. Coulom, R. (2007). Computing elo ratings of move patterns in the game of go, *ICGA Journal 30*, pp. 198 – 208.

[7]. Ikeda, K., Viennot, S. (2013). Efficiency of static knowledge bias in monte-carlo tree search, *In: Computers and Games 2013*

MODEL OF BOARD GAME WITH A LARGE NUMBER OF BRANCHES

Dang Cong Quoc[1], Nguyen Dang Binh[1], Nguyen Quoc Huy[2]

[1] Faculty of Information Technology, University of Sciences, Hue University

[2] Sai Gon University

Email: dangcongquoc1968@gmail.com, nguyendangbinh@gmail.com, nqhuy@sgu.edu.vn

ABSTRACT

Board game is a sequence that repeats the choice of moves between two players; the state of game board will be changed if we played a new move. In other words, this is a problem in which the optimized solution is found. Therefore, the main components of the problem such as the search tree, evaluation functions, features, and the methods of feature choice should be optimized. There are many kinds of board games and each kind is a useful test for AI research. This paper introduces the basic knowledge of board games, and the approaches to make a strong program of board games.

Keywords: Board game, Evaluation, Feature choice, function, Reinforcement learning.

Đặng Công Quốc sinh ngày 21 tháng 06 năm 1968 tại Thành phố Huế. Năm 1991, ông tốt nghiệp cử nhân Toán tại Trường Đại học Sư phạm, Đại học Huế. Năm 1995, ông tốt nghiệp cử nhân Tin học tại Trường Đại học Cần Thơ. Năm 2001, ông tốt nghiệp Thạc sĩ khoa học, chuyên ngành Toán tại Trường Đại học Sư phạm, Đại học Huế. Năm 2015, ông tốt nghiệp thạc sĩ CNTT tại Trường Đại học Công nghệ TP. HCM. Từ năm 1991, ông tham gia giảng dạy và hiện đang là phó Hiệu trưởng Trường Cao đẳng Công thương TP. HCM. Hiện ông là NCS ngành Khoa học máy tính của Trường Đại học Khoa học, Đại học Huế.

Lĩnh vực nghiên cứu: Công nghệ thông tin, Toán học.

Nguyễn Đăng Bình sinh ngày 08 tháng 11 năm 1974 tại Thừa Thiên Huế. Năm 1996, ông tốt nghiệp ngành Toán – Tin tại Trường Đại học Sư phạm, Đại học Huế. Năm 2001, ông tốt nghiệp Thạc sĩ Công nghệ thông tin tại Trường Đại học Bách khoa Hà Nội. Năm 2008, ông tốt nghiệp Tiến sĩ Công nghệ thông tin tại Trường Khoa học Máy tính và Kỹ sư hệ thống, Viện Công nghệ Kyushu, Nhật Bản. Từ năm 1996 ông tham gia giảng dạy và hiện đang là giảng viên khoa CNTT Trường Đại học Khoa học, Đại học Huế.

Lĩnh vực nghiên cứu: Machine Vision; Machine Learning; Thị giác máy tính; Nhận dạng; Xử lý ảnh; Các hệ thống giám sát thông minh; Tương tác giữa người và máy tính.

Nguyễn Quốc Huy sinh ngày 12 tháng 12 năm 1978 tại Đà Nẵng. Năm 2002, ông tốt nghiệp cử nhân khoa học máy tính tại Trường Đại học Khoa học Tự nhiên Tp. HCM. Năm 2008, ông tốt nghiệp Thạc sĩ Khoa học máy tính tại Trường Đại học Khoa học Tự nhiên TP. HCM. Năm 2014, ông tốt nghiệp Tiến sĩ ngành Khoa học máy tính tại JAIST, Nhật Bản. Từ năm 2002, ông tham gia giảng dạy tại Trường Đại học Sài Gòn.

Lĩnh vực nghiên cứu: Machine Learning, Computer Games, Data Mining.