THUẬT TOÁN HỌC PHÂN TÁN CHO HỆ ĐA TÁC TỬ

Từ Minh Phương

Học viện Công nghệ Bưu chính Viễn thông

Một trong những vấn đề đặt ra đối với hệ thống bao gồm nhiều tác tử là tác tử phải có khả năng phối hợp hành động với nhau sao cho hành động chung dẫn tới kết quả mong muốn. Vấn đề này có thể giải quyết bằng cách cho tác tử tự học cách phối hợp với tác tử khác nhờ các kỹ thuật học tự động. Học tăng cường (reinforcement learning) là kỹ thuật học tự động được sử dụng rộng rãi nhất cho tác tử độc lập và gần đây được nghiên cứu mở rộng cho hệ đa tác tử. Báo cáo trình bầy một thuật toán học tăng cường cho hệ thống bao gồm nhiều tác tử cộng tác với nhau trong đó quá trình học được tiến hành song song và phân tán trên tất cả tác tử. Thuật toán học được xây dựng trên cơ sở cải tiến thuật học Q (Q-learning) với bảng Q được phân tán và cập nhật độc lập trên các tác tử. Thuật toán được cài đặt và thử nghiệm cho bài toán di chuyển vật nặng với hai tác tử. Kết quả thử nghiệm cho thấy tính hiệu quả và khả năng ứng dụng của thuật toán.

1. ĐẶT VẤN ĐỀ

Hệ đa tác tử (*multiagent system*) là hệ thống trong đó nhiều tác tử tự chủ tương tác với nhau để thực hiện một số nhiệm vụ nào đó. Yêu cầu quan trọng đối với hệ đa tác tử là tác tử phải có khả năng phối hợp hành động với nhau sao cho toàn hệ thống hoạt động hiệu quả. Cơ chế phối hợp có thể cài đặt sẵn khi xây dựng tác tử dưới dạng các quy ước, các kế hoạch lập sẵn, các kỹ thuật đồng bộ.v.v. [14]. Một phương pháp đảm bảo phối hợp khác là không cài đặt sẵn cơ chế đồng bộ mà để tác tử tự học cách phối hợp hành động thông qua kinh nghiệm thu được trong quá trình tương tác với nhau. Phương pháp này có một số ưu điểm như trực quan, cho kết quả ổn định. Vấn đề đặt ra là cần xây dựng các thuật toán học tự động phù hợp với tính chất phân tán và không đồng bộ vốn có của hệ đa tác tử.

Học tăng cường (*reinforcement learning*) truyền thống là kỹ thuật học tự động dùng cho một tác tử riêng lẻ. Tác tử phải học cách ra quyết định hành động thông qua chuỗi các tương tác với môi trường và quan sát kết quả. Thông thường bài toán học tăng cường được cho bởi tập các trạng thái môi trường S, tập các hành động của tác tử A. Ở mỗi trạng thái s, tác tử thực hiện hành động a. Do kết quả hành động a, môi trường chuyển sang trạng thái mới, đồng thời tác tử nhận được một kết quả gọi là *phần thưởng* hoặc tín hiệu *tăng cường*. Nhiệm vụ của tác tử là học cách hành động sao cho tổng giá trị phần thưởng theo thời gian là lớn nhất.

Khi tác tử cùng hoạt động trong hệ thống với những tác tử khác, trạng thái môi trường và giá trị phần thưởng không chỉ phụ thuộc vào hành động của tác tử mà còn phụ thuộc vào hành vi của các tác tử còn lại. Nhiệm vụ của tác tử khi đó là học cách hành động sao cho phù hợp với với hành vi của các tác tử trong hệ thống. Kỹ thuật học tăng cường truyền thống (cho một tác tử) không cho kết quả mong muốn trong trường hợp này: hoặc tác tử không tìm được hành vi tối ưu hoặc hành vi đó chỉ tối ưu khi tác tử khác hành động theo một cách nhất định. Từ đây xuất hiện vấn đề mở rộng kỹ thuật học tăng cường cho hệ đa tác tử.

Hệ đa tác tử có thể chia thành hai loại chính: hệ bao gồm các tác tử cộng tác và hệ bao gồm các tác tử không cộng tác. Trong trường hợp thứ nhất, tác tử có cùng một mục đích, cũng là mục đích chung của cả hệ thống, và hành động để đạt mục đích đó. Trong trường hợp sau, tác tử trong một hệ thống có lợi ích và mục tiêu khác nhau, thậm chí mâu

thuần, mỗi tác tử đều cố gắng hành động vì lợi ích của riêng mình. Các nghiên cứu về học tăng cường cho hệ tác tử cũng phân biệt riêng kỹ thuật học cho hai trường hợp này.

Littman [8], Hu và Wellman [5] mô tả kỹ thuật học cho hệ bao gồm các tác tử cạnh tranh dựa trên lý thuyết học tăng cường và lý thuyết trò chơi. Tác tử được mô hình hoá dưới dạng các bên tham gia một trò chơi, mỗi bên cố gắng cực đại hóa hàm tiện ích của mình. Tác giả các bài báo này đề xuất hai thuật toán học tăng cường cho phép tìm ra chính sách hành động tối ưu dưới dạng cân bằng theo Nash. Cả hai thuật toán đều đòi hỏi mỗi tác tử có thông tin đầy đủ về hành động của tác tử khác và trạng thái môi trường. Bowling và Veloso [1] mô tả bài toán tương tự và đề xuất thuật toán WoLF trong đó hệ số học thay đổi theo thời gian. Kết quả thực nghiệm cho thấy WoLF luôn tìm được kết quả tối ưu và hội tụ nhanh hơn hai thuật toán trước.

Trường hợp hệ tác tử cộng tác được nghiên cứu trong [3,15,7, 9]. Trong [3], mỗi tác tử hoặc sử dụng thuật học tăng cường Q một cách độc lập (không quan tâm đến hành tác tử khác) hoặc tiến hành học cho hành động chung. Phương pháp này cho kết quả không tối ưu trong nhiều trường hợp. Một giải pháp khác được đề xuất trong [7,15] trong đó hàm phần thưởng được phân tán trên các tác tử. Tác tử có thể thông báo cho nhau giá trị phần thưởng của mình, thông qua đó học cách phối hợp với tác tử khác. Phương pháp này cho kết quả học tương đối tốt mặc dù đòi hỏi tác tử trao đổi thông tin với nhau.

Ngoài các nghiên cứu về lý thuyết, kỹ thuật học cho hệ tác tử đã được nghiên cứu áp dụng cho nhiều bài toán cụ thể. Các ứng dụng được biết đến nhiều nhất là học điều khiển nhóm cầu thang máy [4] với mỗi thang máy được đại diện bởi một tác tử, định tuyến trong mạng [2], điều khiển lưới điện [9], phân phối kênh truyền cho mạng điện thoại di động [11]. Kết quả cho thấy học tăng cường có thể mở rộng với các mức độ khác nhau cho hệ thống đa tác tử và thích hợp với nhiều ứng dụng thực tế.

Trong báo cáo này, đối tượng nghiên cứu được giới hạn là hệ thống với các tác tử cộng tác. Báo cáo trình bầy thuật toán cho phép phân tán quá trình học trên các tác tử. Thuật toán dựa trên cơ sở thuật toán học Q, một trong những thuật toán học tăng cường được sử dụng nhiều nhất. Nội dung cơ bản của thuật toán là mỗi tác tử lưu trữ và cập nhật một bảng Q riêng cho mình. Tác tử chỉ cần có thông tin về giá trị phần thưởng chung và trạng thái chung của hệ thống mà không cần thông tin về hành động của tác tử khác. Thuật toán cho phép tiết kiệm chi phí liên lạc giữa tác tử trong khi vẫn cho phép học được hành vi tối ưu.

Thuật toán được cài đặt và thử nghiệm cho bài toán trong đó hai tác tử cộng tác với nhau để di chuyển một vật tới một vị trí cho trước. Kết quả thử nghiệm cho thấy tác tử học được cách phối hợp hành động cần thiết đồng thời có thời gian học ngắn hơn so với thuật toán *Bucket Brigade* được sử dụng trong [10].

Các phần còn lại được bố cục như sau. Phần 2 trình bầy mô hình quá trình quyết định Markov mở rộng được sử dụng để mô hình hoá bài toán học tự động cho hệ đa tác tử. Phần 3 mô tả thuật toán học Q phân tán cho các tác tử cộng tác. Phần 4 giới thiệu bài toán di chuyển vật nặng bởi nhiều tác tử và trình bầy kết quả thử nghiệm thuật toán cho bài toán đó. Phần 5 là kết luận của báo cáo.

2. PHÁT BIỂU BÀI TOÁN HỌC TỰ ĐỘNG CHO HỆ ĐA TÁC TỬ

Bài toán học tự động cho hệ đa tác tử có thể phát biểu và phân tích trên cơ sở kết hợp mô hình *quá trình quyết định Markov* (*Markov Decission Process – MDP*) và lý thuyết trò chơi.

Quá trình quyết định Markov (MDP) là mô hình hệ thống với một tác tử và nhiều trạng thái. MDP được định nghĩa dưới dạng bộ bốn *(S,A,T,R)* trong đó S là tập các trạng thái, A là tập các hành động của tác tử, T là hàm chuyển tiếp $T: S \times A \times S \to [0,1]$, và R là hàm thưởng $R: S \times A \to \mathbf{R}$. Hàm chuyển tiếp T xác định xác suất chuyển tiếp sang trạng thái tiếp theo tuỳ thuộc vào trạng thái hiện thời và hành động của tác tử. Trong trường hợp T chỉ có thể bằng 0 hoặc 1 hệ thống được gọi là *xác định* (*deterministic*). Hàm thưởng R xác định giá trị phần thưởng hay lợi ích mà tác tử nhận được khi thực hiện một hành động đối với trạng thái cho trước. Mô hình được gọi là Markov nếu hàm chuyển tiếp chỉ phụ thuộc vào trạng thái và hành động hiện thời của tác tử mà không phụ thuộc vào trạng thái và hành động trước đó. Yêu cầu của học tăng cường đối với MDP là tìm ra chính sách hành động $\pi: S \to A$ cho phép lựa chọn hành động tuỳ thuộc vào trạng thái sao cho tổng phần thưởng nhận được trong tương lai là lớn nhất. Tổng phần thưởng được tính bởi

$\sum_{t=0}^{\infty} \gamma^t r_t$ - rong đó γ là *yếu tố chiết khấu* (discount factor). Ý nghĩa của yếu tố chiết khấu là phần thưởng nhận được trong tương lai gần có giá trị hơn trong tương lai xa.

Trò chơi ma trận (matrix game). Khác với MDP, trò chơi ma trận là mô hình hệ thống với một trạng thái duy nhất và nhiều tác tử. *Trò chơi ma trận* được cho bởi $(n, A_{1..n}, R_{1..n})$, trong đó n là số lượng tác tử (người chơi), A_i là tập hành động mà tác tử i có thể lựa chọn, R_i là hàm thưởng của tác tử i $R_i: A \to \mathbf{R}$, ở đây A là tập các hành động chung của tất cả tác tử tham gia trò chơi $A = A_1 \times A_2 \times ... \times A_n$. Mỗi tác tử lựa chọn một hành động trong tập hành động cho phép của mình và nhận được phần thưởng tuỳ thuộc vào hành động chung của tất cả tác tử. Hàm R_i thường được cho dưới dạng ma trận và do vậy mô hình có tên là trò chơi ma trận. Ở đây cần phân biệt hai trường hợp: hệ thống bao gồm các tác tử cộng tác và hệ thống bao gồm tác tử cạnh tranh. Đối với tác tử cộng tác, hàm phần thưởng R_i là như nhau cho tất cả tác tử và tác tử hành động để cực đại hoá hàm phần thưởng chung này. Trong trường hợp cạnh tranh, hàm phần thưởng của tác tử là khác nhau.

Mô hình MDP mở rộng. Trong trường hợp tổng quát, hệ thống có thể có nhiều trạng thái khác nhau và bao gồm nhiều tác tử. Để biểu diễn bài toán học tăng cường cho hệ thống như vậy, hai mô hình ở trên được mở rộng bằng cách kết hợp với nhau [1]. *Mô hình MDP mở rộng* được cho bởi bộ $(n, S, A_{1..n}, T, R_{1..n})$, trong đó các thành phần được định nghĩa tương ứng như trong hai mô hình trên. Mô hình này khác với MDP ở chỗ hành động được xác định bởi nhiều tác tử khác nhau và hàm phần thưởng phụ thuộc vào hành động chung của toàn bộ tác tử. Yêu cầu đặt ra đối với quá trình học là xác định chính sách hành động cho từng tác tử $\pi_i: S \to A$ sao cho tổng phần thưởng đối với tác tử đó trong tương lai là lớn nhất.

Đối tượng nghiên cứu trong bài báo này là hệ thống bao gồm các tác tử cộng tác. Hàm phần thưởng chung của hệ thống được xác định như tổng các hàm phần thưởng thành phần (của từng tác tử). Mục tiêu của học tăng cường là xác định chính sách hành động cho phép cực đại hoá hàm phần thưởng chung này. Gọi Π là chính sách chung của hệ thống và được tạo thành từ các chính sách riêng của tác tử $\Pi(s) = (\pi_1(s), ..., \pi_n(s))$, cần tìm Π sao cho tổng phần thưởng chung của hệ thống theo thời gian t

$\sum_{t=0}^{\infty} \gamma^t R(s_t, \Pi(s_t))$ là lớn nhất.

3. THUẬT TOÁN

3.1. Thuật toán học Q tập trung

Thuật học Q (Q-learning) [13] là một trong những thuật toán học tăng cường được sử dụng nhiều nhất cho MDP. Nội dung cơ bản của thuật học Q là sử dụng khái niệm giá trị tăng cường $Q(s,a)$. $Q(s,a)$ được định nghĩa như giá trị tăng cường (có tính tới chiết khấu) nhận được nếu thực hiện hành động a trong trạng thái s. Giá trị $Q(s,a)$ được khởi tạo ngẫu nhiên (chặn hạn bằng 0) cho tất cả các đôi (s,a) và được cập nhật theo quy tắc đệ quy sau

$$Q(s,a) = (1-\alpha)Q(s,a) + \alpha(R(s,a) + \gamma \max_{a'} Q(s',a')) \quad (1)$$

trong đó s là trạng thái trước khi chuyển tiếp, a là hành động được chọn, $R(s,a)$ là phần thưởng nhận được tức thì nhờ hành động a, s' là trạng thái tiếp theo, a' là hành động tiếp theo, α là hệ số học , $0 \le \alpha \le 1$ và xác định mức độ thay đổi giá trị Q sau mỗi bước học. Nếu mỗi hành động được thực hiện một số lần vô hạn cho mỗi trạng thái thì giá trị Q sẽ hội tụ tại giá trị tối ưu Q^*. Chính sách tối ưu π^* khi đó được xác định từ Q^* như sau:

$$\pi^*(s) = \arg\max_a Q^*(s,a) \quad (2)$$

3.2.Thuật toán học Q tập trung cho hệ đa tác tử

Thuật toán Q được xây dựng cho trường hợp một tác tử. Đối với hệ thống bao gồm nhiều tác tử, cách đơn giản nhất là tiến hành học tập trung. Quá trình học được thực hiện tập trung trên một tác tử duy nhất. Mỗi hành động của tác tử học là một hành động trong không gian hành động chung A. Sau khi đã tìm được chính sách hành động chung, chính sách này sẽ được chia thành các chính sách thành phần và gửi cho tác tử tương ứng.

Thuật toán học Q tập trung cho hệ đa tác tử được mô tả theo quy tắc sau

Khởi tạo $Q(s,a) = 0$

Cập nhật $$Q(s,a) = (1-\alpha)Q(s,a) + \alpha(R(s,a) + \gamma \max_{a'} Q(s',a')) \quad (3)$$

trong đó $a, a' \in A$, $A = (A_1 \times A_2 \times ... \times A_n)$.

Thuật toán học Q tập trung rất giống thuật học Q truyền thống. Cụ thể, nếu mỗi cặp (s,a) được xuất hiện với số lần vô hạn thì thì giá trị Q sẽ hội tụ.

Nhược điểm lớn nhất của giải pháp học tập trung là thông tin về trạng thái, hành động cũng như hàm phần thưởng phải được thu thập và xử lý tập trung. Việc xử lý tập trung như thường làm giảm tính ổn định và độ tin cậy của hệ thống, gây tắc nghẽn ở nút (tác tử) trung tâm, tăng chi phí truyền thông giữa tác tử đồng thời không tận dụng được ưu thế về tốc độ của xử lý song song. Ngoài ra, việc thu thập thông tin tập trung đòi hỏi tác tử phải hiểu được cách biểu diễn hành động của tác tử khác. Yêu cầu này có thể không thoả mãn đối với hệ thống bao gồm các tác tử hỗn tạp. Do vậy, yêu cầu đặt ra là xây dựng thuật toán học phân tán sao cho từng tác tử có thể học chính sách hành động của mình với ít yêu cầu trao đổi thông tin nhất.

3.3. Thuật toán học Q phân tán

Trong trường hợp thuật học Q tập trung, thuật toán học lưu trữ một bảng các giá trị Q cho từng cặp (s,a), $(a = (a_1,...,a_n))$. Tuy nhiên, khi tác tử tiến hành học độc lập với tác tử khác, từng tác tử không thể biết hết các tổ hợp hành động chung a mà chỉ có khả năng

phân biệt các hành động của riêng mình. Mục tiêu của thuật toán học phân tán là cho phép từng tác tử học dựa trên hành động của riêng mình mà không cần tới thông tin về hành động của tác tử khác. Ở đây ta vẫn giả thiết là mỗi tác tử đều biết được giá trị hàm phần thưởng chung R và trạng thái chung s của toàn hệ thống.

Do mỗi tác tử chỉ phân biệt được hành động của mình, thay vì sử dụng bảng Q chung, mỗi tác tử i sẽ lưu trữ và cập nhật bảng Q_i riêng. Giá trị trong bảng Q_i riêng phụ thuộc vào trạng thái s của hệ thống và hành động riêng a_i của tác tử. Vấn đề đặt ra là xác định giá trị Q_i như thế nào và giá trị đó quan hệ với giá trị trong bảng Q trung tâm ra sao. Việc lựa chọn giá trị Q_i quyết định thuật toán và chất lượng học phân tán.

Theo phương pháp đơn giản nhất [3] , mỗi tác tử i tự cập nhật bảng Q_i riêng mà không quan tâm tới tác tử khác theo công thức sau

$$Q_i(s,a_i^j) = \sum_{a=(a_1,\ldots,a_n),a_i=a_i^j} (\Pr(a \mid a_i)(R(s,a) + \gamma \max_{a_i' \in A_i} Q_i(s',a_i'))) \tag{4}$$

trong đó $\Pr(a \mid a_i)$ là xác suất của hành động chung a khi tác tử i chọn hành động a_i trong trạng thái s. Xác suất này có thể tính chẳng hạn bằng cách ghi lại tần số xuất hiện a cùng với a_i trước đó. Tuy nhiên, điều này đòi hỏi tác tử phải có thông tin về hành động chung (không đảm bảo yêu cầu phân tán). Ngoài ra thuật toán này không cho phép tìm ra chính sách hành động tối ưu trong nhiều trường hợp [3].

Dưới đây là phương pháp cho phép tính giá trị Q_i mà không cần thông tin về hành động chung. Nội dung cơ bản của phương pháp này là với mỗi trạng thái s và hành động a_i^j, tác tử i chỉ cập nhật giá trị Q_i tương ứng nếu giá trị Q_i mới lớn hơn giá trị cũ. Nói cách khác, mỗi tác tử chỉ ghi lại giá trị tăng cường cho hành động chung tốt nhất ứng với mỗi trạng thái và hành động riêng của mình cho tới thời điểm hiện tại. Giá trị Q_i riêng của tác tử khi đó được cập nhật theo quy tắc sau

$$Q_i(s,a_i^j) = \max\{Q_i(s,a_i^j), R(s,a_i^j) + \gamma \max_{a_i' \in A_i} Q_i(s',a_i')\},\ (a_i^j \in A_i) \tag{5}$$

Ở thời điểm bắt đầu, $Q_i(s,a_i^j)$ được khởi tạo bằng 0.

Nếu mỗi cặp trạng thái hành động riêng (s,a_i) xảy ra đủ nhiều thì có thể dễ dàng chứng minh bằng quy nạp là việc cập nhật theo quy tắc (5) sẽ cho kết quả sau

$$Q_i(s,a_i^j) = \max_{a=(a_1,\ldots,a_n),a_i=a_i^j} Q(s,a) \tag{6}$$

Tức là bảng Q_i riêng chứa giá trị lớn nhất của bảng Q chung ứng với mỗi hành động riêng a_i^j của tác tử. Từ đây có thể thấy bảng Q riêng được cập nhật theo (5) sẽ chứa giá trị Q^* tối ưu của bảng Q chung được tính theo (4) nếu mỗi đôi *trạng thái -- hành động chung* xảy ra đủ nhiều.

Ví dụ: Để minh hoạ cho thuật toán trên, xét ví dụ đơn giản sau [3,5]. Ví dụ cho dưới dạng trò chơi bao gồm hai tác tử cộng tác, một trạng thái duy nhất s_0 , và ba hành động riêng $A_i = \{a_i^1, a_i^2, a_i^3\}, i = 1,2$ (có thể tổng quát hoá dễ dàng cho trường hợp nhiều trạng thái và nhiều tác tử). Hàm phần thưởng được cho dưới dạng ma trận như sau

	a_1^1	a_1^2	a_1^3
a_2^1	10	0	k
a_2^2	0	2	0
a_2^3	k	0	10

trong đó k là tham số sao cho k<10. Sử dụng quy tắc cho bởi (5), tác tử sẽ xây dựng được bảng Q riêng như sau

	a_i^1	a_i^2	a_i^3
$Q_1(s_0,a_i)$	10	2	10
$Q_2(s_0,a_i)$	10	2	10

Nếu sử dụng xác định chính sách hành động tối ưu từ bảng Q riêng dựa trên công thức (2) ta có $\pi_i^*(s) = \arg\max_{a_i^j \in A_i} Q_i(s,a_i^j)$. Theo công thức này có thể tìm được hai chính sách riêng cho mỗi tác tử $\pi_1^*(s_0) = \{a_1^1, a_1^3\}$ và $\pi_2^*(s_0) = \{a_2^1, a_2^3\}$. Từ đây có thể suy ra bốn chính sách chung (a_1^1,a_2^1), (a_1^3,a_2^3), (a_1^1,a_2^3) và (a_1^3,a_2^1). Dễ dàng nhận thấy chỉ có hai chính sách đầu là tối ưu và cho phần thưởng là 10 trong khi hai chính sách sau không tối ưu với giá trị phần thưởng nhỏ hơn 10. Nguyên nhân của việc xuất hiện các chính sách chung không tối ưu là do tuy từng tác tử tìm được tìm được hành động thành phần tối ưu nhưng không phối hợp được với hành động tương ứng của tác tử khác.

Để đảm bảo thuật toán học hội tụ ở chính sách tối ưu cần bổ sung khả năng phối hợp hành động của tác tử. Trước hết, do tất cả các đôi trạng thái - hành động đều lặp lại với một số lần đủ lớn, sẽ xuất hiện hành động ứng với chính sách tối ưu. Nếu thuật toán học ghi lại hành động tốt nhất cho đến thời điểm hiện tại thì khi kết thúc quá trình học, hành động tốt nhất được ghi lại cho mỗi trạng thái cũng chính là chính sách tối ưu. Như vậy, tác tử tiến hành cập nhật chính sách π_i đồng thời với cập nhật bảng Q_i chứ không xác định chính sách một lần duy nhất sau khi đã học xong theo công thức (2). Thuật toán học phân tán (5) khi đó được mở rộng thành:

Với mỗi tác tử i
Tại thời điểm 0:
Khởi tạo $Q_i^0(s,a_i^j) = 0$
Khởi tạo ngẫu nhiên $\pi_i^0(s) \in A_i$
Tại thời điểm t:
Thực hiện hành động a_i^j
Cảm nhận trạng thái môi trường s'
Xác định giá trị phần thưởng $R(s,a_i^j)$
Cập nhật $Q_i^{t+1}(s,a_i^j) = \max\{Q_i^t(s,a_i^j), R(s,a_i^j) + \gamma \max_{a_i' \in A_i} Q_i^t(s',a_i')\}$
Nếu $\max_{a_i^j \in A_i} Q_i^{t+1}(s,a_i^j) > \max_{a_i^j \in A_i} Q_i^t(s,a_i^j)$
thì cập nhật $\pi_i(s) = a_i^j$

Hình 1. **Thuật toán học Q phân tán**

(Hệ số t ở trên được thêm vào để tiện phân biệt các bước thời gian của quá trình học).

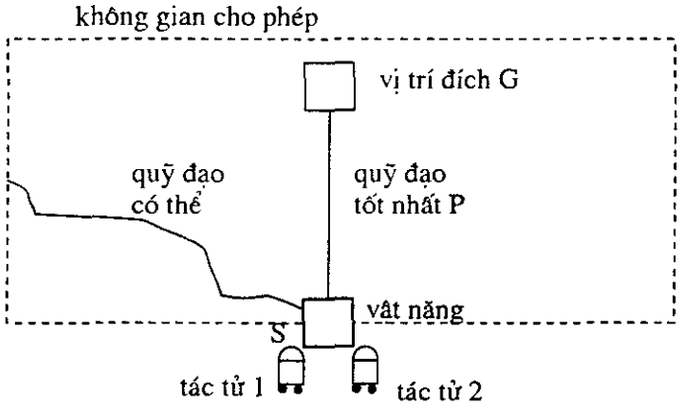
Trở lại ví dụ ở trên. Do hành động chung (a_1^1,a_2^1) và (a_1^3,a_2^3) cho giá trị Q_i (i=1,2) lớn nhất, trong quá trình học, tuỳ thuộc vào (a_1^1,a_2^1) hay (a_1^3,a_2^3) xuất hiện trước, thuật toán sẽ lưu lại $\pi_1(s) = a_1^1$ và $\pi_2(s) = a_2^1$ hay $\pi_1(s) = a_1^3$ và $\pi_2(s) = a_2^3$. Đây chính là chính sách hành động tối ưu cần tìm.

4. THỬ NGHIỆM – BÀI TOÁN DI CHUYỂN VẬT NẶNG

Để kiểm nghiệm thuật toán trình bầy trong phần trước, thuật toán được cài đặt và thử nghiệm cho bài toán di chuyển vật nặng [10]. Trong bài toán này, hai tác tử (rôbốt) được phân công di chuyển một vật nặng từ điểm xuất phát S đến điểm đích G theo một quỹ đạo P trên một mặt phẳng (được biểu diễn trên hệ toạ độ Đề các) như trên Hình 2. Hai tác tử không biết trước khả năng hành động của tác tử kia và cần học cách thích ứng với nhau. Mỗi tác tử đều có khả năng đánh giá khoảng cách từ vị trí hiện tại của vật cần di chuyển đến quỹ đạo cần đi (một phát biểu khác khó hơn của bài toán là tác tử chỉ có khả năng phân biệt vị trí hiện tại đã là đích chưa, tuy nhiên phát biểu này đòi hỏi thời gian học lâu hơn và không được đề cập đến ở đây). Tại mỗi thời điểm, mỗi tác tử i (i=1,2) có thể tác động vào vật nặng một lực $\vec{F}_i$, $0 \le |\vec{F}_i| \le F_{max}$ dưới một góc θ_i. Tổng hợp tác động của hai tác tử được xác định bởi véctơ lực $\vec{F} = \vec{F}_1 + \vec{F}_2$. Lực này làm vật nặng dịch chuyển theo trục x và y khoảng cách tương ứng là $|\vec{F}|\cos(\theta)$ và $|\vec{F}|\sin(\theta)$ đơn vị. Giả sử vị trí hiện tại là (x,y), $P_x(y)$ là tọa độ x của quỹ đạo P tại cùng tung độ y, $\Delta x = |x - P_x(y)|$ là khoảng cách theo trục x giữa vật nặng và quỹ đạo P. Khi đó phản hồi (phần thưởng) mà mỗi tác tử nhận được cho hành động cuối cùng là $K * a^{-\Delta x}$. Giá trị K và a được chọn giống như trong [10]: K=50, a=1.15. Để đơn giản, các đại lượng có giá trị liên tục như lực, góc, khoảng cách được rời rạc hoá bằng cách chia đều thành những khoảng bằng nhau.

Trạng thái của bài toán được xác định bởi vị trí của vật nặng. Dễ dàng nhận thấy, từ một ví trí (trạng thái) hiện tại, một hành động chung cụ thể luôn dẫn đến một vị trí (trạng thái) xác định khác. Như vậy, bài toán di chuyển vật nặng có dạng *MDP xác định* như định nghĩa ở phần 2.

Không gian bài toán được giới hạn trong hình chữ nhật có toạ độ (0,0), (100,100). Quá trình học được tiến hành qua nhiều vòng, mỗi vòng bao gồm nhiều hành động của tác tử. Một vòng bắt đầu từ vị trí xuất phát S và kết thúc nếu xảy ra một trong ba tình huống sau: 1) tác tử đẩy được vật nặng tới đích; 2) vật nặng bị đẩy ra khỏi không gian cho phép; 3) số hành động vượt quá một giới hạn cho trước mà không xảy ra hai tình huống trước. Vượt hạn chế số lượng hành động cho phép tránh tình trạng tác tử không làm gì cả khi vật nặng nằm trên quỹ đạo mong muốn nhưng chưa đạt tới đích.



***Hình 2*: Bài toán di chuyển vật nặng với hai tác tử**

Nhiệm vụ của tác tử là học cách cùng nhau di chuyển vật nặng tới đích sau khi lặp lại các vòng học. Trong quá trình học, các tham số như K, γ được giữ nguyên không thay đổi. Quá trình học kết thúc khi tác tử di chuyển vật nặng thành công trong N vòng liên tiếp (N

được chọn bằng 10) hoặc khi số vòng học vượt qua một ngưỡng nhất định (1500). Mặc dù bài toán được mô tả và thực nghiệm chỉ gồm hai tác tử, có thể mở rộng bài toán cho trường hợp với nhiều tác tử hơn.

Vị trí bắt đầu và kết thúc được lựa chọn giống như trong [10] tương ứng là (40,0) và (40,100). Góc tác động của lực được chia thành 11 khoảng rời rạc, cường độ lực tác động được chia thành 10 khoảng bằng nhau. Hệ số chiết khấu được chọn $\gamma = 0.9$. Với các tham số được lựa chọn như vậy, không gian trạng thái sẽ bao gồm 10^4 trạng thái. Kích thước không gian trạng thái trong trường hợp này là tương đối nhỏ và do vậy trong quá trình cài đặt thuật toán, giá trị Q được biểu diễn dưới dạng bảng. Trong trường hợp không gian trạng thái lớn hơn, có thể sử dụng các phương pháp biểu diễn bảng Q dưới dạng rút gọn như dùng mạng nơron hay cây hồi quy (regression tree) [6].

Một vấn đề quan trọng trong học tăng cường là tác tử cần khảo sát số lượng đủ lớn trạng thái và hành động. Để giải quyết vấn đề này, tại mỗi trạng thái, tác tử lựa chọn hành động (hướng và cường độ tác động) ngẫu nhiên theo một phân bố xác suất nào đó. Có hai phương pháp chính. Phương pháp thứ nhất chọn một trong số các hành động cho phép với xác suất bằng nhau. Phương pháp này không quan tâm đến kinh nghiệm trước đó. Phương pháp thứ hai tính xác suất của hành động tiếp theo phụ thuộc vào giá trị của Q. Theo phương pháp này, tác tử chọn hành động tốt nhất tại thời điểm đó (hành động có giá trị Q tương ứng lớn nhất) với xác suất p và chọn các hành động khác với xác suất (1-p). Xác suất p thường được điều tăng dần theo thời gian do tác tử đã có thêm nhiều kinh nghiệm. Một phương pháp thông dụng loại này là phương pháp Boltzman được sử dụng cho thực nghiệm trình bầy trong báo cáo này. Theo phương pháp khảo sát theo Boltzman (*Boltzman exploration*): hành động a được chọn với xác suất

$$\frac{e^{Q(a)/T}}{\sum_{a'} e^{Q(a')/T}}$$

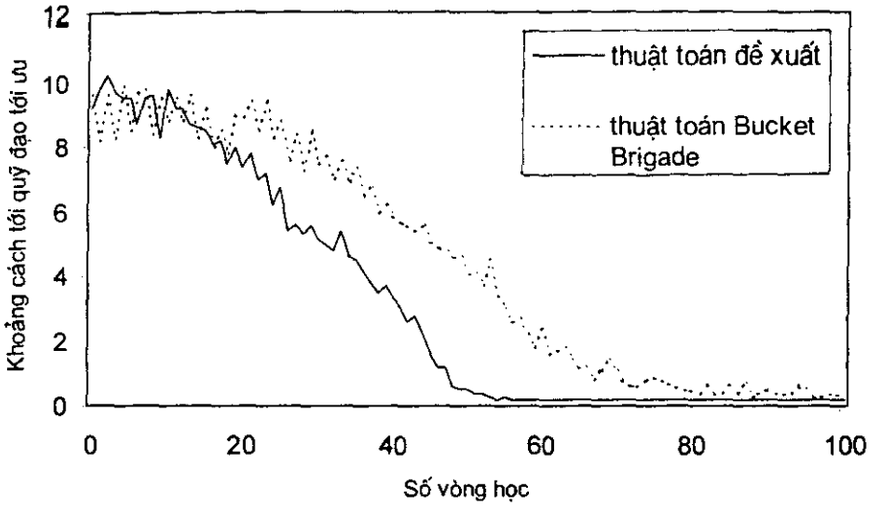
T là tham số và có giá trị giảm dần theo thời gian. Mục đích chính của phương pháp này là cho phép tác tử áp dụng sử dụng kinh nghiệm trước đó để thu hẹp không gian khảo sát.

Kết quả: Tiêu chí chính để đánh giá thuật toán học là số vòng học cần thiết trước khi thuật toán hội tụ. Một thông số nữa liên quan đến quá trình học là khoảng cách trung bình từ quỹ đạo thực đến quỹ đạo mong muốn. Giá trị các tiêu chí trên được xác định bằng cách lấy giá trị trung bình qua 100 lần thực hiện thuật toán. Giá trị Q được coi là hội tụ nếu $Q^{t+1} - Q^t \le \varepsilon$ với ε là một số dương nhỏ tuỳ ý.

Trong [10], bài toán di chuyển vật nặng với hai tác tử được giải quyết bằng cách sử dụng thuật toán học tăng cường *Bucket Brigade (BB)* cho hệ đa tác tử. Kết quả thử nghiệm thuật toán trình bầy ở trên được so sánh với kết quả thuật toán BB như trên đồ thị ở Hình 3.

Giá trị theo trục tung là khoảng cách trung bình từ quỹ đạo thực đến quỹ đạo mong muốn. Giá trị theo trục hoành là số lượng vòng học của tác tử. Kết quả thực hiện Bucket brigade lấy từ [10] với hệ số học $\gamma = 0.6$ và các tham số khác như K, a, tham số dùng để rời rạc hoá, kích thước không gian học, hạn chế về số bước trong mỗi vòng được lựa chọn giống như trình bầy ở trên.

Thuật toán trình bầy ở trên hội tụ sau khoảng 50 vòng. Sau khi hội tụ, tác tử hành động rất ổn định và di chuyển vật nặng theo đúng quỹ đạo mong muốn. Trong khi đó, theo kết quả của [10] thuật toán Bucket brigade cần khoảng 80 vòng trước khi hội tụ.



Hình 3. **Kết quả học tự động cho bài toán di chuyển vật nặng**

5. KẾT LUẬN

Báo cáo trình bầy một thuật toán học tăng cường cho hệ đa tác tử cộng tác. Thuật toán là một cải tiến của thuật toán học Q trong đó thay vì sử dụng bảng Q chung duy nhất, mỗi tác tử lưu trữ và cập nhật bảng Q riêng của mình. Tác tử chỉ tiến hành cập nhật giá trị bảng Q riêng nếu giá trị mới lớn hơn giá trị cũ. Bằng cách đồng thời ghi lại chiến lược hành động tốt nhất từng gặp, thuật toán đảm bảo tìm được hành vi tối ưu cho tác tử. Kết quả thử nghiệm cho thấy tác tử học được cách phối hợp hành động trong bài toán di chuyển vật nặng với thời gian học tương đối ngắn.

Thuật toán chỉ có thể áp dụng trong trường hợp hàm chuyển tiếp giữa các trạng thái là xác định. Một hạn chế khác của thuật toán là mỗi tác tử phải có thông tin về trạng thái và giá trị phần thưởng chung của hệ thống. Trong trường hợp phần thưởng chung được tổng hợp từ phần thưởng của từng tác tử, hệ thống cần tính toán phần thưởng chung và thông báo đến từng thành viên. Quá trình này có thể đòi hỏi chi phí liên lạc tương đối lớn.

LỜI CẢM ƠN

Nghiên cứu được thực hiện với sự hỗ trợ kinh phí của Hội đồng khoa học tự nhiên.

TÀI LIỆU THAM KHẢO

[1] M. Bowling, M. Veloso, Multiagent learning using a variable learning rate, *Artificial intelligence*, vol. 136, pp 215-250, Elsevier, 2002.

[2] J.A. Boyan, M.L. Littman, Packet routing in dynamically changing network: A reinforcement learning approach. *Advances in Neural Information Processing Systems*, 6, Morgan Kaufman, 1994.

[3] C. Claus, C. Boutilier, The dynamics of reinforcement learning in cooperative multiagent systems, *In Proc. of the 15th National Conference on Artificial Intelligence*, Madison, WI, AAAI press, 1998.

[4] R.H. Crites, A.G. Barto, Elevator group control using multiple reinforcement learining. *Machine learning*, vol. 33, pp 235-262, Kluwer Acad., 1998.

[5] J. Hu, M.P. Wellman, Multiagent reinforcement learning: theoretical framework and an algorithm, *In Proc. of the 15th International Conference on Machine Learning*, San Francisco, Morgan Kaufman, 1998.

[6] L.P. Kaebling, M.L. Littman, A.W. Moore, Reinforcement learning: A survey, *Journal of Artificial Intelligence Research*, Vol. 4, pp 237-285, Morgan Kaufman, 1996.

[7] M. Lauer, M. Riedmiller, An algorithm for distributed reinforcement learning in cooperative multiagent system, *In Proc. of the 17th International Conference on Machine Learning*, Stanford, CA, Morgan Kaufman, 2000.

[8] M.L. Litman, Markov games as a framework for multiagent reinforcement learning, *In Proc. of the 11th International Conference on Machine Learning*, New Brunswick, Morgan Kaufman, 1994.

[9] J. Schneider, W.K. Wong, A.W. Moore, M. Riedmiller, Distributed value functions, *In Proc. of the 16th International Conference on Machine Learning*, San Francisco, Morgan Kaufman, 1999.

[10] S. Sen, M. Sekaran, Individual learning of coordination knowledge, *Journal of Experimental and Theoretical AI*, pp 156-170, 1998.

[11] S. Singh, D. Bertsekas, Reinforcement learning for dynamic channel allocation in cellular phone systems, *Advances in Neural Information Processing Systems*, 9, MIT Press, 1997.

[12] R. Sutton, A. Barto. *Reinforcement Learning: An Introduction*, MIT Press, Cambridge, 1998.

[13] C. Watkins, P. Dyan, Q-learning. *Machine learning*, Vol. 8, 279-292, Kluwer Acad., 1992.

[14] G. Weiss (eds), Multiagent systems, a modern approach to DAI, MIT Press, Cambridge, 1999.

[15] D.H. Wolpert, K.R.Wheeler, K. Tumer, General principles of learning-based multiagent systems. *In Proc. of the 3th Conference on Autonomous Agents*, New York, ACM, 1999.