Ứng dụng thuật giải di truyền vào game xếp hình Tetris

Trần Việt Hoàng - khoa Khoa học máy tính, trường Đại học Công nghệ thông tin

Tóm tắt

Tetris là trò chơi tiêu chuẩn cho những nghiên cứu về Trí tuệ nhân tạo và Máy học. Bài viết của chúng tôi đầu tiên sẽ giới thiệu khái quát về trò chơi Tetris. Tiếp theo là lược sử hình thành và phát triển của các thuật toán áp dụng, qua đó là những thách thức mới. Từ những tài liệu nghiên cứu tiền nhiệm, chúng tôi nhận thấy Approximate dynamic prog-ramming (quy hoạch động thích nghi), Genetic algorithms (giải thuật di truyền) và Reinforcement learning (học tăng cường) đã góp phần tạo nên những giải pháp hiệu quả. Học hỏi từ các công trình tiền nhiệm, chúng tôi quyết định xây dựng một thuật toán A.I với chiến lược tham lam (greedy stralegy) với một hàm đánh giá meta-heuristic. Số điểm (score) đạt được sẽ đánh giá độ hiệu quả của thuật toán này, nó tương ứng với những hàng đã xóa. Và nếu thuật toán của chúng tôi đủ tốt, nó có thể tiến tới vô hạn. Trên thực tế, chúng tôi đã triển khai thuật toán này trên game Tetris có bàn cờ kích thước 22x10, với độ khó tăng dần và chưa có dấu hiệu thua cuộc.

Giới thiệu

Tetris đã được sử dụng hơn hai mươi năm như một lĩnh vực để nghiên cứu việc ra quyết định tuần tự trong điều kiện không chắc chắn. Do độ lớn của không gian quyết định khiến nó trở thành một vấn đề phức tạp nhưng có thể giải quyết được. Ngoài ra, Tetris cũng thể hiện tính thú vị là giới hạn thời gian đưa ra quyết định trước khi một mảnh Tetro-mino rơi xuống bề mặt của bàn cờ. Nó thường được coi là một lĩnh vực khá khó khăn. Cho đến nay, các thuật toán khác nhau đã mang lại các chiến lược chơi tốt. Nhưng chúng chưa đạt đến mức hiệu suất đạt được của những người chơi chuyên nghiệp chơi mà không bị áp lực về thời gian. Các nghiên cứu sâu hơn về trò chơi có khả năng đóng góp vào các chủ đề quan trọng trong trí tuệ nhân tạo và máy học. Bao gồm khám phá tính năng, học tự động các cấu trúc phân cấp hành động và học tăng cường.

Trong bài viết này, trước tiên, chúng tôi mô tả sơ lược về Tetris, các công trình liên quan về những thuật toán đã ứng dụng trong trò chơi cổ điển này. Ngoài ra, những nỗ lực của chúng tôi là làm cho sản phẩm trở nên đa dạng bằng cách mở rộng nhiều tính năng thử nghiệm nổi bật hơn. Cốt lõi của trò chơi Teris mới của chúng tôi là khả năng sáng tạo hiệu ứng và hàm heuristic cải tiến của thuật toán A.I.

Chúng tôi đã thảo luận về những hướng đi và thách thức hiện tại. Qua đó đã học được rất nhiều từ các nghiên cứu/báo cáo tiền nhiệm có liên quan, hầu hết chúng đều được coi là hoàn hảo. Chúng tôi lấy cảm hứng từ rất nhiều thứ, đặc biệt là Genetic algorithm (giải thuật di truyền). Vấn đề lớn nhất của chúng tôi là xây dựng một tác tử thú vị mới và mở rộng công việc hiện có về các vấn đề của Tetris. Nhằm tiếp cận với các trò chơi khác hoặc thậm chí là thế giới thực.

1. Trò chơi tetris

1.1. Lịch sử

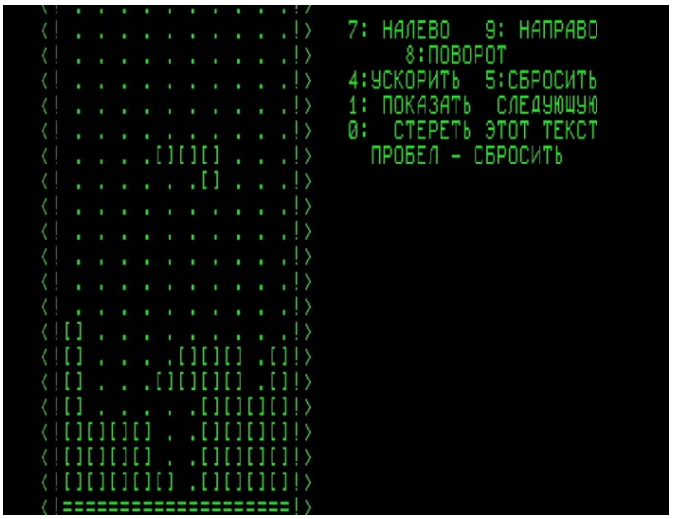
Tetris là một trò chơi xếp gạch do kỹ sư phần mềm người Nga Alexeu Pajinov tạo ra vào năm 1984. Nó đã được phát hành bởi một số công ty, nổi bật nhất là trong cuộc chiến tranh giành quyền sở hữu ứng dụng của trò chơi vào cuối những năm 1980. Sau một khoản thời gian phát hành phiên bản kinh điển của

Nintendo, quyền sở hữu trí tuệ được hoàn lại cho Pajitnov vào năm 1996, người đồng sáng lập công ty The tetris với Henk Roger để quản lý việc cấp phép Tetris.

1.2. Lối chơi

Trò chơi được chơi trên một bàn cờ có hai chiều, khởi tạo rỗng. Bàn cờ dần dần đầy lên khi các khối gạch có hình dạng khác nhau, được gọi là Tetromino, rơi từ trên xuống và rơi từng mảnh một. Người chơi có thể kiểm soát cách mỗi Tetromino tiếp đất bằng cách xoay nó và di chuyển nó theo chiều ngang, sang trái hoặc sang phải, bất kỳ số lần nào, Tetromino sẽ rơi từng hàng một cho đến khi một trong các ô của nó nằm ngay trên đầu của những viên gạch khác hoặc trên đáy bàn cờ. Khi toàn bộ hàng trở nên đầy, toàn bộ hàng này sẽ bị xóa, tạo thêm không gian trên bàn cờ và ghi điểm.[1]

Trò chơi kết thúc khi không còn khoảng trống trên đầu bàn cờ cho lượt Tetromino tiếp theo. Người chơi có thể trì hoãn trạng thái kết thúc này càng lâu thì điểm của họ sẽ càng cao.



Hình 1: The Original Tetris (1984), Retro by Old Classic Retro Gaming. Nguồn từ video: https://www.youtube.com/watch?v=omXjhq7DqQY

1.3. Phương pháp tiếp cận

Trước khi xây dựng tác tử cho Tetris, chúng ta hãy bắt đầu bằng cách xác định đâu là hướng có thể tiếp cận của thuật toán cho trò chơi này. Nói chung, tác tử của trò chơi này quyết định hành động sẽ chọn tiếp theo ở bất kỳ trạng thái nhất định nào của trò chơi dựa trên phần thưởng mà họ sẽ nhận được sau khi thực hiện hành động đó. Một khi tác tử chọn, trong số những hành động có thể thực hiện tiếp theo, tác tử sẽ trả lại kết quả tốt nhất. Tuy nhiên, rất khó và đôi khi không thể đánh giá giá trị của một hành động, và một cách giải quyết vấn đề này là đánh giá giá trị của trạng thái mà hành động sẽ dẫn đến. Cụ thể hơn, điều này đạt được bằng cách sử dụng một hàm đánh giá gán các giá trị số cho các trạng thái trò chơi. Và tác tử sẽ lựa chọn hành động dẫn đến trạng thái có giá trị tốt nhất. Đây cũng có thể được coi là một chiến lược tham lam, trong đó tác nhân đưa ra quyết định được đánh giá cao nhất ở mỗi bước.

Tetris được ước tính là có 7×2^{200} trạng thái. Với số lượng trạng thái khổng lồ này, phương pháp chung là ước lượng một hàm giá trị, hoặc đưa về một chính sách sử dụng một tập hợp các đặc trưng mô tả trạng thái hiện tại hoặc cặp trạng thái hiện tại - hành động. Trong bài báo này, các tác tử Tetris sử dụng những hàm đánh giá, đánh giá trạng thái bàn cờ hiện tai bằng cách gán các giá trị số. Với mỗi một Tetromino

xuất hiện, tác tử sẽ ra quyết định (dịch và xoay) của mảnh đó sẽ đưa ra trạng thái bàn cờ tốt nhất khi bị rơi. Tiếp theo, tác tử này sẽ mô phỏng và đánh giá tất cả các trạng thái của bàn cờ từ tất cả các nước đi có thể có của Tetromino, rồi chọn nước đi dẫn đến bàn cờ có giá trị tốt nhất. Cách tiếp cận phổ biến nhất đối với Tetris là phát triển một hàm đánh giá tuyến tính, trong đó mỗi vị trí có thể có của khối gạch này được đánh giá để chọn vị trí có giá trị cao nhất. Cụ thể về chức năng định giá sẽ được đề cập trong các phần sau. [2]

2. Các công trình liên quan

Approximate dynamic programming (quy hoạch động thích nghi) và Reinforcement learning (học tăng cường) là hai hướng tiếp cận phổ biến đã và đang được sử dụng Tetris. Hai thuật toán này xây dựng Tetris như là một tiến trình quyết định Markov (MDP) trong đó trạng thái được xác định bởi trạng thái bàn cờ hiện tại và Tetromino rơi tiếp theo, các hành động tiếp diễn là những nước đi khả thi của các khối gạch này, các vị trí có thể có mà nó có thể được đặt trên bàn cờ và phần thưởng được xác định sao cho việc tối đa hóa tổng phần thưởng dự kiến từ mỗi trạng thái, đồng thời với việc tối đa hóa điểm số từ trạng thái đó.

Tsitsiklis & Van Roy (1996) phát triển những ứng dụng đầu tiên của quy hoạch động thích nghi vào trò chơi Tetris. Họ đã sử dụng những đặc trưng đánh giá đơn giản như: số hố (các khoảng rỗng bị che lấp hoàn toàn bởi các Tetromino), và độ cao của cột cao nhất. Tác tử của họ đã xóa được 30 hàng trên bàn cờ 10x16.

Bertsekas và Ioffe đã đề xuất thuật toán λ-Policy Iteration (λ-PI) và áp dụng nó cho Tetris. Họ đã ước tính hàm giá trị như một tổng hợp tuyến tính của một bộ 22 gồm đặc trưng và kết quả thu được là 3,200 hàng.

Farias & Van Roy (2006) đã nghiên cứu một thuật toán lấy mẫu các ràng buộc dưới dạng phương trình Bellman cho một hệ giải lập trình tuyến tính. Hệ giải này thu được nghiệm là một chính sách có thể xóa khoảng 4,500 hàng bằng cách sử dụng các đặc trưng từng được phát triển bởi Bertsekas & Tsitsiklis (1996).

Giải thuật di truyền là giải thuật meta-heuristic, là tập con của thuật toán tiến hóa. Nó đóng góp một bước đột phá mới cho hệ thống giải trò chơi Tetris.

Bohm và cộng sự (2005) đã sử dụng các thuật toán tiến hóa để phát triển hệ giải Tetris. Trong quá trình thực hiện, tác tử không chỉ biết được khối Tetromino đang rơi mà còn biết được khối tiếp theo. Điều này làm cho kết quả của họ không thể so sánh với kết quả đạt được trên các phiên bản chỉ cập nhật trạng thái Tetrimino hiện tại. Tác tử triển khai theo cả hai chính sách tuyến tính và lũy từa. Kết quả của họ gồm 480,000,000 hàng được xóa bằng cách sử dụng hàm tuyến tính và 34,000,000 hàng bằng cách sử dụng hàm mũ, cả hai đều trên cùng một bàn cờ tiêu chuẩn.

Tiếp đến, Thiery & Scherrer (2009a; b) đã thêm một số đặc trưng (độ sâu hố và các hàng có hố) và phát triển hệ giải BCTS bằng cách sử dụng thuật toán cross-entropy, trong đó BCTS là viết tắt của bộ điều khiển xây dựng cho Tetris. Họ đã đạt được số điểm trung bình là 35,000,000 hàng được xóa.

Năm 2009, Boumaza đã giới thiệu một thuật toán tiến hóa khác cho Tetris, là chiến lược tiến hóa thích ứng của ma trận hiệp phương sai. Số điểm trung bình là 35,000,000 hàng được xóa. Sự thành công của các thuật toán di truyền xứng đáng được chúng ta chú ý đến. Do sự nổi lên gần đây của các chiến lược tiến hóa, được xem là những đối thủ cạnh tranh mạnh mẽ với các thuật toán học tăng cường. [3]

3. Phương pháp

Lấy cảm hứng từ các Thuật toán di truyền tiền nhiệm và chiến lược tìm kiếm tham làm với các hàm heuristic. Chúng tôi đã phát triển một tác tử AI đủ thông minh để xóa các hàng một cách vô thời hạn trong trò chơi tetris đơn giản (rõ ràng trò chơi phải dừng lại ở một điểm nhất định, do phần cứng của chúng tôi không có khả năng vô hạn như vậy).

3.1. Định hình trò chơi

Mặc dù phiên bản tiêu chuẩn của Tetris là trò chơi kinh điển, nhưng vẫn tồn tại rất nhiều biến thể của trò chơi. Để có sự rõ ràng cho sự trực quan và phương pháp tiếp cận, chúng tôi xây dựng tác tử AI từ nền tảng có các quy tắc như sau:

- Bàn cờ Tetris có kích thước hàng và cột lần lượt là 10 và 22, thêm vào đó là có 2 hàng bị ẩn ở trên.
- Tất cả các Tetrominoes (các mảnh Tetris) sẽ bắt đầu ở giữa 2 hàng trên cùng.
- Có tất cả bảy loại Tetromino: 'I', 'O', 'J', 'L', 'S', 'Z', 'T'.
- Tác tử sẽ biết trước Tetrimino rơi xuống tiếp theo (look-ahead).

3.2. Bốn đặc trưng đánh giá

Cũng giống như những người chơi đã chơi cho đến nay. Mục tiêu của chúng tôi là xóa càng nhiều đường càng tốt, nói cách khác là hoãn trạng thái kết thúc nhất có thể..

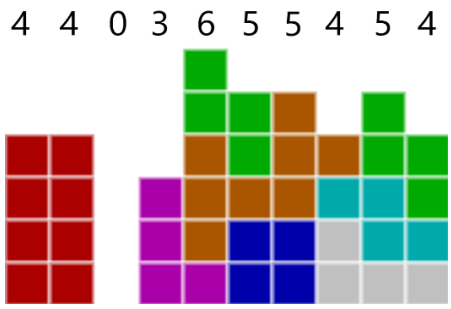
Để đạt được mục tiêu này, tác tử của chúng tôi sẽ quyết định nước đi tốt nhất cho một Tetromino nhất định bằng cách suy tính tất cả các động tác có thể (xoay và vị trí). Nó tính điểm số (điểm đánh giá) cho mỗi nước đi có thể (look-ahead) và chọn một nước có điểm tốt nhất làm nước đi tiếp theo.

Điểm số cho mỗi bước di chuyển được tính bằng cách đánh giá trạng thái bàn cờ mà bước di chuyển sẽ dẫn đến. Kết quả này dựa trên bốn đặc trưng: tổng chiều cao của trạng thái, số đường đã xóa, hố và độ gập ghềnh. Mỗi đặc trưng này tác tử sẽ cân nhắc, hoặc là cố gắng giảm thiểu, hoặc là tối đa hóa.

3.2.1. Tổng chiều cao của trạng thái

Đặc trưng này cho chúng ta biết lưới "cao" như thế nào. Để tính toán chiều cao tổng hợp, chúng tôi lấy tổng chiều cao của mỗi cột (khoảng cách từ ô cao nhất trong mỗi cột đến cuối bàn cờ).

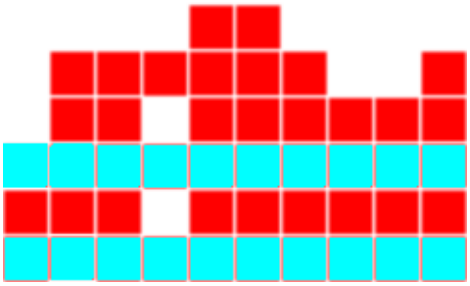
Tất nhiên chúng tôi muốn *giảm thiểu* giá trị này, vì chiều cao tổng hợp thấp hơn có nghĩa là chúng tôi có thể thả nhiều Tetromino hơn vào bàn cờ trước khi chạm vào đỉnh của nó.



Hình 3: Tổng chiều cao của trạng thái này là 48.

3.2.2. Số hàng đã xóa

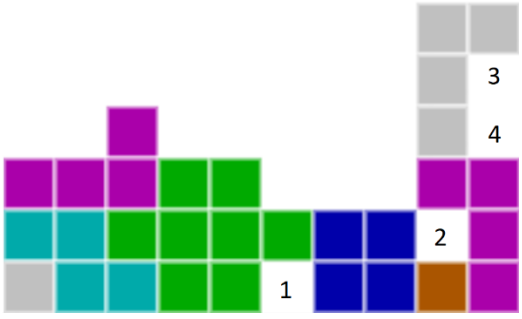
Đây có lẽ là đặc trưng dễ hiểu nhất trong số bốn đặc trưng. Nó chỉ đơn giản là số hàng đã hoàn thành trong một bàn cờ. Chúng tôi muốn *tối đa hóa* số hàng đã hoàn thành này, do mục tiêu của AI là xóa càng nhiều hàng càng tốt và việc xóa các hàng này sẽ cho chúng ta nhiều khoảng trống hơn để có nhiều khối gạch hơn.



Hình 4: Số hàng đã xóa là 2

3.2.3. Số hố

Như đã đề cập ở trên, một hố được định nghĩa là một khoảng trống sao cho có ít nhất một ô trong cùng một cột phía trên nó. Hố thì khó xóa hơn, do chúng ta phải xóa tất cả các đường phía trên trước khi chúng tôi có thể chạm tới hố và lấp đầy nó. Do đó, chúng tôi sẽ phải *giảm thiểu* số các hố này.



Hình 5: Số hố hiện tại là 2

3.2.4. Độ gập ghềnh

Từ Hình 3, xét trường hợp sau, các Tetromino xếp trên bàn cờ của chúng ta và chừa ra một cái *giếng sâu* (được đánh số 0). Điều này thật khó chịu, bởi vì: Những giếng này có thể được lấp đầy một cách dễ dàng, và thật khủng khiếp khi giếng bị chặn bởi những khối gạch. Và sau đó, rất khó để có thể lấp đầy. Để khái quát hóa ý tưởng về giếng, chúng tôi định nghĩa một đặc trưng được đặt tên là *độ gập ghềnh*.

Độ gập ghềnh cho chúng ta biết sự thay đổi chiều cao của từng cột trong bàn cờ. Nó được tính bằng cách tính tổng các giá trị tuyệt đối của hiệu chiều cao hai cột liền kề.

Từ các chỉ số chiều cao của các cột trên bàn cờ của Hình 3, dưới đây là cách tính độ gập ghềnh:

$$|4-4| + |4-0| + ... + |4-5| + |5-4| = 14$$

Mục tiêu của chúng tôi là giảm thiểu sự xuất hiện và độ sâu của những cái giếng này. Vì vậy, đây là giá trị cần được *giảm thiểu*.

3.2.5. Kết hợp các đặc trưng đánh giá

Bây giờ chúng tôi sẽ tính điểm của một trạng thái bàn cờ, bằng cách lấy sự kết hợp tuyến tính của bốn đặc trưng đã đề cập ở trên. Hàm đánh giá của tác tử có công thức như sau:

hàm đánh giá = a*(*tổng chiều cao của trạng thái*) + b*(*số hàng đã xóa*) + c*(*số hố*) + d*(*độ gập ghềnh*)

Trong đó a, b, c, d là các tham số ràng buộc. Như đã phân tích chức năng và nhiệm vụ của bốn đặc trưng, để tối đa hóa khả năng của hàm đánh giá này, chúng tôi đặt a, c, d âm và b dương.

Để có được hàm đánh giá tối ưu một cách thuyết phục, chúng tôi tạo ra bốn tham số trên bằng cách sử dụng Giải thuật di truyền, chúng tôi sẽ hiển thị kết quả của các tham số này sau.

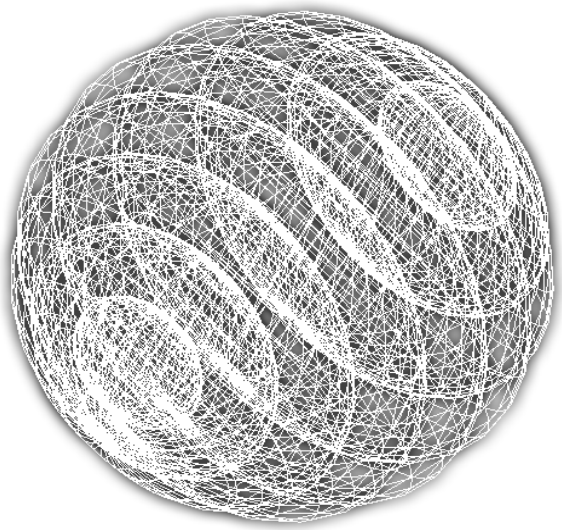
Bằng cách sử dụng bộ thông số này, AI có thể chọn nước đi tốt nhất có thể bằng cách giải trừ tất cả các nước đi có thể có (bao gồm cả phần look-ahead) và chọn nước đi có điểm đánh giá cao.

3.3. Tối ưu bộ tham số bằng Giải thuật di truyền

Mỗi bộ tham số (a, b, c, d) có thể được biểu diễn dưới dạng vectơ trong không gian $\mathbb{R}^4$.

$$\vec{p} = (a, b, c, d)$$

Một Giải thuật di truyền chuẩn cho các gen có giá trị thực sẽ liên quan đến việc tìm kiếm trong toàn bộ không gian $\mathbb{R}^4$ để có một tập hợp các tham số tối ưu. Tuy nhiên, trong bài viết này, chúng ta chỉ cần xem xét các điểm trên unit 3-sphere (hay là hình cầu đơn vị trong không gian 4 chiều). Điều này là do hàm đánh giá được định nghĩa là một hàm tuyến tính và kết quả so sánh là bất biến theo tỷ lệ.



Hình 7: minh họa hình cầu trong không gian 4 chiều.

Để dễ hình dung, trong hình cầu 4 chiều, mọi mặt phẳng di chuyển dọc theo một vectơ tọa độ có thể biểu diễn thành các hình cầu khác. Chúng tôi sẽ hạn chế việc chỉ tìm kiếm các điểm trên bề mặt của khối unit 3-sphere, vì bề mặt của khối cầu bao gồm tất cả các hướng có thể.

Về mặt toán học, hàm đánh giá trong ngữ cảnh của vectơ được biểu diễn như sau:

$$f(\vec{x}) = \vec{p}\ (*)\ \vec{x}$$

Với $\vec{p}$ là bộ tham số (a, b, c, d), $\vec{x}$ là bộ đặc trưng (*tổng chiều cao của trạng thái, số hàng đã xóa, số hố, độ gập ghềnh*), f() là hàm đánh giá cho mỗi nước đi và (*) là phép nhân tích chập.

Giả sử chúng ta có hai nước đi, chúng tôi sẽ so sánh - move1 và move2, và lựa chọn nước đi nào tốt hơn (cũng nghĩa là move nào cho điểm đánh giá cao hơn). Giả sử hai nước đi đó được biểu diễn dưới dạng vectơ $\overrightarrow{x_1}$ và $\overrightarrow{x_2}$. Nếu move2 cho kết quả tốt hơn move1, thì chúng ta phải kiểm tra điều kiện if $f(\overrightarrow{x_2}) > f(\overrightarrow{x_1})$ hoặc $f(\overrightarrow{x_2}) - f(\overrightarrow{x_1}) > 0$.

Điều kiện kiểm tra có thể viết lại thành:

$$f(\overrightarrow{x_2}) - f(\overrightarrow{x_1}) = \vec{p}(\overrightarrow{x_2} - \overrightarrow{x_1})$$

Do tất cả các vectơ tham số cùng hướng tạo ra kết quả tương đương. Vì vậy, chỉ cần xem xét một vectơ duy nhất cho mỗi hướng là đủ.

Tiếp theo, chúng tôi cần xác định Hàm fitness, Tournament selection, Toán tử crossover, Toán tử đột biến và tái tổ hợp được sử dụng để tối ưu bộ tham số $\vec{p}$.

3.3.1. Hàm fitness

Lấy ý tưởng từ Học tăng cường, Hàm fitness được khởi tạo ngẫu nhiên với 100 vectơ tham số $\vec{p}$ (các chỉ số được ràng buộc theo unit 3-sphere hoặc khởi tạo ngẫu nhiên từ -0.5 đến 0.5). Mỗi $\vec{p}$ sẽ chơi 10 ván và mỗi ván sẽ rơi nhiều nhất 500 Tetro-mino. Sau đó, chúng tôi sẽ kiểm tra xem mỗi $\vec{p}$ xóa được tất cả bao nhiêu dòng. Một vài vectơ cho ra kết quả xấu nhất khi chúng không thể xóa bất kỳ dòng nào. Hàm fitness của $\vec{p}$ là số dòng mà $\vec{p}$ có thể xóa:

$$fitness(\vec{p}) = \text{ tổng số hàng mà } \vec{p} \text{ có thể xóa}$$

3.3.2. Tournament selection

Các cá thể mẹ được chọn để tạo ra thế hệ tương lai bằng cách sử dụng Tournament selection. Chúng tôi chọn khoảng 10% quần thể một cách ngẫu nhiên và hai cá thể có điểm fitness cao nhất. Từng cá thể trong quần thể con này tiến hành crossover để tạo ra con cái mới. Quá trình này được lặp lại cho đến khi số lượng off-spring mới được tạo ra đạt 30% kích thước quần thể (bằng 30).

3.3.3. Toán tử Crossover

Đối với Crossvoer, cho hai vectơ cha mẹ là $\overrightarrow{p_1}$, $\overrightarrow{p_2}$ kết hợp với nhau bằng cách cộng hai vectơ đã nhân với điểm fitness của chính nó. Vectơ cá thể con $\vec{p}$ được hình thành dưới dạng như sau:

$$\vec{p} = \overrightarrow{p_1} fitness(\overrightarrow{p_1}) + \overrightarrow{p_2} fitness(\overrightarrow{p_2})$$

Về bản chất, vectơ con vẫn ràng buộc theo "unit 3-sphere", nhưng thiên về phía cá thể mẹ nào có điểm fitness cao hơn. Cá thể mẹ nào mang điểm fitness cao hơn sẽ mang tính trạng trội, ngược lại mang tính trạng lặn. Các vectơ con sẽ được chuẩn hóa (normalization).

3.3.4. Toán tử đột biến

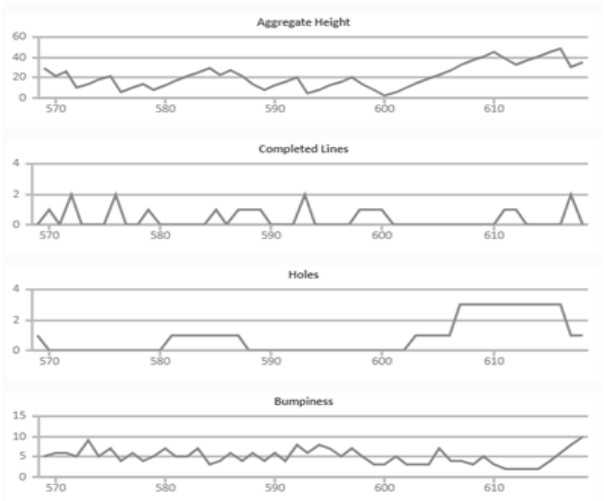
Toán tử đột biến chúng tôi chọn khá đơn giản. Mỗi vectơ con được tạo ra có một cơ hội nhỏ (5%) để đột biến. Nếu nó thực sự đột biến, thì một thành phần ngẫu nhiên (một trong bốn tham số a, b, c, d) của vectơ con sẽ được điều chỉnh bởi một lượng ngẫu nhiên lên đến 0.2. Sau đó vectơ được chuẩn hóa (bằng cách chia từng thành phần cho độ dài của vectơ).

3.3.5. Thay thế các cá thể yếu kém

Khi số cá thể con tạo ra đạt ngưỡng 30%. Những cá thể yếu nhất (có ít điểm fitness nhất trong quần thể) sẽ bị xóa và thay thế bởi các vectơ con mới được tạo ra. Đây là sự biến đổi thế hệ tiếp theo của quần thể. Kích thước quần thể vẫn giữ nguyên. Toàn bộ quá trình có thể lặp lại cho đến khi fitness của quần thể chấp nhận được. [4] [5]

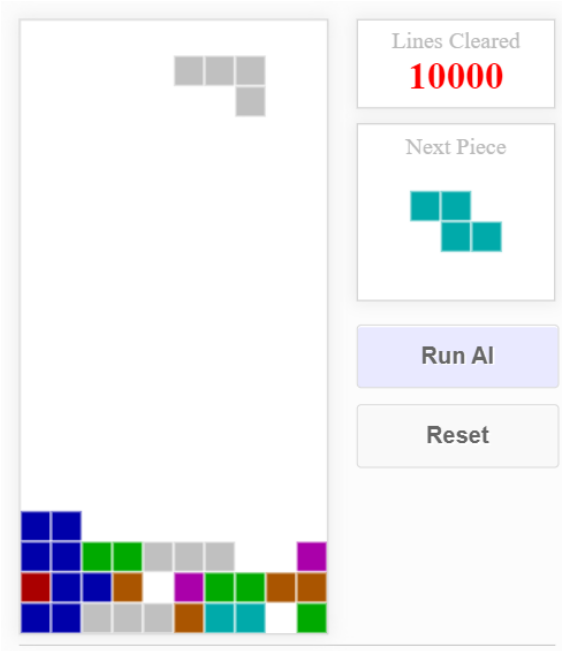
4. Kết quả

Nhìn vào hình dưới đây, chúng ta có thể thấy tác tử hoạt động khá tốt. Các hố được giảm đến mức tối thiểu trong khi ba thông số còn lại được cân bằng.



Hình 8: Biểu đồ các thông số theo thời gian

Biểu đồ xung ở trên chỉ mô tả những gì đang xảy ra vào một khoảng thời gian nhất định. Việc đánh giá hiệu năng nên là tổng số hàng mà tác tử này có thể xóa.



Đây là kết quả thu được của chúng tôi, chúng tôi đã chạy thử trong 3 giờ và xóa tổng cộng 10000 dòng, nó vẫn chạy và chưa có dấu hiệu thua cuộc.

5. Kết luận

Chúng tôi đã sử dụng chiến lược tham lam kết hợp với hàm đánh giá. Vấn đề lớn nhất của chúng tôi là chúng tôi đã tùy tiện chọn ra rất nhiều tham số. Chúng có thể hoạt động tốt hoặc có thể không, nhưng chúng tôi không thực sự biết liệu có những giá trị tốt hơn cho chúng hay không. Do đó, chúng tôi đã cố gắng tối ưu hóa các thông số này bằng Giải thuật di truyền.

Giải thuật di truyền là thuật toán có tên gọi được lấy cảm hứng từ Thuyết tiến hóa của Mendel. Các thành tố của thuật toán này được trình bày vắn tắt như sau:

- Một *nhiễm sắc thể* biểu thị một giải pháp khả thi cho vấn đề dưới dạng một chuỗi.
- Hàm fitness lấy một nhiễm sắc thể làm đầu vào và trả về giá trị fitness
- Một *quần thể* gồm rất nhiều các nhiễm sắc thể
- Một phương pháp *selection* xác định cách các cá thể mẹ được chọn để nhân giống từ quần thể.
- Một toán tử *crossover* xác định cách các cá thể mẹ kết hợp để tạo ra cá thể con.
- Toán tử đột biến xác định cách đột biến của các cá thể.

Chúng tôi nhận thấy việc huấn luyện và hiệu năng của tác tử AI được xem là thành công. Đây là những vấn đề còn lại mà chúng tôi cần cải thiện trong những công trình sau này.

Về hàm đánh giá heuristic

Đặc trưng tổng chiều cao trạng thái là hệ số rất lớn, áp đảo mọi đặc trưng khác và có thể gây ra sự dư thừa. Mặc dù đúng là tổng chiều áp dụng cho mọi khối gạch trên trên bàn cờ, nhưng nó không thực sự hoạt động như thế mà chỉ ảnh hưởng đến những khối Tetromino rơi hiện tại. Lý do là, phần còn lại của bàn cờ - mọi thứ trừ khối rơi hiện tại - sẽ không đổi cho dù nó đi đến đâu. Giống như việc bạn bỏ phiếu cho tất cả mọi người, bạn thực sự khổng bỏ phiếu cho ai cả vì phiếu bầu của bạn không ảnh hưởng đến kết quả.

Đặc trưng các hàng đã xóa cũng có một chút sai sót. Trong khi b (tham số thứ hai) có trọng số dương đối với việc xóa một hàng, nó vẫn xóa các đường bất cứ khi nào có thể: và lần nữa ràng buộc trở lại hệ số chiều cao.Việc xóa toàn bộ một hàng các khối - và thậm chí là xóa nhiều hàng đó sẽ tác động rất lớn đến hệ số chiều cao.

Về Giải thuật di truyền

Hàm fitness đôi lúc sẽ gây rắc rối. Vào thời điểm AI đã và đang xóa vài nghìn hàng trên bàn cờ 22 cột, điều duy nhất khiến AI rơi vào bế tắc là một chuỗi các khối S và Z khó xử lý. Thực tế điều này cũng gây khó dễ cho con người - và trong bất kỳ bộ tham số được tạo ngẫu nhiên nào, bạn cũng sẽ nhận được một loạt các Tetromino không may mắn như thế. Vì vậy, ở các thế hệ sau, nếu chỉ đơn giản mô phỏng một không gian Tetris 22 cột thì khá là tệ trong việc phân loại AI tốt và AI xuất sắc.

Selection cũng gặp vấn đề. Sau mười thế hệ, toàn bộ quần thể ít nhiều đều có các giá trị giống nhau, chỉ có một số thay đổi nhỏ - hơi mỉa mai vì tình huống này cũng là tình huống mà thuật toán *Tournament selection* được lập trình để ngăn chặn

Mặc dù Giải thuật di truyền có thể hội tụ về tối ưu cục bộ khá nhanh chóng, tạo ra một giải pháp phù hợp, nhưng rất khó để nó đạt được một giải pháp tối ưu toàn cầu - một giải pháp tốt nhất của bài toán. Việc thay đổi bất kỳ một giá trị nào trong bộ tham số sẽ gây hại nghiêm trọng cho cá thể, nhưng việc thay đổi hai hoặc nhiều giá trị đồng thời theo một cách nhất định sẽ làm cho nó tốt hơn. Đây là một nhược điểm đối với các thuật toán di truyền nói chung.

Tài liệu tham khảo

[1] Many Authors, "Tetris," Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Tetris. [Accessed 28 6 2020].

[2] M. G. B. S. Victor Gabillon, "Approximate Dynamic Programming Finally Performs Well in the Game of Tetris," p. 9, 2013.

[3] Ö. Ş. Simón Algorta, "The Game of Tetris in Machine Learning," p. 9, 23 8 2017.

[4] B. Li, "Coding a Tetris AI using a Genetic Algorithm," LUCKY'S NOTES, 27 5 2011. [Online]. Available: https://luckytoilet.wordpress.com/2011/05/27/coding-a-tetris-ai-using-a-genetic-algorithm/. [Accessed 8 7 2020].

[5] V. Mallawaarachchi, "Introduction to Genetic Algorithms — Including Example Code," Towards Data Science, 8 7 2017. [Online]. Available: https://towardsdatascience.com/introduction-to-genetic-algorithms-including-example-code-e396e98d8bf3. [Accessed 9 7 2020].

[6] H. R. Luc Lamontagne, "Reinforcement of Local Pattern Cases for Playing Tetris," 1 2008.