**Các lý thuyết nền tảng tạo nên thuật toán Monte Carlo Tree Search**

**Markov Decision Process (MDPs)**

Markov decision process là một mô hình toán học dùng để giải quyết các vấn đề yêu cầu đưa ra các quyết định tuần tự trong một môi trường mà chủ thể có thể quan sát đc đầy đủ. Mô hình toán học MDP gồm 4 thành phần:

* + S: một tập hợp các trạng thái với s0 là trạng thái ban đầu
  + A: một tập hợp các hành động
  + T(s, a, s’): là một mô hình chuyển đổi sẽ đưa ra xác suất để đạt đc trạng thái s’ nếu áp dụng hành động a cho trạng thái s
  + R(s): là một hàm tính điểm

Mục tiêu của MDP là dựa trên mô hình nói trên để tìm ra một quy luật hay chiến lược đem lại phần thưởng cao nhất cho mỗi quyết định đc đưa ra từ đó có thể đạt đc mục đính mong muốn

**Partially Observation Markov Decision Process (POMDP)**

Khi chủ thể không thể quan sát đầy đủ môi trường xung quanh thì mô hình Partially Observable Markov Decision Process sẽ đc áp dụng thay thế cho MDP.Tương tự MDP, POMDP cũng hướng đến việc tìm ra một chiến lược hay quy tắc giúp cho chủ thể đạt được mục đích mong muốn trong môi trường mà chủ thể không thể quan sát đầy đủ. Do thiếu thông tin về môi trường xung quanh nên quá trình đưa ra chiến lược tối ưu sẽ phức tạp hơn so với MDP

**Lý thuyết trò chơi**

Lý thuyết game đc kế thừa từ lý thuyết quyết định trong trường hợp có nhiều tác nhân tương tác với nhau. Một trò chơi đc định nghĩa là một tập hợp các quy tắc đc thiết lập trước và một hay nhiều tác nhân sẽ thực hiện các quy tắc trên để tạo ra một kết quả. Một trò chơi bao gồm các thành phần sau:

* + S: là tập hợp các trạng thái của trò chơi, s0 là trạng thái bắt đầu
  + St C S: là một tập các trạng thái dẫn đến kết thúc trò chơi
  + N thuộc N: là số lượng người chơi
  + A: là tập hợp các hành động
  + F: S x A => S: là hàm chuyển đổi trạng thái
  + R: S => Rk: là hàm cho biết giá trị lợi ích (hiểu đơn giản là điểm) của từng người chơi tại trạng thái S
  + P: S => (0, 1, 2, …): là hàm xác định người chơi nào sẽ đc hành động ở trạng thái S

Mỗi game sẽ bắt đầu từ trạng thái S0 và thay đổi trang thái theo thời gian t = 1, 2, … cho đến khi đạt đc một trạng thái St. Mỗi người chơi sẽ thực hiện một hành động, hành động này thông qua hàm f dẫn chuyển đổi từ trạng thái St thành St + 1. Sau mỗi lượt hành động, người chơi sẽ nhận đc 1 điểm số đc tính toán bằng hàm R. Giá trị của điểm số thông thường sẽ là 0 cho các trạng thái không phải St, 1 cho các trạng thái đưa chủ thể đến chiến thắng, -1 với trạng thái khiến chủ thể thua cuộc và 0 đối với các trạng thái có kết quả hòa

**Multi Armed Bandit–Based Method**

Bài toán Multi Armed Bandit (MBA) đặt ra tình huống khi bạn có nhiều hành động khác nhau, mỗi hành động sẽ đem lại 1 phần thưởng có giá trị giao động trong một khoảng cố định. Bài toán yêu cầu bạn tối đa hóa phần thưởng thông qua việc chọn ra hành động mang lại phần thưởng cao nhất. Tuy nhiên bạn không biết về phần thưởng mà môt hành động trực tiếp mang lại mà phải thông qua việc thống kê từ các lần mà hành động đó đã được thực hiện trong quá khứ.

Để tối đa đc phần thưởng ta cần cân bằng giữa 2 khái niệm “Khai thác” (Exploration) và “Thăm dò” (Exploitation)

* Khai thác đề cập đến việc khi ta biết được đâu là hành động mang lại phần thưởng cao nhất, ta sẽ tiếp tục thực hiện hành động lại đó để tối đa phần thưởng đạt được. Khai thác giúp ta tận dụng hành động mang lại phần thưởng cao để có thể tối đa tổng phần thưởng đạt được.
* Thăm dò đề cập đến việc ta sẽ thực hiện tất cả các hành động một số lần cho đến khi ta có đủ dữ kiện để tính toán xem đâu là hành động sẽ mang lại phần thưởng cao nhất. Thăm dò giúp tìm ra các hành động có tiềm năng sẽ mang lại phần thưởng cao hơn hành động đang khai thác.

UCB1 là một trong các chiến lược giúp cân cân bằng “Khai thác” và “Thăm dò” một cách hiệu quả và nó cũng được áp dụng vào thuật toán MCTS. Trong chiến lược UCB, tại lần thứ t, ta sẽ chọn hành động có r có giá trị lớn nhất được tính theo công thức sau:

* + là phần thưởng trung bình của hành động r
  + là tổng số lượt chọn hành động
  + là số lần mà hành động r đã được chọn cho đến thời điểm t

Phần của công thức đại diện cho sự “Khai thác”. Khi ta chọn hành động có giá trị cao nhất được tính theo công thức trên đồng nghĩa với chọn ra hành động r có phần thưởng trung bình cao**.** Tuy nhiên giá trị trung bình không phải là yếu tố duy nhất quyết định hành động r có được chọn hay không.

Phần của công thức đại diện cho sự “Thăm dò” hiểu cách khác là mức khuyến cáo nên chọn hay không của hành động r. Trong bài toán MAB ta cần tránh việc chỉ chọn hành động có phần thưởng trung bình cao mà bỏ qua việc thăm dò các hành động khác. Nếu một hành động hiện đang có phần thưởng trung bình ở mức thấp được chọn đủ nhiều để có thêm dữ liệu cho việc ước lượng phần thưởng thì nó có thể trở thành hành động mang lại phần thưởng trung bình cao nhất. Theo công thức khi một hành động r được chọn nhiều lần tức Nt(r) tăng kéo theo giá trị của phần công thức trên giảm từ đó làm giảm giá trị của toàn bộ công thức cho dù phần thưởng trung bình µr đang cao. Điều này đồng nghĩa với độ khuyến cáo chọn hành động r thấp hơn so với các hành động khác vì các hành động khác được chọn ít hơn tức Nt(r) nhỏ dẫn đến mức độ khuyến cáo chọn cao hơn. Từ đó chuyển từ trạng thái khai thác hành động r sang khám phá các hành động khác để tránh được tình huống nêu trên trong bài toán MAB. Một góc nhìn khác là nếu như một hành động chưa từng được lựa chọn hay Nt(r) = 0 sẽ dẫn đến giá trị của phần công thưc trên là ∞ nên đảm bảo các node chưa được chọn sẽ được chọn ít nhất một lần tránh bỏ qua các node có tiềm năng mang lại phần thưởng cao.

**Monte Carlo method**

Phương pháp Monte Carlo được sử dụng để tính toán các bài toán phức tạp mà các phép tính thông thường rất khó để tính toán và thường là các bài toán có liên quan đến nhiều biến số.Phương pháp Monte Carlo sử dụng máy tính để tạo ra một số lượng lớn các mô phỏng ngẫu nhiên (giá trị đầu vào ngẫu nhiên) của bài toán và dựa vào kết quả của các lần mô phỏng để ước lượng kết quả của bài toán.Độ chính xác của kết quả thu được phụ thuộc vào số lượng mô phỏng được thực hiện và độ biến thiên của các mô phỏng. Số lượng và độ biến thiên càng lớn thì kết quả sẽ càng chính xác.

**Monte Carlo Tree Search**

Trong phạm vị bài viết, ta sẽ dùng thuật toán MCTS để đưa ra quyết định trong một trò chơi

Mục tiêu của thuật toán MCTS tính toán và đưa ra nước đi mang lại giá trị lớn nhất từ một trạng thái của trò chơi được cung cấp.

Dựa trên mô hình MDP và lý thuyết trò chơi, thuật toán MCTS tạo ra một môi trường mô phỏng các trạng thái có thể xảy ra từ một trạng thái gốc được cung cấp. Ý tưởng của thuật toán dựa trên 2 quá trình

* + Tính toán giá trị mà những hành động có thể thực hiện từ trạng thái gốc mang lại thông qua phương pháp mô phỏng Monte Carlo.
  + Dùng kết quả của mỗi hành động để tối ưu cũng như điều chỉnh quá trình tìm ra hành động thông qua chiến lược UBC1 đem lại giá trị lớn nhất cho trạng thái gốc được cung cấp

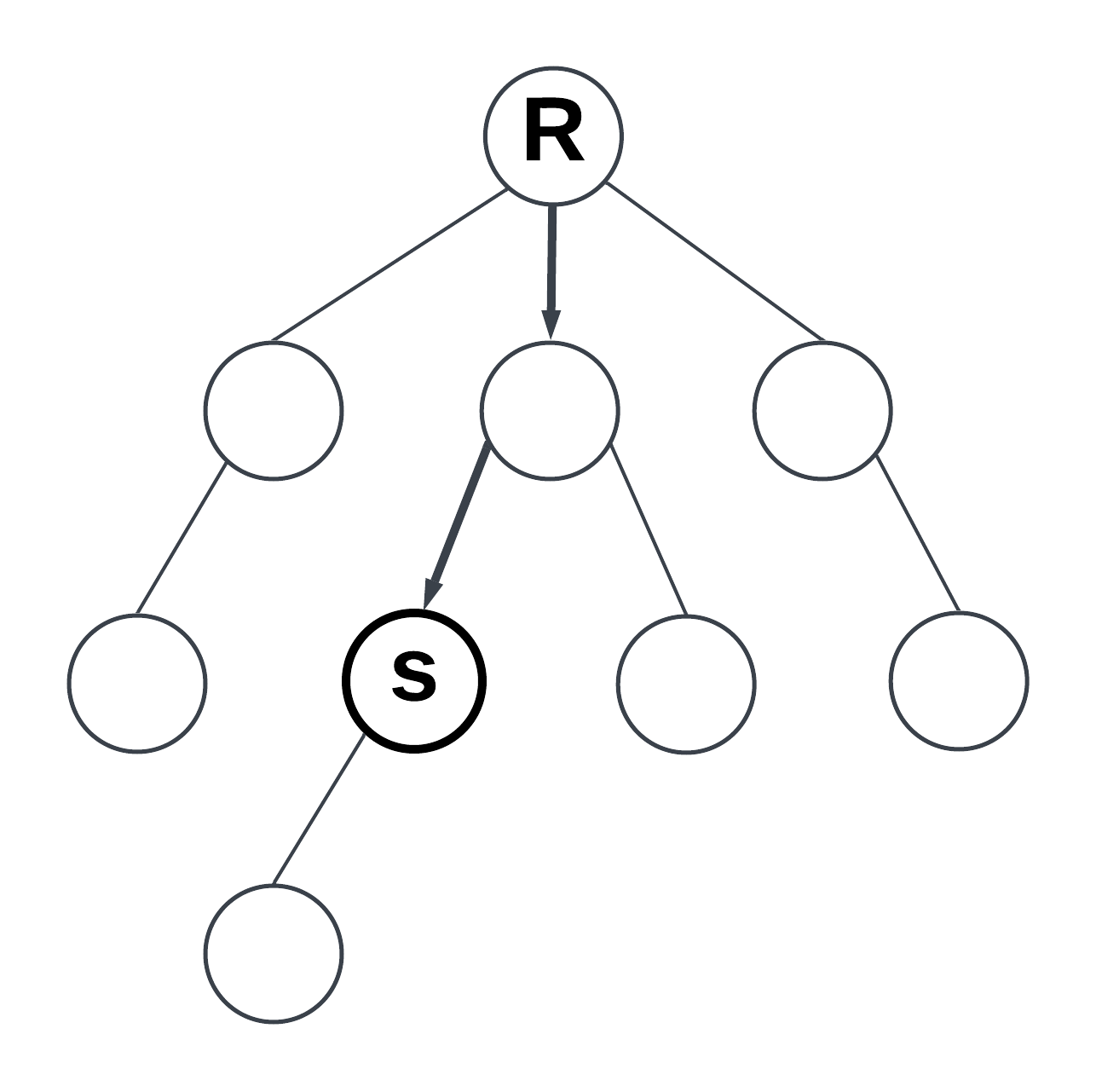
Môi trường mô phỏng được biểu diễn bằng cấu trúc dữ liệu cây. Cây tìm kiếm Monte Carlo sẽ được xây dựng bằng việc lặp lại 2 quá trình nêu trên. Mỗi node trong cây biểu thị một trạng thái của trò chơi, các node con là các trạng thái khi thực hiện các hành động hợp lệ từ node cha.

**Các bước trong 1 vòng lặp để xây dựng cây tìm kiếm Monte Carlo**

Mỗi vòng lặp của thuật toán sẽ trải qua 4 bước

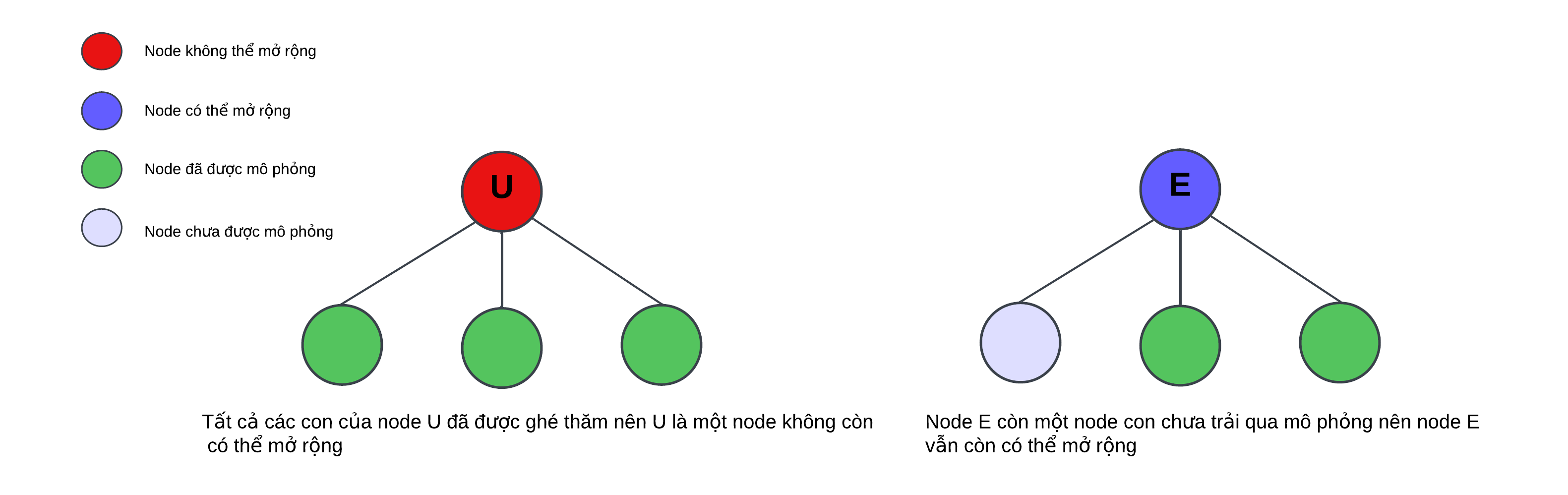
**Lựa chọn (Selection)**

Bắt đầu từ Node gốc R, duyệt qua toán bộ các node trong cây theo chiến lược **UCT**(sẽ được đề cập sau) thể cho đến khi gặp được một node **S** **có** **thể mở rộng** hoặc node **T biểu diễn trạng thái kết thúc**, nếu node T được chọn tiến thằng đến bước lan truyền ngược bỏ qua hai bước bên dưới.



Node biểu diễn trạng thái kết thúc (terminated node) là node mà nó biểu diễn trạng thái trạng thái đã phân định thắng thua hoặc trạng thái mà game không thể tiếp tục được chơi.

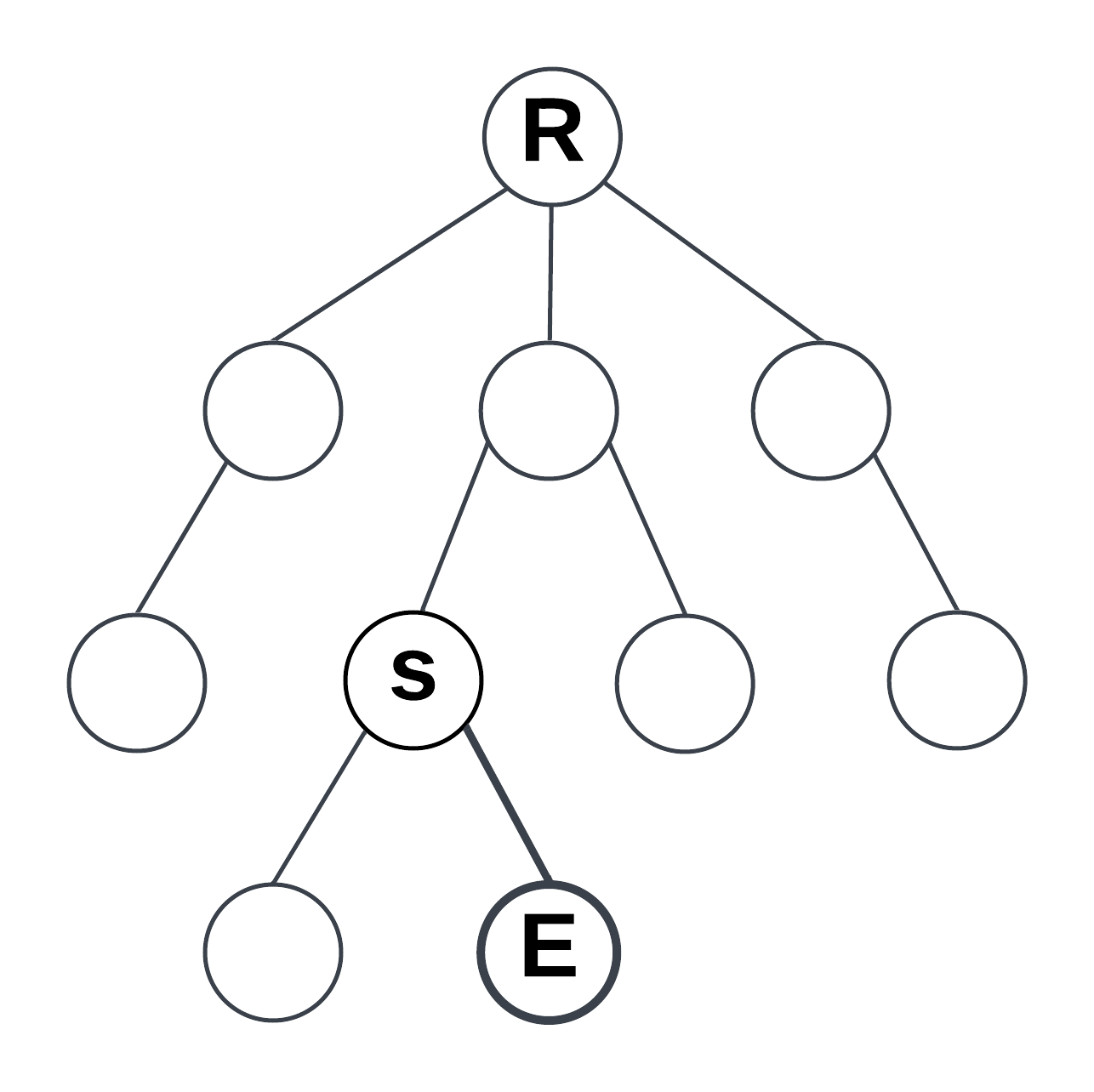
Một node được coi là **có thể mở rộng** khi có một trong các node con của nó **chưa được ghé thăm.** Một node được coi là **chưa được ghé thăm** khi chưa có mô phỏng nào được thực hiện từ node đó hay nói cách khác là giá trị của node đó chưa được tính toán.



Lưu ý rằng node lá (node chưa có con hay chưa được mở rộng thêm các trạng thái) cũng được xem là một node có thể mở rộng và cũng có thể được chọn trong quá trình lựa chọn.

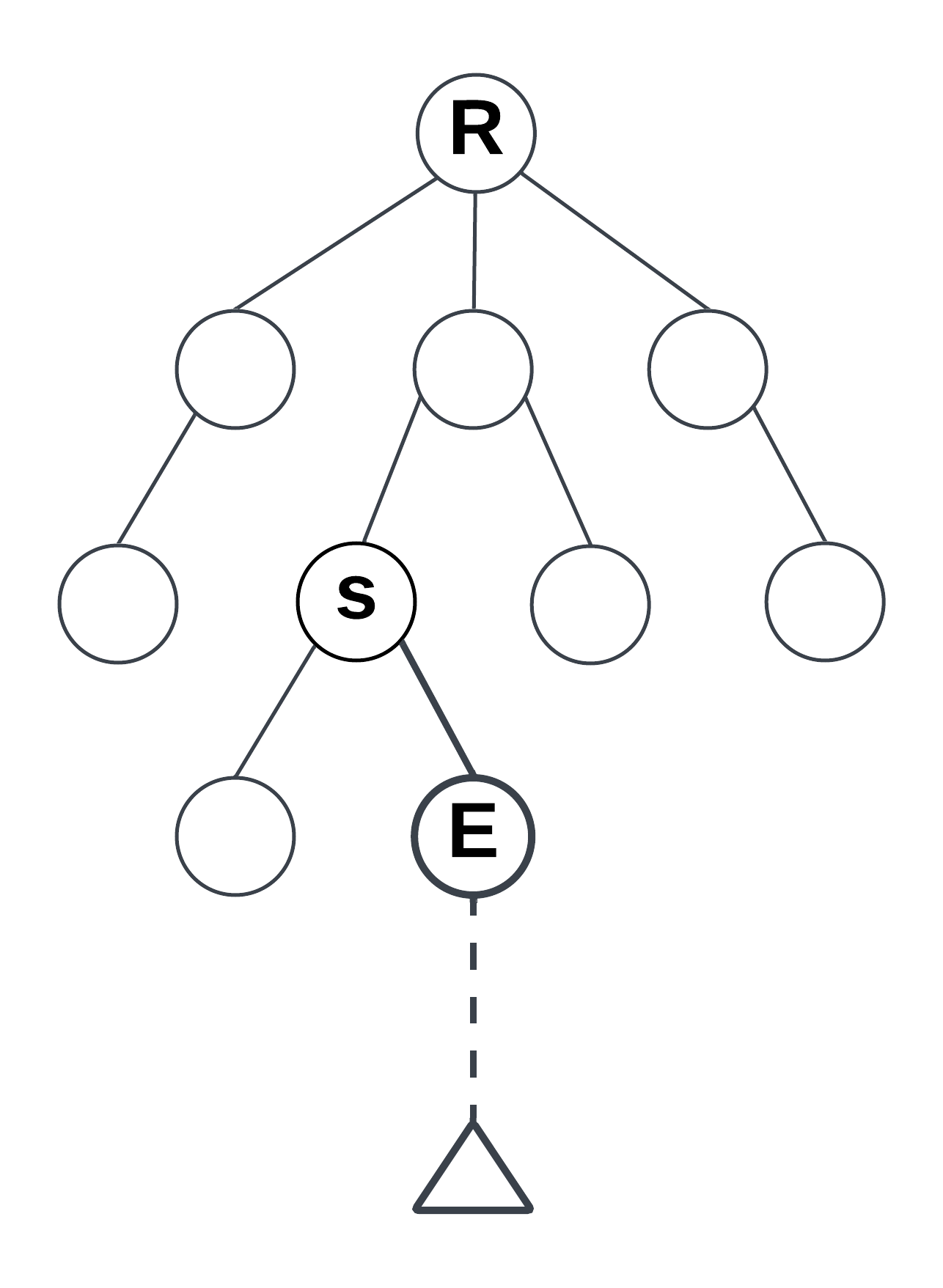
**Mở rộng (Expand)**

Nếu node được chọn ở bước lựa chọn không phải là node biểu diễn trạng thái kết túc thì tiến hành thêm một (hoặc nhiều) node con **E** dựa trên các nước đi hợp lệ từ node S.



**Mô phỏng (simulation hay rollout hoặc playout)**

Từ node **E** (nếu ở bước mở rộng thêm nhiều con thì chọn 1 node bất kì) được thêm vào ở bước mở rộng tiến hành mô phỏng Monte Carlo để ước lượng giá trị cho node đó. Mô phỏng trong ngữ cảnh này nghĩa là thực hiện một loạt các nước đi ngẫu nhiên hợp lệ cho phía bản thân và đối thủ bắt đầu từ node **E** cho đến khi trò chơi đạt trạng thái kết thúc và thu về kết quả **∆.** Trong thực tế quá trình mô phỏng sẽ được thiết kế sao cho có thể hoàn tất trong thời gian ngắn nhất mà vẫn giữ được tính ngẫu nhiên trong các lần lựa chọn. Sau khi hoàn thành mô phỏng từ node **E** thì **E** sẽ trở thành node đã được ghé thăm.



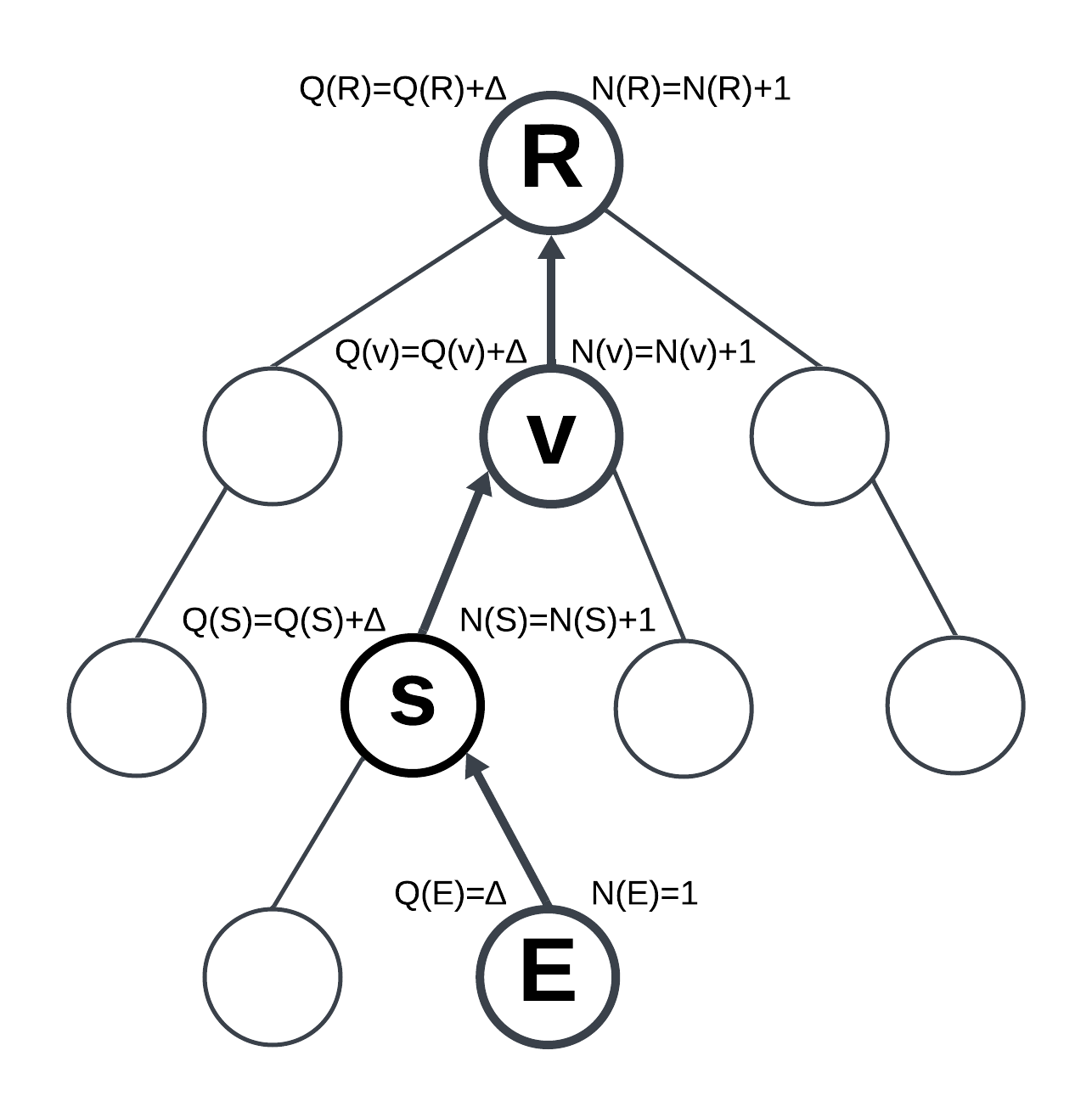
**Lan truyền ngược (Backpropagation)**

Kết quả **∆** của node **E** sau khi được mô phỏng hoặc lấy từ node đạt trạng thái kết thúc trạng thái **T** sẽ được dùng để cập nhật lại số liệu cho các node trên đường đi từ node **E** ngược về node gốc **R**.

Số liệu của một node bao gồm:

* + **Q(v)**:là tổng phần thưởng của của các mô phỏng có đi qua node v
  + **N(v)**: là tổng số lần mà node v được duyệt qua

Với mỗi node **v** nằm trên đường lan truyền ngược từ **E** về node gốc **S**, cộng phần thưởng của **∆** vào **Q(v),** cộng một cho **N(v)**



**UCT (Upper Confidence Bound of Tree or UCB apply to Tree)**

Việc xây dựng cây tìm kiếm phụ thuộc phần lớn vào quá trình lựa chọn. Ta có thể xem việc mỗi lần thực hiện lựa chọn node là một bài toán Mulit Armed Bandit độc lập với giá trị của mỗi hành động là chính giá trị đang được ước tính của một node mang lại thông qua mô phỏng Monte Carlo. Để cân bằng sự “khai thác” và “thăm dò” việc lựa chọn node ta sẽ áp dụng chiến lược UCB1 đã được đề cập ở bài toán MAB. Node S được lựa chọn ở mỗi lần duyệt cây sẽ là node có giá trị lớn nhất được tính theo công thức:

Công thức trên được phát triển từ công thức UCB1 và có thêm hằng số Cp với Cp > 0. Cp có thể dùng để điều chỉnh mức độ “Thăm dò”. Theo lý thuyết Cp = 1/√2 sẽ thõa mãn bất đẳng thức Hoffding với giá trị trong khoảng [0,1] (với 0 là thua và 1 là thắng). Đối với các phần thưởng nằm ngoài khoảng này thì cần điều chỉnh Cp để chiến lược UCT có thể hoạt động tốt hơn.

**Chọn ra nước đi tốt nhất**

Bốn bước trên sẽ được lặp lại để mở rộng cây tìm kiếm cho đến khi đạt đến giới hạn. Giới hạn ở đây có thể là thời gian thực hiện, dung lượng bộ nhớ, ... hoặc đạt trạng thái kết thúc trong quá trình mô phỏng. Sau khi kết thúc việc xây dựng cây tìm kiếm, nước đi có số lần được duyệt nhiều nhất thường sẽ được chọn làm nước đi tốt nhất cho trạng thái gốc **R**.

**Mã giả của thuật toán MCTS**

def tìm\_kiếm\_MCTS(trạng\_thái\_gốc):

while còn\_tài\_nguyên():

node\_được\_chọn = duyệt\_cây(trạng\_thái\_gốc);

node\_mở\_rộng = mở\_rộng(node\_được\_chọn);

kết\_quả\_mô\_phỏng = mô\_phỏng(node\_mở\_rộng);

lan\_truyền\_ngược(node\_mở\_rộng, kết\_quả\_mô\_phỏng);

return tìm\_con\_tốt\_nhất(trạng\_thái\_gốc)

def duyệt\_cây(node):

while có\_thể\_mở\_rộng(node):

node = max\_UCT(node);

return node;

def có\_thể\_mở\_rộng(node):

if not là\_node\_kết\_thúc(node) or len(node.danh\_sách\_node\_con\_chưa\_ghé\_thăm) == 0:

return False;

return True:

def mở\_rộng(node):

node\_mở\_rộng = node.danh\_sách\_node\_con\_chưa\_ghé\_thăm.pop();

node.danh\_sách\_node\_con.append(node\_mở\_rộng);

return node\_mở\_rộng;

def mô\_phỏng(node):

while not là\_node\_kết\_thúc(node):

node = chọn\_ngẫu\_nhiên(node.danh\_sách\_node\_con\_chưa\_ghé\_thăm);

return kết\_quả(node);

def lan\_truyền\_ngược(node, kết\_quả):

while node != None:

node.kết\_quả += kết\_quả;

node.số\_lần\_ghé\_thăm += 1;

node = node.node\_cha

def tìm\_con\_tốt\_nhất(node):

Tìm node con được ghé thăm nhiều nhất (có thể có tìm theo các tiêu chí khác)

**Một vài tính chất quan trọng của thuật toán MCTS**

**Không cần kinh nghiệm (Aheuristic)**

Khác với các thuật toán đưa ra quyết định trước đây chẳng hạn minimax alpha-beta, MCTS chỉ cần luật chơi hay cách thức hoạt động của vẫn đề và không yêu cầu thêm bất cứ kiến thức cũng như kinh nghiệm nào khác về vấn đề nó đang giải quyết. Tuy nhiên thuật toán MCTS vẫn có thể tận dụng các kiến thức và kinh nghiệm về vấn đề cần giải quyết để cải thiện và hoạt động hiệu quả hơn. Đây là một trong các tính chất quan trọng khiến MCTS có thể áp dụng vào rất nhiều lĩnh vực và không chỉ dừng lại ở trò chơi.

**Có thể dừng bất cứ lúc nào**

Ngay khi mỗi vòng lặp kết thúc dữ liệu của toàn bộ cây sẽ được cập nhật ngay lặp tức thông qua quá trình lan truyền ngược. Điều này có nghĩa là thuật toán có thể dừng lại bất cứ khi nào muốn mà vẫn có đầy đủ thông tin để đưa ra được quyết định tiềm năng nhất. Tuy nhiên độ chính xác sẽ tỉ lệ thuân với số lần lặp, nếu dừng quá sớm thì kết quả cho ra có thể không được chính xác.

**Hạn chế của MCTS**

Một điểm yếu của MCTS là với một vị trị nhất định, có thể sẽ có những nước đi trông rất tiềm năng nhưng thực sự nước đi “bẫy” đã đối thủ tính toán trước để dành chiến thằng. MCTS chỉ chọn nước đi tiềm năng nhất trước mắt chứ không phân tích sâu các nước phía sau. Dựa trên tính chất ưu tiên nước đi tiềm năng nhất nước “bẫy” nói trên vẫn sẽ được lựa chọn và đó là tiền đề cho một chuỗi các nước đi dẫn đến thua cuộc đã được đối thủ tính toán trước.

**Các cải tiến cho thuật toán MCTS**

**Cai tien selection**

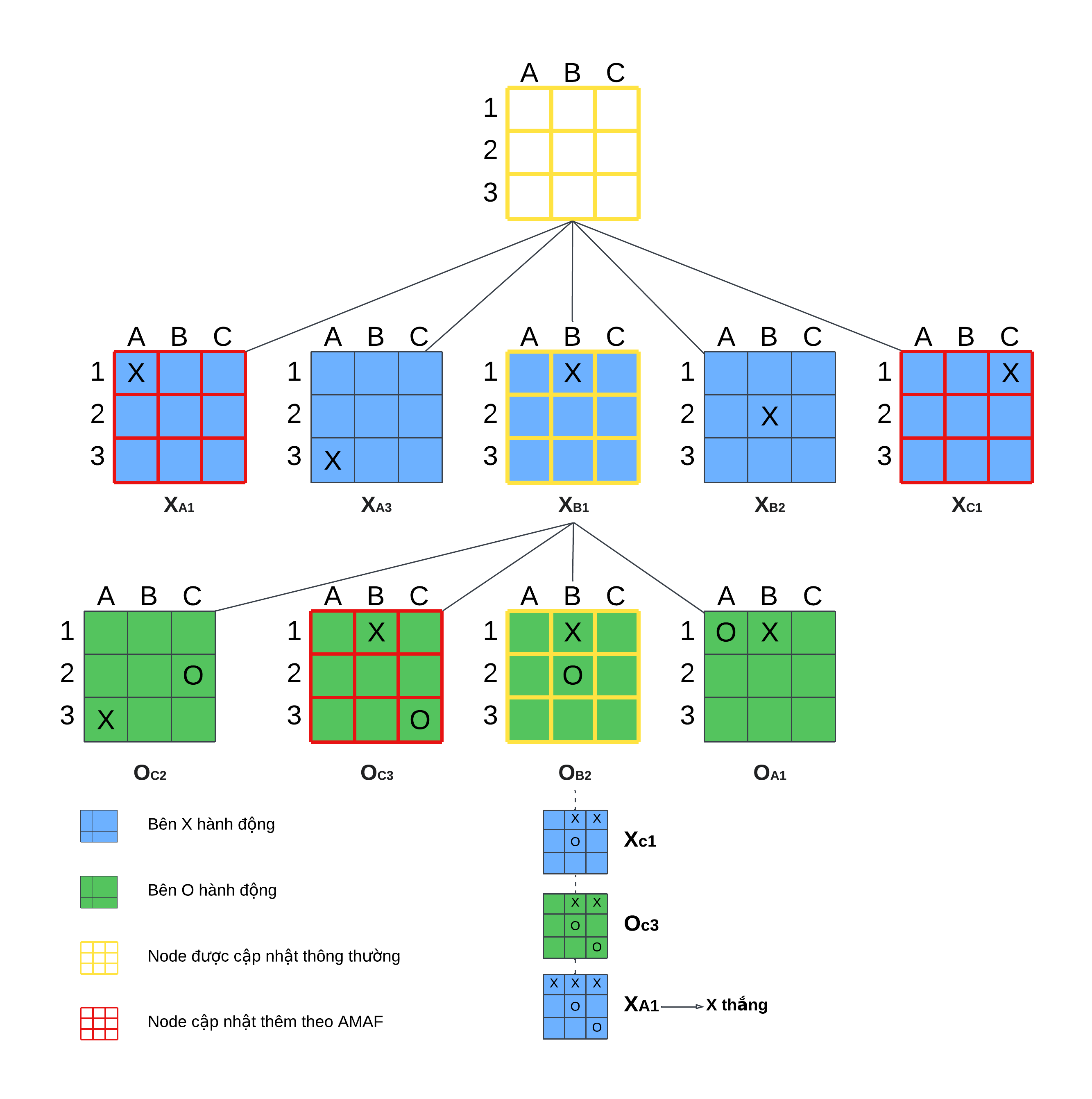
Them may cau linh tinh ve viec tai sao phai cai tien mo phong

**All Move As First (AMAF)**

AMAF là cơ chế dùng để nâng cấp độ chính xác của thuật toán MCTS sử dụng các dữ liệu trong quá khứ.

Trong thuật toán MCTS, giá trị của một node chỉ đáng tin cậy nếu nó đã trãi qua mô phỏng đủ nhiều. Ở các vòng lặp đầu khi chưa có nhiều mô phỏng được thực hiện, việc chọn ra nước đi tiềm năng có thể nói gần như là ngẫu nhiên vì các thông tin của các node tại thời điểm này quá ít. Khi này cần một giải pháp để cung cấp thêm nhiều thông tin hơn cho các node và AMAF ra đời để giải quyết vấn đề trên.

Ý tưởng của AMAF là mọi nước đi được thực hiện trong quá trình mô phỏng đều có thể được dùng để cung cấp thêm thông tin cho các node đang tồn tại trong cây tìm kiếm, nói cách khác AMAF thay đổi cách thức hoạt động trong bước lan truyền ngược của MCTS. Cụ thể, thay vì chỉ cập nhật dữ liệu cho các node trên đường từ node gốc R đến node có thể mở rộng E (được chọn thông qua bước lựa chọn), các node con Ci của các node N nằm trên đường đi từ node R đến E cũng sẽ được cập nhật dữ liệu nếu như hành động A tạo ra trạng thái con Ci từ node N được thực hiện trong quá trình mô phỏng node E.



Khi áp dụng AMAF, mỗi node sẽ có thêm 2 thông tin là là QAMAF và NAMAF. QAMAF là tổng phần thưởng nhận được thông qua quá trình cập nhật theo AMAF và NAMAF là tổng số lần được cập nhật theo AMAF hay có thể hiểu NAMAF là số lần mà hành động đó xuất hiện trong những lần mô phỏng. Khi này mỗi node sẽ có tổng 4 thông tin là QUCT, NUCT cho quá trình cập nhật thông thường và QAMAF, NAMAF cho qua trình cập nhật theo AMAF.

**RAVE (Rapid Action Value Estimates)**

Sau khi có được thêm các thông tin từ quá trình cập nhật theo AMAF, các node được chọn trong bước lựa chọn phải đạt giá trị cao nhất được tính theo công thức RAVE:

Giá trị RAVE khi này sẽ được dùng để quyết định node được lựa chọn thay vì UCT. Giá trị RAVE được tính toán bằng cách kết hợp giá trị UCT và giá trị AMAF. Giá trị UCT được tính toán theo công thức UCT. Giá trị AMAF được tính như sau:

Theo công thức RAVE, một node có được lựa chọn hay không phụ thuộc vào giá trị UCT và giá trị AMAF. Số α ở đây thể hiện mức độ ảnh hưởng của hai giá trị UCT và AMAF lên giá trị RAVE hay nói cách khác là mức độ ảnh hưởng của hai giá trị UCT và AMAF lên quyết định liệu node có được lựa chọn hay không. α được tính như sau:

Theo công thức trên giá trị của α sẽ nằm trong khoảng từ 0 đến 1. CAMAF cho biết số lần mà một node phải được ghé thăm theo chiến lược UCT (không tính các lần cập nhật AMAF) cho đến khi giá trị AMAF không còn ảnh hưởng đến giá trị cuối cùng của node đó. Giá trị CAMAF thường nằm trong khoảng từ 200 đến 700 và có thể được tinh chỉnh tùy theo yêu cầu bài toán.

Giả sử cho CAMAF = 200 tức một node trước khi được lựa chọn lần thứ 200 thì giá trị của nó sẽ bị ảnh hưởng bởi giá trị AMAF. Tiếp tục giả sử như node hiện tại đã được ghé thăm 20 lần thì giá trị α lúc này sẽ là:

Theo công thức RAVE với α = 0.9,giá trị AMAF chiếm 90% giá trị của node, giá trị UCT sẽ chiếm 10% giá trị còn lại của node. Có thể thấy nếu như số lần được ghé thăm theo chiến lược UCT của một node tăng lên thì giá trị α càng gần về 0, giá trị α càng giảm tức mức độ ảnh hưởng của giá trị AMAF lên giá trị của node càng giảm, đến khi bằng α = 0 thì giá trị của một node hoàn toàn tính theo giá trị UCT đồng nghĩa với node đã được ghé thăm đủ nhiều và giá trị của nó đã đáng tin cậy, không còn cần thêm thông tin từ quá trình cập nhật AMAF.

Cải tiến quá trinh mô phỏng

Them may cau linh tinh ve viec tai sao phai cai tien mo phong

Decisive move

Tại một trạng thái trò chơi có thể sẽ có các nước đi dẫn đến chiến thằng ngay lập tức. Để rút ngắn thời gian mô phỏng, thay vì chọn ngẫu nhiên hành động, trước tiên ta kiểm tra xem ở trạng thái hiện tại có hành động nào dẫn đến chiến thắng ngay lập tức không, nếu có thực hiện hành động đó để kết thúc nhanh quá trình mô phỏng, nếu không có thì chọn ngẫu nhiên một hành động như mô phỏng thông thường

Last Good Reply

**Áp dụng MCTS vào thỏa thuận mức độ dịch vụ (SLA)**

**Service Level Agreement (SLA)**

SLA là một thỏa thuận chính thức, có ràng buộc pháp lý giữa nhà cung cấp dịch vụ và khách hàng. Mục tiêu của SLA là thiết lập kỳ vọng rõ ràng, đảm bảo tính minh bạch và trách nhiệm giải trình, cung cấp cơ sở pháp lý để giải quyết tranh chấp giữa hai bên. SLA bao gồm nhiều yếu tố như số lượng hàng hóa mong muốn, chất lượng, trách nhiệm của nhà cung cấp, thời gian phản hồi…đã được 2 bên thỏa thuận.

Môt số thành phần chính cần quan tâm của một SLA bao gồm:

* Nhu cầu của khách hàng: số lượng hàng mà khách mong muốn
* Thời gian xem xét hiệu suất (Performance Review Period) (PRP): là khoảng thời gian mà hiệu suất (tỉ lệ lập đầy) của nhà cung cấp được đo lường và đánh giá với các mục tiêu dã thỏa thuận. Ví dụ PRP là 5 ngày thì hiệu suất sẽ được tính toán tại ngày thứ 5, 10, 15
* Tỉ lệ lấp đầy (fill rate): là thước đo hiệu suất cho biết có bao nhiêu phần trăm đơn hàng của khách hàng được đáp ứng ngay lập tức với lượng hàng hiện có trong kho
  + Target fill rate: là hiệu suất mà khách hàng yêu cầu phải đạt ở mỗi lần giao hàng
  + Real fill rate: là hiệu suất thực tế mà nhà cung cấp đáp ứng được
* Các biện pháp khắc phục: là các biện pháp mà nhà cung cấp sẽ thực hiện nếu như không đáp ứng đúng các mục tiêu đã thỏa thuận

Trong đó thời gian xem xét hiệu suất và tỉ lệ lấp đầy ảnh hưởng trực tiếp đến hiệu suất tổng thể của hợp đồng.

Việc đề ra một chính sách phân bố hàng hóa hợp lí là điểm mấu chốt để quản lí hiệu quả SLA. Chính sách phân bố sẽ chỉ ra các quy luật cần tuân theo và hướng dẫn phân bố hàng hóa dựa trên nhu cầu của khách hàng đảm bảo đạt được hiệu suất đã thỏa thuận. Việc có một chính sách phân bố hợp lí sẽ cân bằng được chi phí hàng tồn kho và sự hài lòng của khách, giảm thiểu các mức phạt phải nhận đồng thời giúp nhà cung cấp phân bố hàng hóa linh hoạt với sự thay đổi nhu cầu của khách hàng trong thực tế.

**Hạn chế của các chính sách phân bố hàng hóa không có chiến lược cụ thể**

Đối với các chính sách phân bố không có chiến lược cụ thể, hàng hóa phân bổ cho các khách hàng được định trước và không thay đổi theo thời gian. Chính sách phân bố này chỉ tập trung vào nhu cầu hiện tại, ưu tiên lợi nhuận trong ngắn hạn mà bỏ qua các yếu tố khách có thể ảnh hưởng đến lợi nhuận trong tương lại.

Giả sử như bởi sự hạn chế về thời gian, chất lượng, chi phí của việc sản xuất cũng như vẫn chuyển hàng hóa, kho hàng của một nhà cung cấp mất 3 tháng để được lấp đầy lại. Khi này họ sẽ phải lên kế hoạch phân bố hàng để sao cho trong 3 tháng chờ hàng về phải phân bố được hết hàng hóa tránh tính trạng tồn kho và đồng thời phải đáp ứng tốt nhu cầu của khách hàng ở mỗi lần giao hàng. Đối với các chính sách phân bố hàng hóa “tĩnh”, số lượng hàng hóa giao cho khách hàng là như nhau ở mỗi lần giao hàng dẫn đến có thể sẽ bị thiếu hụt hàng hóa trong tương lai. Ngoài ra khi nhu cầu thay đổi đột ngột, các chính sách phân bố này cấn thời gian để tính toán lại từ đó làm mất sự ổn định của tỉ lệ lấp đầy giữa các chu kì xem xét hiệu suất dẫn đến giảm lợi nhuận của hà cung cấp.

Khi áp dụng thuật toán MCTS, chính sách phân bố sẽ cân nhắc tinh chỉnh số lượng hàng giao cho các khách hàng khác nhau ở mỗi lần giao hàng tùy thuộc vào tình hình hiện tại của kho hàng và lợi nhuận của từng khách hàng. Thêm vào đó MCTS có thể phản ứng nhanh với các sự thay đổi đột ngột từ phía khách hàng. Các đặc điểm trên đảm bảo độ ổn định của tỉ lệ lấp đầy giữa các chu kì xem xét hiệu suất và tối ưu lợi nhuận cho nhà sản xuất.

**Mô hình hóa vấn đề SLA**

**Thời điểm đưa ra quyết định**

Gọi H là khoảng thời gian mà nhà cung cấp sẽ lên kế hoạch phân bố ta có T = {1, 2, 3, …, H} là tập chứa các thời điểm t phải đưa ra quyết định trong khoảng thời gian H. t ở đây có thể linh hoạt không nhấn thiết là ngày, tháng, … miễn là các t cách nhau một đơn vị thời gian bằng nhau.

**Các kí hiệu**

R: tập hợp chứa các nhà bán lẻ

S: lượng hàng hiện có trong kho của nhà cung cấp

drt: nhu cầu của nhà bán lẻ r tại thời điểm t

art: lượng hàng giao cho nhà bán lẻ r tại thời điểm t

ps: lợi nhuận của nhà cung cấp trên một sản phẩm

πt: lợi nhuận của nhà cung cấp tại thời điểm t

: tỉ lệ lấp đầy của nhà bán lẻ r mong muốn

: tỉ lệ lấp đầy của nhà bán lẻ r được tính tại thời điểm kết thúc chu kì xem xét hiệu suất, bắt đầu từ thời điểm t0 đến thời điểm t0 + τ - 1

τ: chu kì xem xét hiệu suất

λt: chi phí phạt cho mỗi đơn vị mà tỉ lệ lấp đầy thực tế lệch khỏi mong muốn của nhà bán lẻ

Ѱ: tập hợp các trạng thái. ѱ là các state (node) trong cây tìm kiếm

**Các thành phần trong cây tìm kiếm**

*State: ѱt = (S, dt)*

Mỗi state chứa tất cả thông tin cần biết của nh à cung cấp tại thời điểm t. Trong trường hợp này state chỉ cần biết sức chứa kho hàng của nhà cung cấp (S) và nhu cầu của các nhà bán lẻ tại thời điểm t được biểu diễn bằng vector dt với các phần tử là nhu cầu của mỗi nhà bán lẻ drt (dt = (drt) r ϵ R, drt ϵ Z)

*Action:*

Mỗi action là một cách phân bố hàng hóa đến cho tất cả các nhà bán lẻ tại thời điểm t được biểu diễn bằng vector at với các phần tử của vector là lượng hàng phân bố cho mỗi nhà bán lẻ art. Mỗi action phải tuân thủ 2 điều kiện

* + Tổng số hàng phân bổ phải nhỏ hơn hoặc bằng với số hàng hiện có trong kho
  + Số lượng hàng cung cấp cho một nhà bán lẻ phải nhỏ hơn hoặc bằng nhu cầu của nhà bán lẻ đó ⎣

*Reward:*

Reward ở đây là lợi nhuận của nhà cung cấp tại thời điểm một thời điểm t. Cách tính lợi nhuận sẽ chia làm 2 trường hợp:

* + Thời điểm t chưa phải là ngày hết hạn của chu kì xem xét hiệu suất, khi này lợi nhuận tính bằng lợi nhuận trên 1 sản phẩm (ps) nhân với tổng số lượng hàng phân bố tại thời điểm t
  + Thời điểm t là ngày hết hạn chu kì xem xét hiệu suất, khi này lợi nhuận sẽ tính bằng giá trị của trường hợp trên trừ cho tổng số tiền phạt. Tổng tiền phạt được tính bằng số tiền phạt nhân với lượng chênh lệch giữa tỉ lệ lấp đầy mà nhà bán lẻ r mong muốn với tỉ lệ lập đầy thực tế tại thời điểm t của nhà bán lẻ r . Lưu ý [x]+ = max(x, 0) tức chênh lệch tỉ lệ lấp đầy chỉ được tính khi tỉ lệ lấp đầy tại cuối PRP nhỏ hơn tỉ lệ lấp đầy mong muốn của nhà bán lẻ (). Trong trường hợp tỉ lệ lấp đầy ở cuối PRP lớn hơn nhu cầu của nhà bán lẻ mức chênh lệch sẽ bằng 0 (thì chênh lệch là số âm dẫn đến []+ = max(, 0 ) = 0). Tỉ lệ lấp đầy tại thời điểm kết thúc PRP tính bằng cách chia tổng lượng hàng hóa trong chu kì PRP cho tổng lượng hàng mong muốn của các nhà bán lẻ trong chu kì PRP

Từ một state ở thời điểm ta thực hiện một action để tạo ra state con của state đó tại thời điểm t + 1. Mỗi cấp t của cây đại diện cho trạng thái của nhà cung cấp tại thời điểm t trong khoảng thời gian lên kế hoạch H.

**Các cải tiến của thuật toán MCTS khi áp dụng vào SLA**

*Loại bỏ các phân bố không hợp lệ*

Giả sử như chỉ có 2 nhà bán lẻ. Khi này, tập các phân bố hợp lệ Fѱ sẽ chứa các phân bố cho các nhà bán lẻ sao cho lượng hàng phân bố cho mỗi nhà bán lẻ sẽ không lớn hơn nhu cầu của họ.

Tại mỗi State sẽ có thể xảy 2 trường hợp

Trường hợp 1 (c1):

Khi tổng nhu cầu của các nhà bán lẻ cao hơn hoặc bằng số hàng có trong kho thì tổng lượng hàng hóa đang có phải được phân phối hết, tất cả các phân bố có tổng lượng hàng phân bố nhỏ hơn lượng hàng hiện có đều dẫn đến việc giảm lợi nhuận cho nhà cung cấp

Trường hợp 2 (c2):

Khi tổng số hàng trong kho lớn hơn tổng nhu cầu của các nhà bán lẻ thì các lượng hàng phân bố cho mỗi nhà bán lẻ phải đúng bằng nhu cầu của họ

*Cải tiến quá trình lựa chọn trong thuật toán MCTS*

Ở cải tiến này ta sẽ áp dụng các số liệu trong SLA vào công thức UCT để điều chỉnh quá trình lựa chọn trong thuật toán. Ta có là khoảng cách giữa vector phân bố và vector nhu cầu của nhà bán lẻ hay dễ hiệu là sự chênh lệch của tổng hàng phân bố với tổng nhu cầu của nhà bán lẻ tại mỗi node. Nếu chênh lệch tại node lớn thì tức node đó không tối ưu lợi nhuận nên ta sẽ trừ UCT cho phần chênh lệch để điều chỉnh quá trình lựa chọn của thuật toán. Công thức mới gọi là AUCT (Allocation distance UCT)

Ngoài ta kết hợp thêm tham số Cd có thể điều chỉnh mức độ ảnh hưởng của khoảng cách phân bố lên quá trình lựa chọn