Apply Novelty Search And Evolution Strategy To Imporve Exploration For Deep Reinforcement Learning

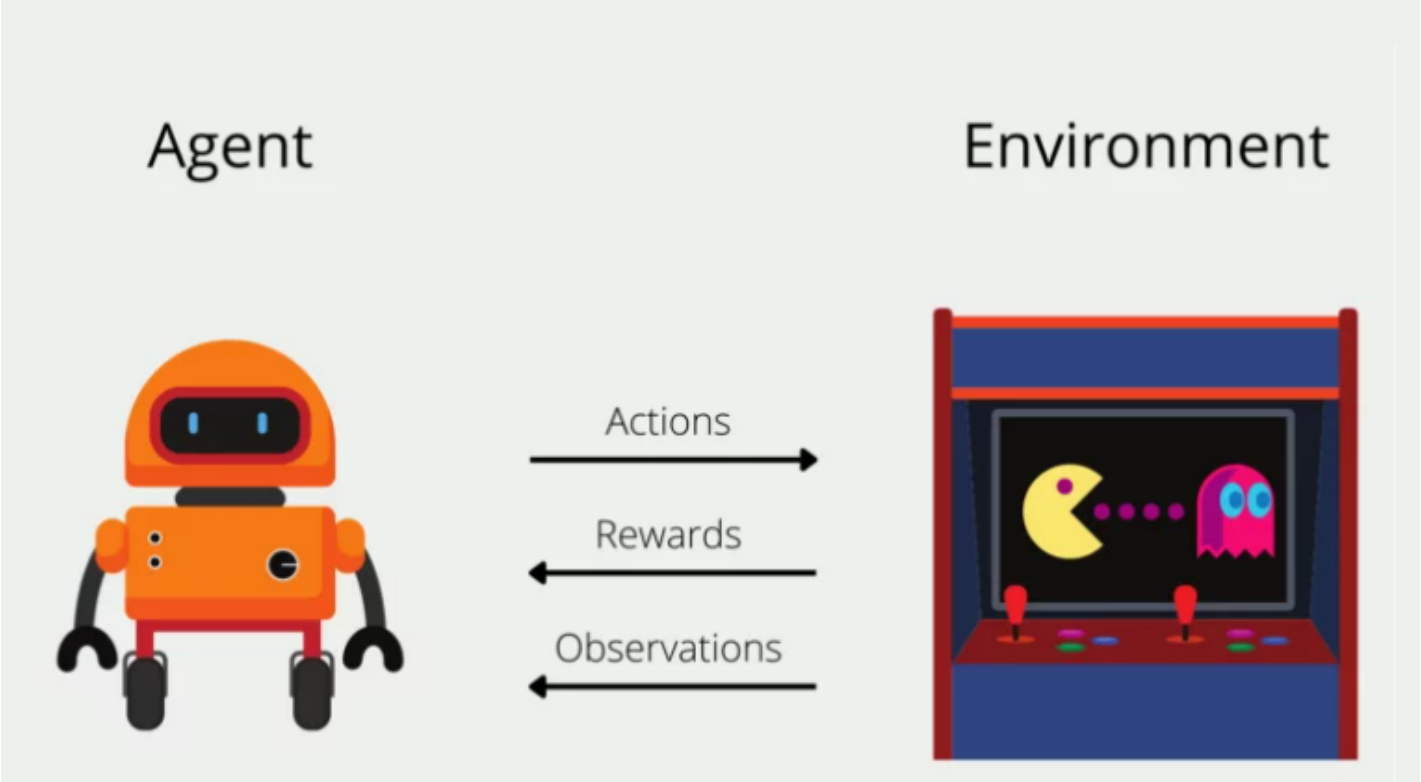
Võ Đức Dương duong922003@gmail.com

Ngày 16 tháng 4 năm 2025

Mục lục

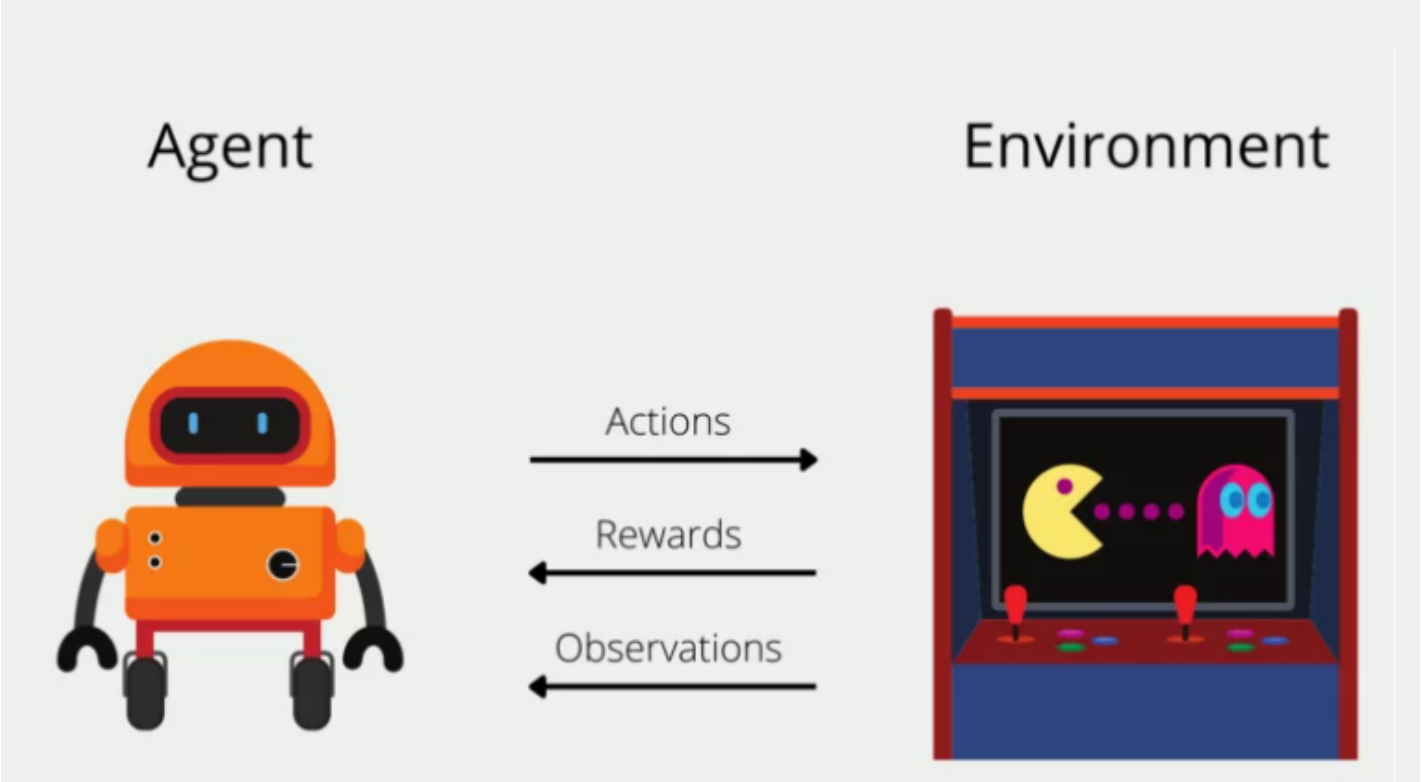
Mục lục

Reinforcement Learning



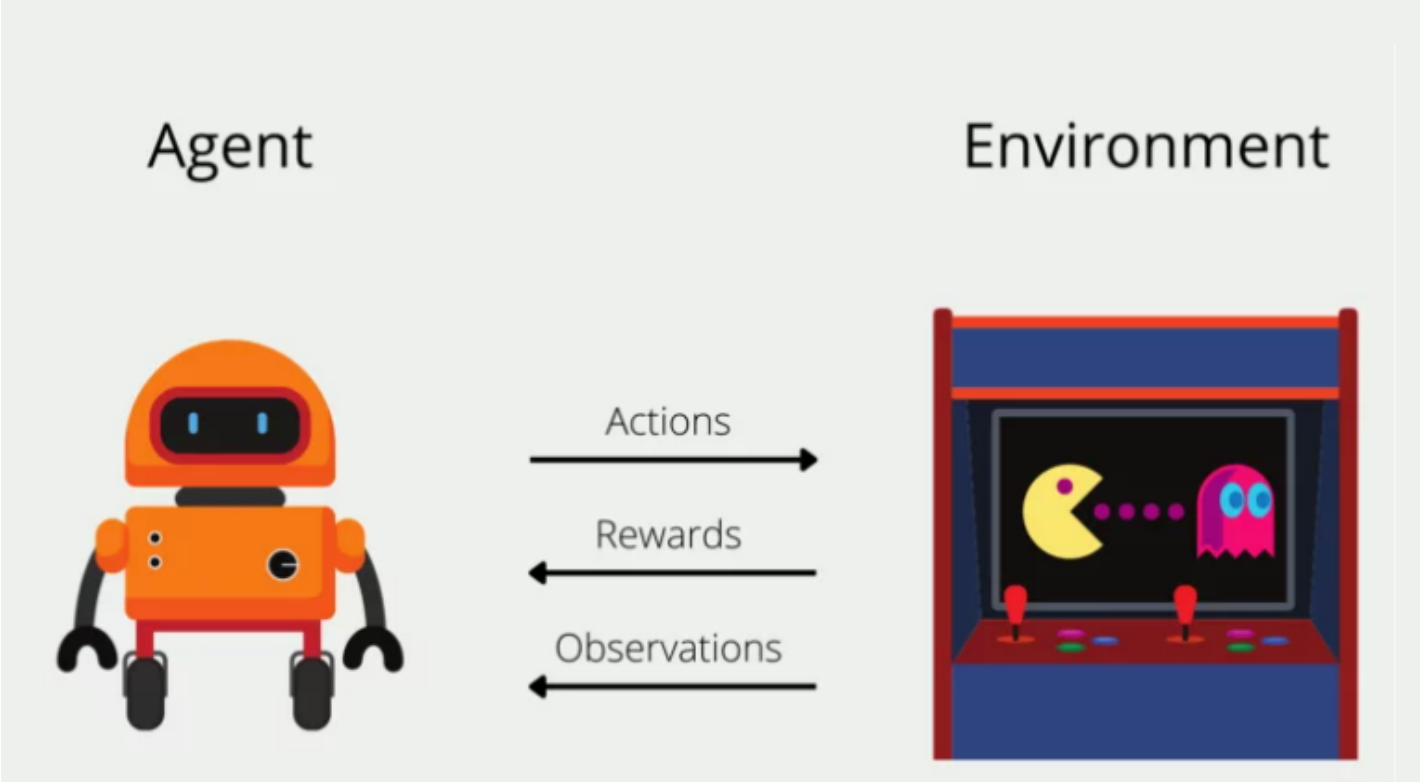
- Agent: tối ưu hóa điểm thưởng

Reinforcement Learning



- Agent: tối ưu hóa điểm thưởng
- **Vấn đề:** hàm (tín hiệu) phần thưởng có thể gây nhầm lẫn hoặc khan hiếm. Khiến agent kẹt ở điểm tối ưu cục bộ

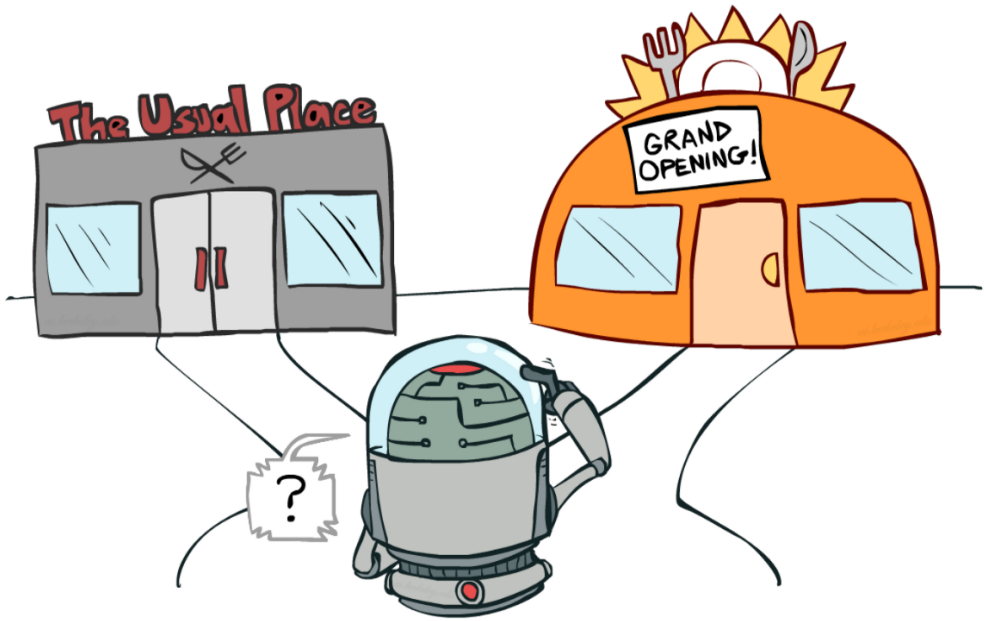
Reinforcement Learning



- Agent: tối ưu hóa điểm thưởng
- **Vấn đề:** hàm (tín hiệu) phần thưởng có thể gây nhầm lẫn hoặc khan hiếm. Khiến agent kẹt ở điểm tối ưu cục bộ

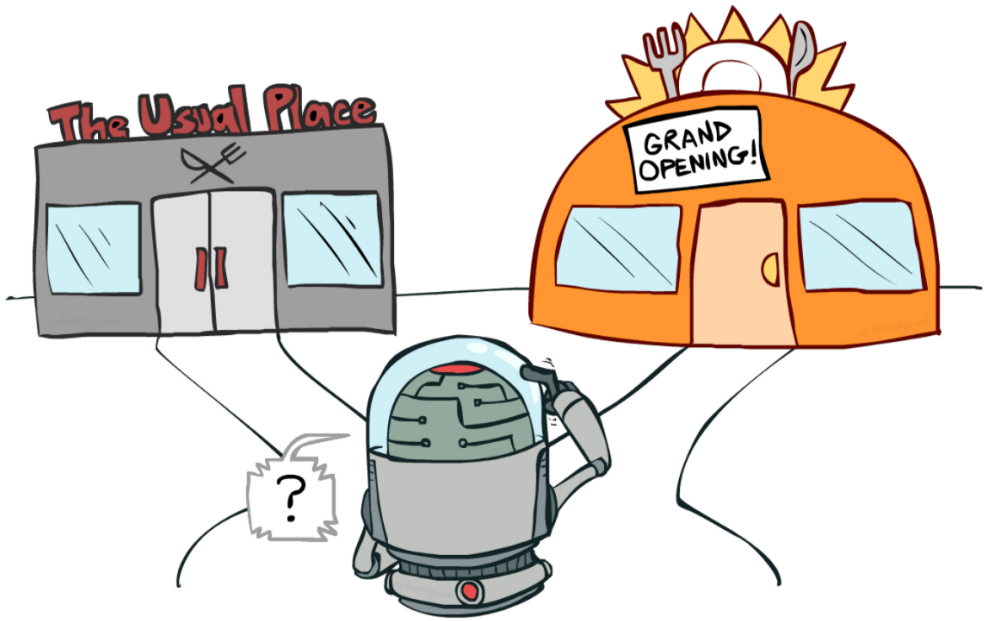
  $\rightarrow$ Cần chiến lược khám phá thông minh hơn

Exploration in RL



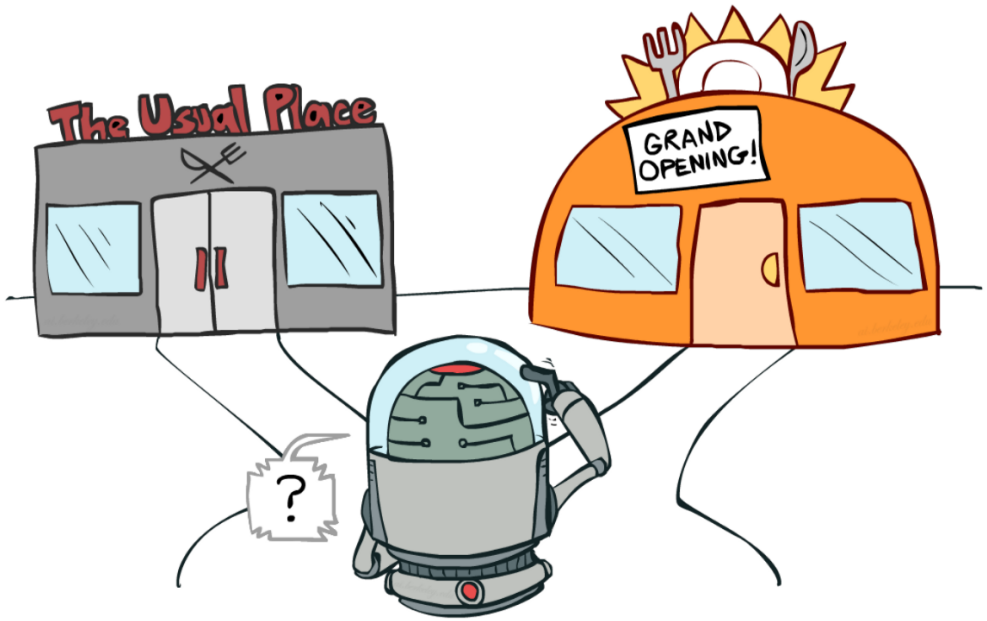
- Giải pháp: thúc đẩy agent đến các trạng thái hiếm hoặc chưa được khám phá $\rightarrow$ khuyến khích khám phá (exploration)

Exploration in RL



- Giải pháp: thúc đẩy agent đến các trạng thái hiếm hoặc chưa được khám phá $\rightarrow$ khuyến khích khám phá (exploration)
- **Ví dụ**: Kĩ thuật đếm tần suất thăm trạng thái dựa trên mã hóa trạng thái hoặc mô hình mật độ

Exploration in RL



- Giải pháp: thúc đẩy agent đến các trạng thái hiếm hoặc chưa được khám phá $\rightarrow$ khuyến khích khám phá (exploration)
- **Ví dụ**: Kĩ thuật đếm tần suất thăm trạng thái dựa trên mã hóa trạng thái hoặc mô hình mật độ
- **Vấn đề:** Các trạng thái được đếm một cách độc lập với nhau.

NS & QD for DRL

- **Hướng tiếp cận khác**: Thiết kế hoặc học một mô tả về hành vi trong suốt lifetime của agent.

NS & QD for DRL

- **Hướng tiếp cận khác**: Thiết kế hoặc học một mô tả về hành vi trong suốt lifetime của agent.
- **Mục tiêu**: Khuyến khích agent thực hiện hành vi khác với những hành vi trước đó. Ví dụ *novelty search (NS)* và *quality diversity (QD)*.
- **Đặc điểm**: Khám phá bằng một quần thể các agent.

NS & QD for DRL

- **Hướng tiếp cận khác**: Thiết kế hoặc học một mô tả về hành vi trong suốt lifetime của agent.
- **Mục tiêu**: Khuyến khích agent thực hiện hành vi khác với những hành vi trước đó. Ví dụ *novelty search (NS)* và *quality diversity (QD)*.
- **Đặc điểm**: Khám phá bằng một quần thể các agent.

Nội dung đề tài: Sử dụng ý tưởng của NS và QD trong các thuật toán tiến hóa để giải quyết các bài toán Deep Reinforcement Learning (DRL).

Mục lục

DQN

- DQN (Deep Q-Network) là thuật toán được mở rộng từ thuật toán Q-Learning, sử dụng một mạng nơ-ron để ước lượng Q-value thay vì sử dụng Q-table như Q-Learning truyền thống.

Q-Learning

- Hàm $Q(s, a)$ đại diện cho giá trị kỳ vọng nhận được khi thực hiện hành động a ở trạng thái s và tuân theo chính sách π.
- Cập nhật Q-value thông qua phương trình Bellman:

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[r + \gamma \max_{a'} Q(s',a') - Q(s,a) \right]$$

DQN

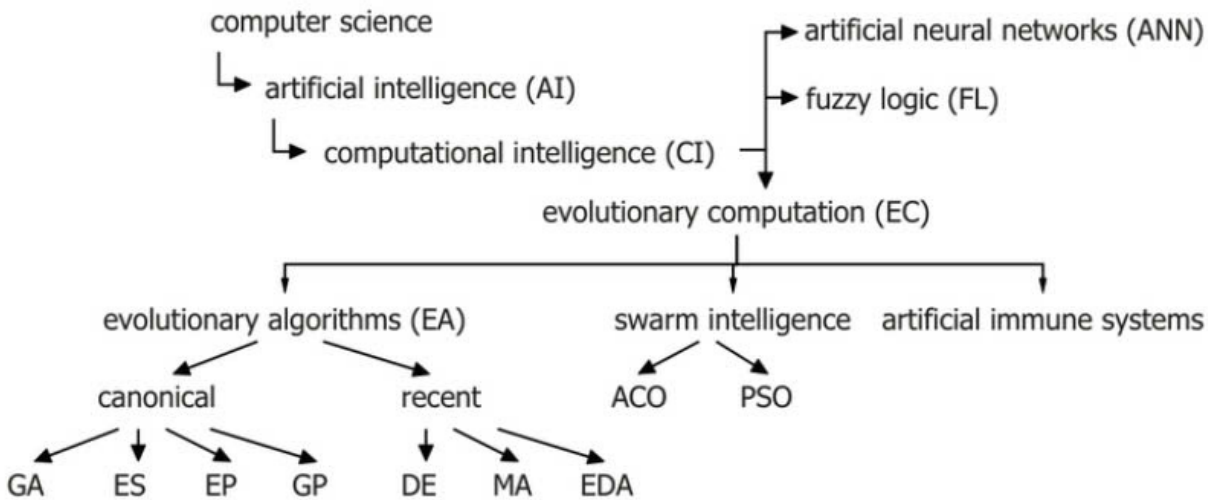
- DQN thay thế Q-table bằng một mạng nơ-ron $Q_\theta(s, a)$ có tham số θ. Phương trình Bellman được ước lượng thông qua tối ưu hóa hàm mất mát:

$$\mathcal{L}(\theta) = \mathbb{E}_{(s,a,r,s')\sim\mathcal{D}}\left[\left(r + \gamma \max_{a'} Q_{\theta^-}(s', a') - Q_\theta(s, a)\right)^2\right]$$

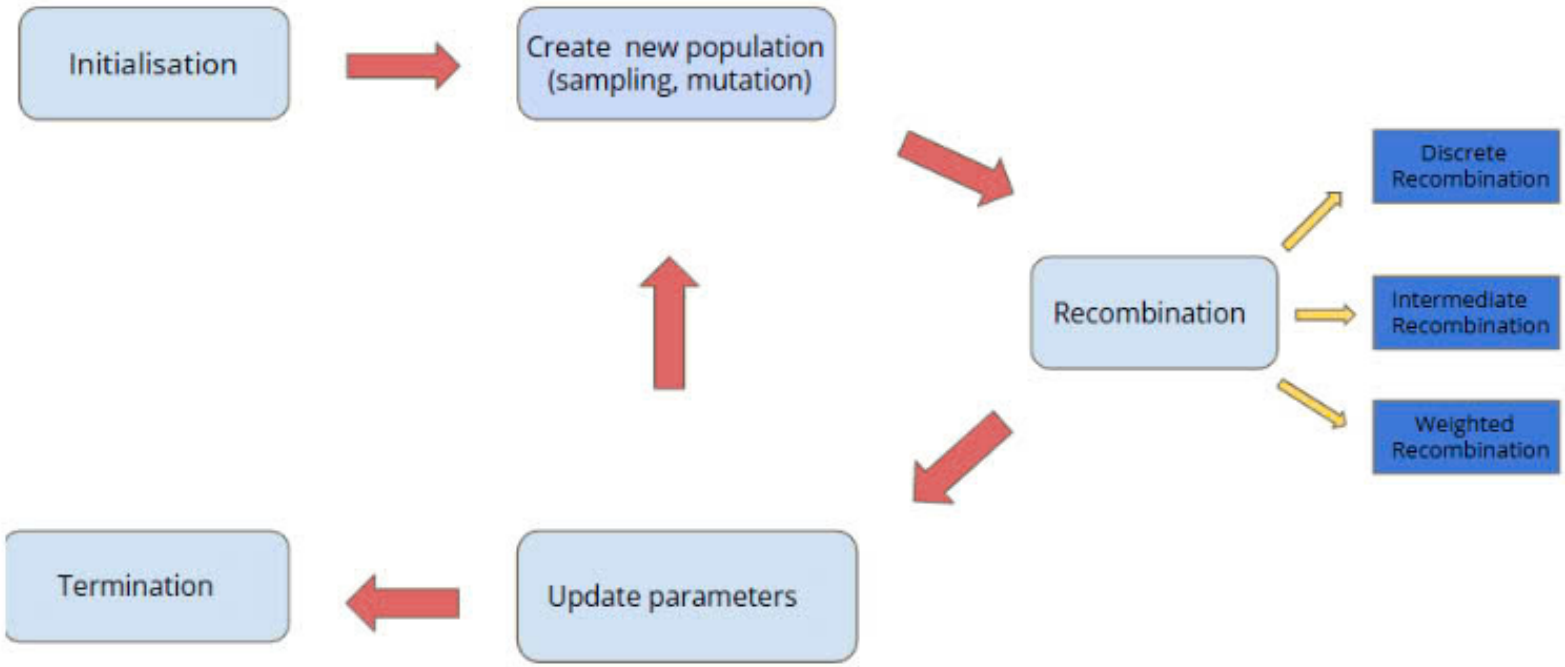
- Q_θ: Mạng chính (online network).
- Q_{θ^-}: Mạng mục tiêu (target network), cập nhật định kỳ từ mạng chính.
- $\mathcal{D}$: Bộ nhớ kinh nghiệm (Replay Buffer) lưu trữ các trạng thái, hành động, phần thưởng và trạng thái tiếp theo để huấn luyện.

Evolution Strategies (ES)

- Evolution Strategy (ES) là một thuật toán thuộc nhóm thuật toán tiến hóa (EAs)
- ES thường sử dụng cho các biểu diễn số thực, với phép biến đổi (variation) chủ yếu là đột biến (mutation)



Evolution Strategies (ES)



Hình: Ý tưởng cơ bản của Evolution Strategies

Novelty Search (NS)

- NS được lấy cảm hứng từ xu hướng tự nhiên hướng tới sự đa dạng.
- NS trong thuật toán tiến hóa là một phương pháp được sử dụng để tìm kiếm các giải pháp mới mẻ
- NS khuyến khích sự đa dạng trong hành vi, ưu tiên những giải pháp khác biệt so với các giải pháp trước đó thay vì tập trung vào một mục tiêu cố định.

NS - working ideas

- Behavior Characterization
  - Mỗi chính sách π được gán một đặc trưng hành vi (BC) là $b(\pi)$, mô tả cách nó hoạt động trong môi trường

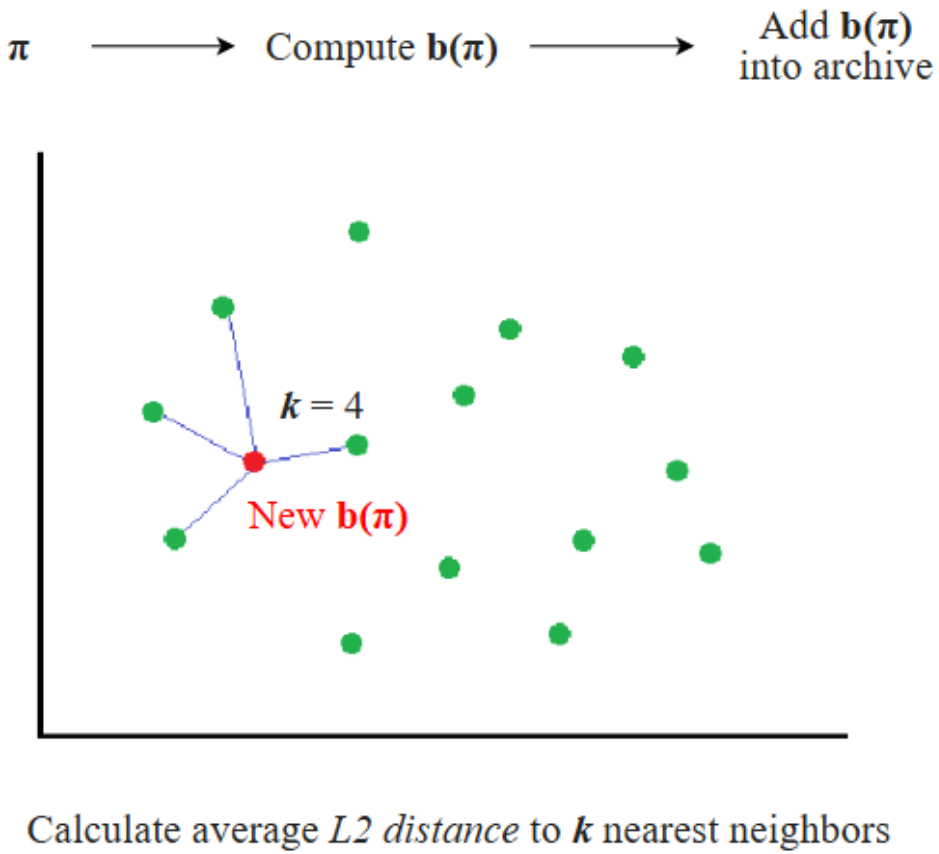
NS - working ideas

- Behavior Characterization
  - Mỗi chính sách π được gán một đặc trưng hành vi (BC) là $b(\pi)$, mô tả cách nó hoạt động trong môi trường
- Behavior Archive
  - Trong quá trình huấn luyện, mỗi một BC được thêm vào archive A

NS - working ideas

- Behavior Characterization
  - Mỗi chính sách π được gán một đặc trưng hành vi (BC) là $b(\pi)$, mô tả cách nó hoạt động trong môi trường
- Behavior Archive
  - Trong quá trình huấn luyện, mỗi một BC được thêm vào archive A
- Tính toán mức độ mới lạ
  - Novelty score của một một chính sách π_θ được đo bằng khoảng cách giữa $b(\pi_\theta)$ và các BC trước đó trong archive.
  - Chọn k hàng xóm gần nhất của $b(\pi)$ trong Archive A và tính trung bình khoảng cách L2 giữa $b(\pi_\theta)$ và các hàng xóm gần nhất

NS - working ideas



Hình: Minh họa cách thuật toán NS hoạt động

Quality Diversity - QD

- QD trong thuật toán tiến hóa là một phương pháp để tìm kiếm một tập hợp đa dạng các giải pháp có chất lượng cao thay vì chỉ tập trung vào một giải pháp tối ưu duy nhất.

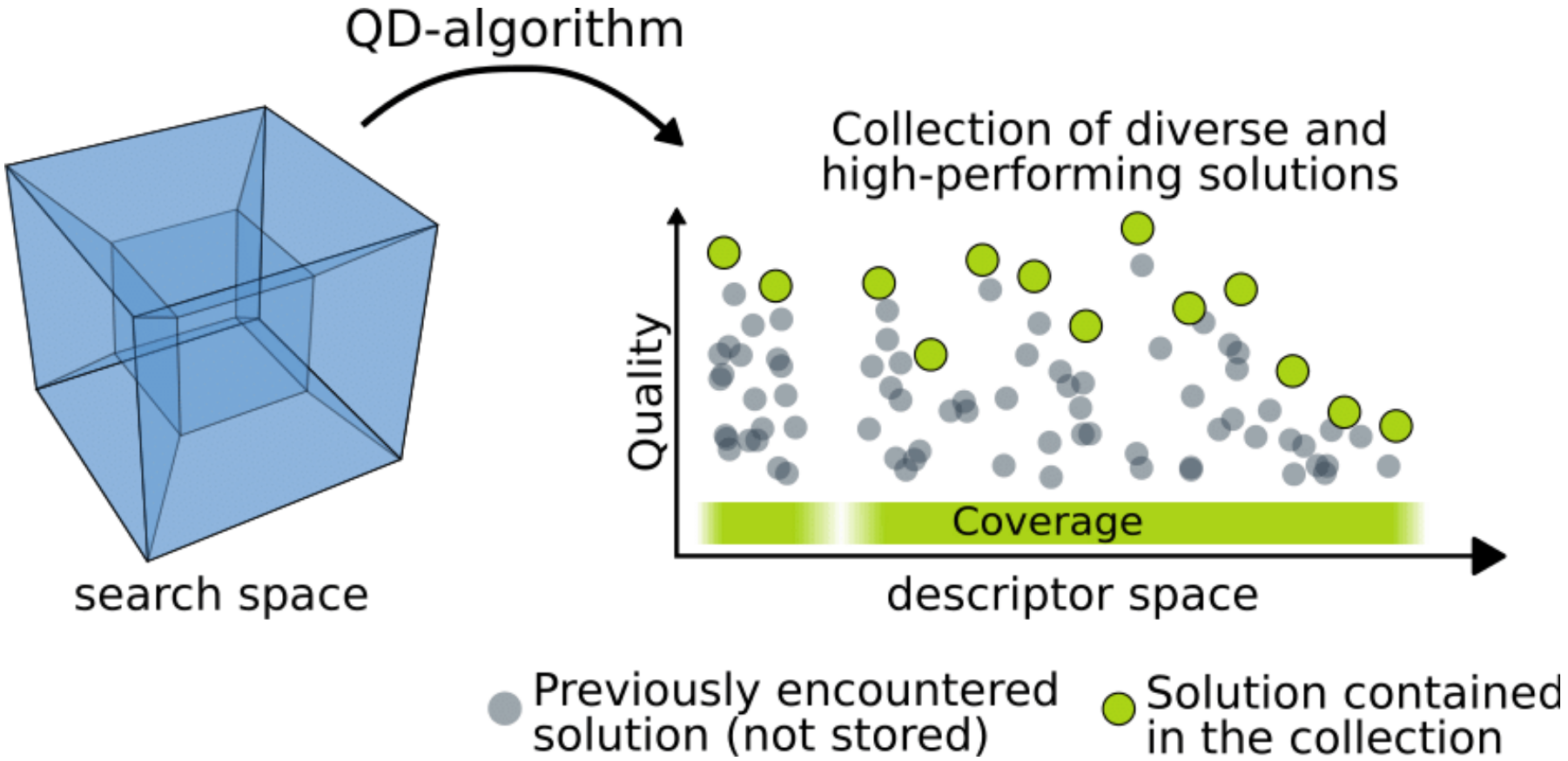
Quality Diversity - QD

- QD trong thuật toán tiến hóa là một phương pháp để tìm kiếm một tập hợp đa dạng các giải pháp có chất lượng cao thay vì chỉ tập trung vào một giải pháp tối ưu duy nhất.
- Ý tưởng chính
  - **Đa dạng**: Chia không gian tìm kiếm thành các khu vực khác nhau, mỗi khu vực đại diện cho một dạng hành vi hoặc đặc điểm cụ thể của giải pháp
  - **Chất lượng cao**: Mỗi cá thể cần phải đạt được chất lượng cao theo một hàm mục tiêu

Quality Diversity - QD

- QD trong thuật toán tiến hóa là một phương pháp để tìm kiếm một tập hợp đa dạng các giải pháp có chất lượng cao thay vì chỉ tập trung vào một giải pháp tối ưu duy nhất.
- Ý tưởng chính
  - **Đa dạng**: Chia không gian tìm kiếm thành các khu vực khác nhau, mỗi khu vực đại diện cho một dạng hành vi hoặc đặc điểm cụ thể của giải pháp
  - **Chất lượng cao**: Mỗi cá thể cần phải đạt được chất lượng cao theo một hàm mục tiêu
- *Không triển khai trực tiếp QD*: sử dụng NS kết hợp với thông tin từ fitness value (độ thích nghi)

Quality Diversity - QD



Hình: Minh họa Archive trong QD

Mục lục

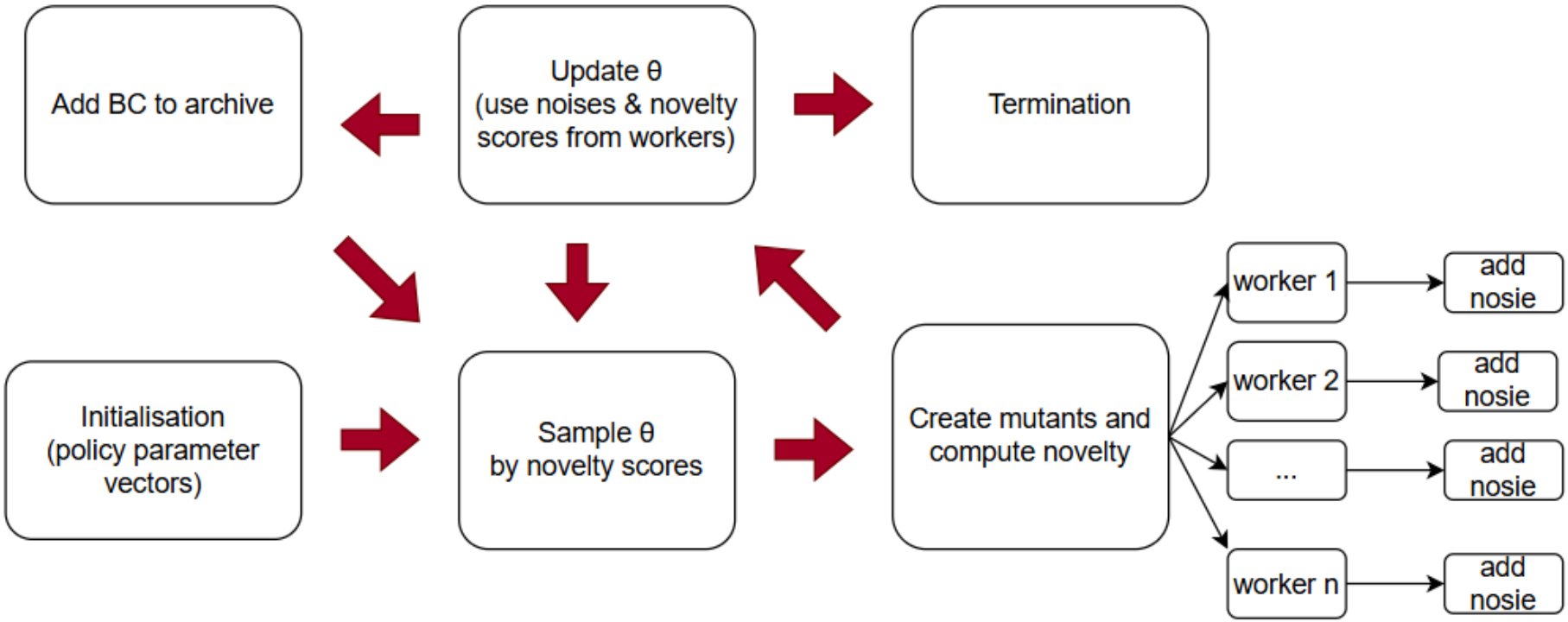
NS-ES

Algorithm 1 NS-ES

Input: learning rate α, noise standard deviation σ, number of policies to maintain M, iterations T, behavior characterization $b(\pi_\theta)$
Initialize: M randomly initialized policy parameter vectors $\{\theta_0^1, \theta_0^2, \ldots, \theta_0^M\}$, archive A, number of workers n
for $j = 1$ **to** M **do**
    Compute $b(\pi_{\theta_0^j})$
    Add $b(\pi_{\theta_0^j})$ to A
end for
for $t = 0$ **to** $T - 1$ **do**
    Sample θ_t^m from $\{\theta_t^1, \theta_t^2, \ldots, \theta_t^M\}$ via eq 1
    for $i = 1$ **to** n **do**
        Sample $\epsilon_i \sim \mathcal{N}(0, \sigma^2 I)$
        Compute $\theta_t^{i,m} = \theta_t^m + \epsilon_i$
        Compute $b(\pi_{\theta_t^{i,m}})$
        Compute $N_i = N(\theta_t^{i,m}, A)$
        Send N_i from each worker to coordinator
    end for
    Set $\theta_{t+1}^m = \theta_t^m + \alpha \frac{1}{n\sigma} \sum_{i=1}^n N_i \epsilon_i$
    Compute $b(\pi_{\theta_{t+1}^m})$
    Add $b(\pi_{\theta_{t+1}^m})$ to A
end for

Hình: Thuật toán NS-ES

NS-ES - working ideas



NS-ES - working ideas

- Khởi tạo

NS-ES - working ideas

- Khởi tạo
  - Tạo M vector tham số ngẫu nhiên $\theta_0^1, \theta_0^2, ..., \theta_0^M$ đại diện cho M agent.
  - Với mỗi agent, tính giá trị đặc trưng hành vi $b(\pi_{\theta_0^j})$ và thêm vào tập lưu trữ A.

NS-ES - working ideas

- Khởi tạo
  - Tạo M vector tham số ngẫu nhiên $\theta_0^1, \theta_0^2, ..., \theta_0^M$ đại diện cho M agent.
  - Với mỗi agent, tính giá trị đặc trưng hành vi $b(\pi_{\theta_0^j})$ và thêm vào tập lưu trữ A.
- Với mỗi vòng lặp t:

NS-ES - working ideas

- Khởi tạo
  - Tạo M vector tham số ngẫu nhiên $\theta_0^1, \theta_0^2, ..., \theta_0^M$ đại diện cho M agent.
  - Với mỗi agent, tính giá trị đặc trưng hành vi $b(\pi_{\theta_0^j})$ và thêm vào tập lưu trữ A.
- Với mỗi vòng lặp t:
  - Tính xác suất được chọn của mỗi θ_i bằng cách lấy độ mới lạ của $b(\pi_{\theta_i})$ đó chia cho tổng độ mới lạ của tất cả các BC trong archive A.
  - Chọn ngẫu nhiên một θ dựa theo các phân phối xác suất đã tính.

NS-ES - working ideas

- Khởi tạo
  - Tạo M vector tham số ngẫu nhiên $\theta_0^1, \theta_0^2, ..., \theta_0^M$ đại diện cho M agent.
  - Với mỗi agent, tính giá trị đặc trưng hành vi $b(\pi_{\theta_0^j})$ và thêm vào tập lưu trữ A.
- Với mỗi vòng lặp t:
  - Tính xác suất được chọn của mỗi θ_i bằng cách lấy độ mới lạ của $b(\pi_{\theta_i})$ đó chia cho tổng độ mới lạ của tất cả các BC trong archive A.
  - Chọn ngẫu nhiên một θ dựa theo các phân phối xác suất đã tính.
  - Gửi θ đến mỗi worker. Thêm nhiễu và tính toán $b(\pi_\theta)$ cùng với $N(\theta, A)$

NS-ES - working ideas

- Khởi tạo
  - Tạo M vector tham số ngẫu nhiên $\theta_0^1, \theta_0^2, ..., \theta_0^M$ đại diện cho M agent.
  - Với mỗi agent, tính giá trị đặc trưng hành vi $b(\pi_{\theta_0^j})$ và thêm vào tập lưu trữ A.
- Với mỗi vòng lặp t:
  - Tính xác suất được chọn của mỗi θ_i bằng cách lấy độ mới lạ của $b(\pi_{\theta_i})$ đó chia cho tổng độ mới lạ của tất cả các BC trong archive A.
  - Chọn ngẫu nhiên một θ dựa theo các phân phối xác suất đã tính.
  - Gửi θ đến mỗi worker. Thêm nhiễu và tính toán $b(\pi_\theta)$ cùng với $N(\theta, A)$
  - Lấy các kết quả từ các worker và tiến hành cập nhật θ

NS-ES - Experiment configs

- Policy: Mạng nơ-ron (trạng thái $\rightarrow$ xác suất thực hiện hành động).
- Policy parameters: trọng số của mạng
- Population: Tập hợp các policy ($M = 3$)
- n_workers: tận dụng khả năng tính toán song song ($n = 5$)
- Behavior characterization: Tính trung bình các state
- Behavior archive: list các BC với môi trường CartPole-v0 và MountainCar-v0 / deque(maxlen $= 500$) (BreakoutNoFrameskip-v4)
- Novelty score (N function): Trung bình euclidean distance của BC hiện tại với $k = 5$ BC gần nhất
- Sample: Theo phân phối xác suất được tính từ novelty score
- Mutation: Gửi policy đến từng worker $\rightarrow$ Thêm nhiễu, tính toán BC và novelty score (N) $\rightarrow$ Tổng hợp kết quả $\rightarrow$ Thực hiện bước thêm nhiễu cuối cùng cho policy

NS-ES: problem

- NS-ES giúp agent thoát khỏi các điểm tối ưu cục bộ trong hàm phần thưởng.

NS-ES: problem

- NS-ES giúp agent thoát khỏi các điểm tối ưu cục bộ trong hàm phần thưởng.
- **Vấn đề**: NS-ES bỏ qua hoàn toàn tín hiệu phần thưởng trong khi đây là một thông tin hữu ích.

$\rightarrow$ Ảnh hưởng đến hiệu suất của thuật toán

NS-ES: problem

- NS-ES giúp agent thoát khỏi các điểm tối ưu cục bộ trong hàm phần thưởng.
- **Vấn đề**: NS-ES bỏ qua hoàn toàn tín hiệu phần thưởng trong khi đây là một thông tin hữu ích.

$\rightarrow$ Ảnh hưởng đến hiệu suất của thuật toán

Một biến thể của NS-ES sử dụng thông tin phần thưởng đó là NSR-ES.

- NSR-ES về cơ bản giống với NS-ES, điểm khác biệt duy nhất là NSR-ES sử dụng thêm "tín hiệu phần thưởng" để cập nhật trọng số

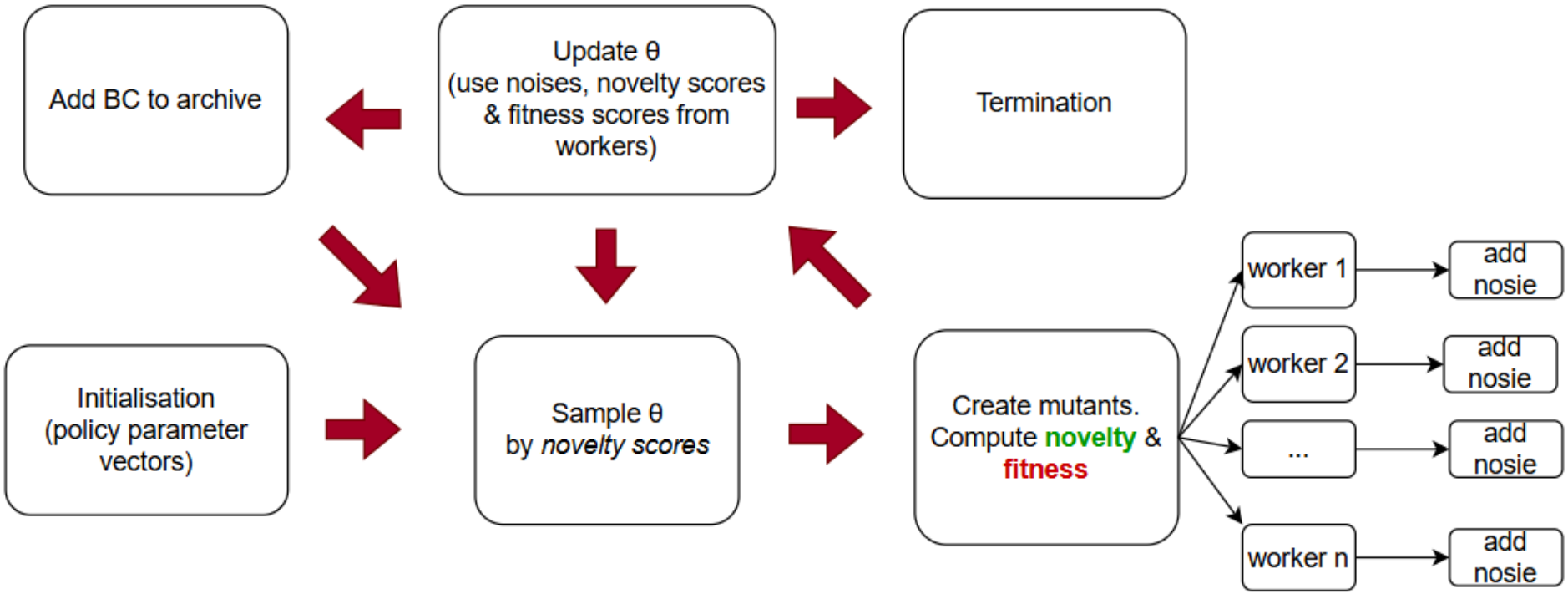
NSR-ES

Algorithm 2 NSR-ES

Input: learning rate α, noise standard deviation σ, number of policies to maintain M, iterations T, behavior characterization $b(\pi_\theta)$
Initialize: M sets of randomly initialized policy parameters $\{\theta_0^1, \theta_0^2, \ldots, \theta_0^M\}$, archive A, number of workers n
for $j = 1$ **to** M **do**
    Compute $b(\pi_{\theta_0^j})$
    Add $b(\pi_{\theta_0^j})$ to A
end for
for $t = 0$ **to** $T - 1$ **do**
    Sample θ_t^m from $\{\theta_t^0, \theta_t^1, \ldots, \theta_t^M\}$ via eq. 1
    for $i = 1$ **to** n **do**
        Sample $\epsilon_i \sim \mathcal{N}(0, \sigma^2 I)$
        Compute $\theta_t^{i,m} = \theta_t^m + \epsilon_i$
        Compute $b(\pi_{\theta_t^{i,m}})$
        Compute $N_i = N(\theta_t^{i,m}, A)$
        Compute $F_i = f(\theta_t^{i,m})$
        Send N_i and F_i from each worker to coordinator
    end for
    Set $\theta_{t+1}^m = \theta_t^m + \alpha \frac{1}{n\sigma} \sum_{i=1}^n \frac{N_i + F_i}{2} \epsilon_i$
    Compute $b(\pi_{\theta_{t+1}^m})$
    Add $b(\pi_{\theta_{t+1}^m})$ to A
end for

Hình: Thuật toán NSR-ES

NSR-ES


Add BC to archive
Update θ
(use noises, novelty scores & fitness scores from workers)
Termination
Initialisation
(policy parameter vectors)
Sample θ
by *novelty scores*
Create mutants.
Compute novelty & fitness
worker 1
worker 2
...
worker n
add nosie
add nosie
add nosie
add nosie

NS-ESc

Algorithm 3 NS-ESc

Input: noise standard deviation σ, population size M, max generations G, behavior characterization $b(\pi_\theta)$, number of individuals to remain k

Initialize: a population of M random policy parameters $pop = \left[\theta_0^1; \theta_0^2; \ldots; \theta_0^M\right]$, archive A

for $j = 1$ **to** M **do**

Compute $b\left(\pi_{\theta_0^j}\right)$; Add $b\left(\pi_{\theta_0^j}\right)$ to A

end for

for $t = 0$ **to** $G - 1$ **do**

next_pop = [ ], *mutant_pop* = [ ]

for $i = 1$ **to** M **do**

Compute $b\left(\pi_{\theta_t^i}\right)$

Compute $N_i = N\left(\theta_t^i, A\right)$

end for

Ranking current population by novelty score: *ranked_pop* = ranking(*pop*)

Select top k θ_t^j from *ranked_pop*, add to *next_pop* and *mutant_pop*

While *mutant_population size* < M - k **do**

Randomly select θ_t^j from the rest of *ranked_pop*

Add θ_t^j to *mutant_pop*

end while

for $i = 1$ **to** $M - k$ **do**

Select θ_t^i from *mutant_pop*

Sample $\varepsilon_i \sim \mathrm{N}\left(0, \sigma^2 I\right)$

Compute $\theta_t^i = \theta_t^i + \varepsilon_i$; Add $\pi_{\theta_t^i}$ to *next_pop*

Compute $b\left(\pi_{\theta_t^i}\right)$; Add $b\left(\pi_{\theta_t^i}\right)$ to A

end for

Update *pop* = *next_pop*

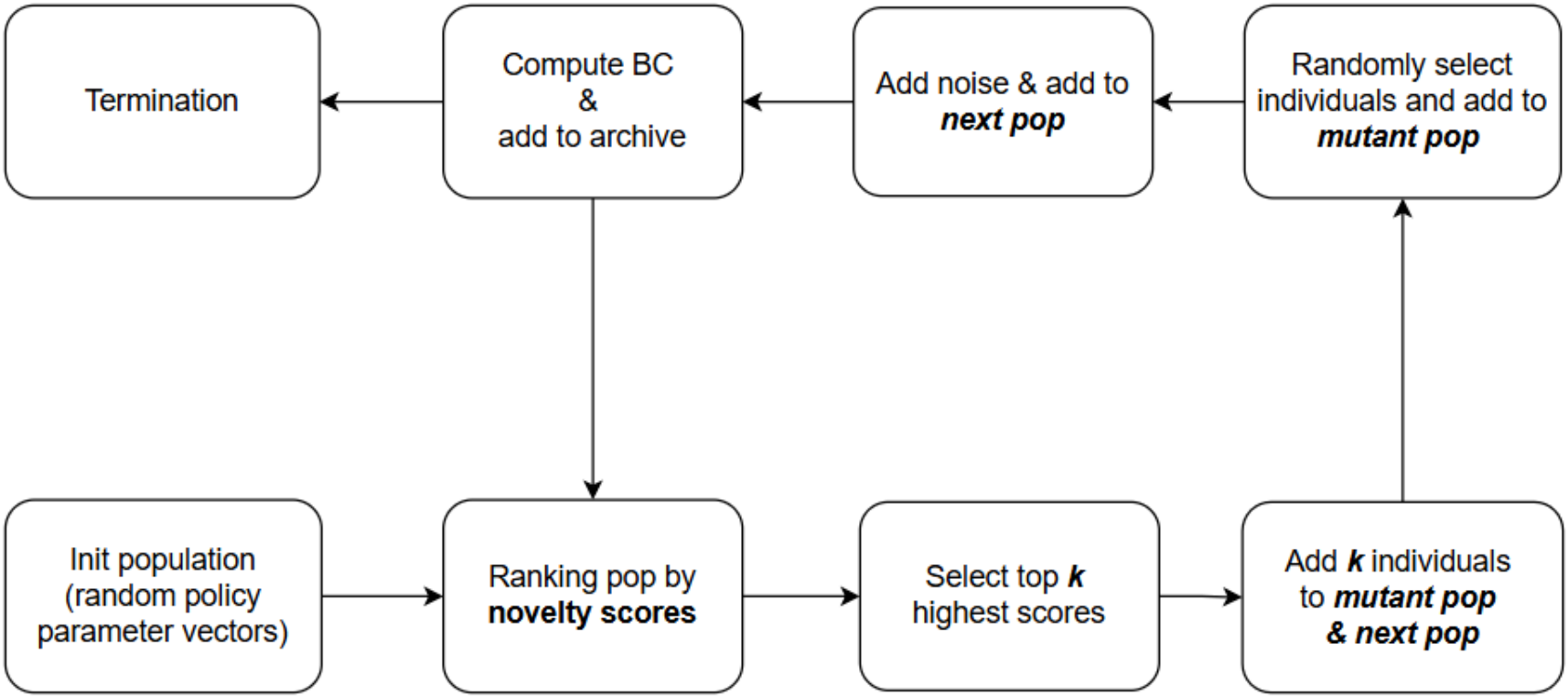
end for

NS-ESc - working ideas

Điểm khác so với NS-ES:

- NS-ES huấn luyện qua từng vòng lặp (iteration), trong khi NS-ESc huấn luyện theo thế hệ.
- Ở mỗi thế hệ, k cá thể có *novelty score* **cao nhất** sẽ được giữ lại và đưa vào **next_pop** (quần thể ở thế hệ kế tiếp).
- **mutant_pop** được xác định bằng cách lấy k cá thể có novelty score cao nhất và chọn ngẫu nhiên trong số các cá thể còn lại cho đến khi đủ $M - k$ cá thể.
- Phép đột biến được thực hiện trên `mutant_pop` và các cá thể sinh ra được bổ sung vào `next_pop` để làm quần thể cho thế hệ kế tiếp.

NS-ESc


Termination
Compute BC & add to archive
Add noise & add to *next pop*
Randomly select individuals and add to *mutant pop*
Init population (random policy parameter vectors)
Ranking pop by **novelty scores**
Select top *k* highest scores
Add *k* individuals to *mutant pop* & *next pop*

NSR-ESc

Algorithm 4 NSR-ESc

Input: noise standard deviation σ, population size M, max generations G, behavior characterization $b(\pi_\theta)$, number of individuals to remain k

Initialize: a population of M random policy parameters $pop = \left[\theta_0^1; \theta_0^2; \ldots; \theta_0^M\right]$, archive A

for $j = 1$ **to** M **do**

Compute $b\left(\pi_{\theta_0^j}\right)$; Add $b\left(\pi_{\theta_0^j}\right)$ to A

end for

for $t = 0$ **to** $G - 1$ **do**

next_pop = [ ], *mutant_pop* = [ ]

for $i = 1$ **to** M **do**

Compute $b\left(\pi_{\theta_t^i}\right)$

Compute $N_i = N\left(\theta_t^i, A\right)$, $F_i = F\left(\theta_t^i, A\right)$, $score_i = \dfrac{N_i + F_i}{2}$

end for

Ranking current population by score: *ranked_pop* = ranking(*pop*)

Select top k θ_t^j from *ranked_pop*, add to *next_pop* and *mutant_pop*

While *mutant_population size* < M - k **do**

Randomly select θ_t^j from the rest of *ranked_pop*

Add θ_t^j to *mutant_pop*

end while

for $i = 1$ **to** $M - k$ **do**

Select θ_t^i from *mutant_pop*

Sample $\varepsilon_i \sim \mathrm{N}\left(0, \sigma^2 I\right)$

Compute $\theta_t^i = \theta_t^i + \varepsilon_i$; Add $\pi_{\theta_t^i}$ to *next_pop*

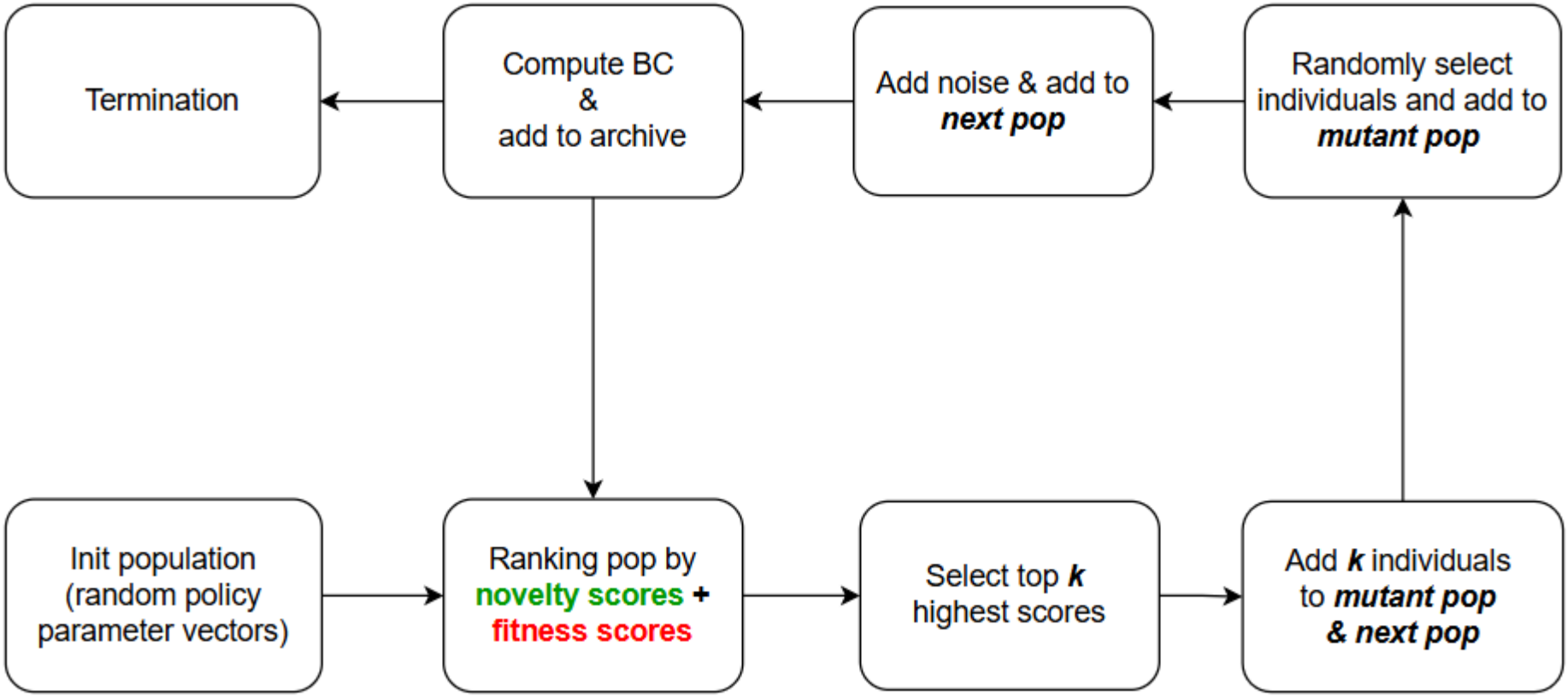
Compute $b\left(\pi_{\theta_t^i}\right)$; Add $b\left(\pi_{\theta_t^i}\right)$ to A

end for

Update *pop* = *next_pop*

end for

NSR-ESc


Termination
Compute BC & add to archive
Add noise & add to next pop
Randomly select individuals and add to mutant pop
Init population (random policy parameter vectors)
Ranking pop by novelty scores + fitness scores
Select top k highest scores
Add k individuals to mutant pop & next pop

Mục lục

Environment

- CartPole-v0
- MountainCar-v0
- BreakoutNoFrameskip-v4

Experiment results

Bảng: Best result for each method

Environment	Method	Best score	Game play
CartPole-v0	DQN	500	video
	NS-ES	500	video
	NSR-ES	500	video
	NS-ESc	500	video
	NSR-ESc	500	video

Experiment results

Bảng: Best result for each method

Environment	Method	Best score	Complete	Game play
MountainCar-v0	DQN	-91	YES	video
	NS-ES	-129.33	YES	video
	NSR-ES	-183	YES	video
	NS-ESc	-115.66	YES	video
	NSR-ESc	-109.66	YES	video

Experiment results

Bảng: Best result for each method

Environment	Method	Best score	Game play
BreakoutNoFrameskip-v4	DQN	38	video
	NS-ES	16	video
	NSR-ES	21	video
	NS-ESc	13	video
	NSR-ESc	22	video

Comments

- Trên cả 3 môi trường, DQN cho kết quả tốt nhất
- Trong số 4 thuật toán ES, NSR-ESc cho kết quả tốt nhất
- Do cách cài đặt của các thuật toán khác nhau nên bảng kết quả không đưa ra thời gian thực thi, tuy nhiên khi thực nghiệm, nhóm có một số nhận xét như sau:
  - Đối với môi trường CartPole, mặc dù kết quả sau cùng của 5 thuật toán đều bằng nhau nhưng trong quá trình thực nghiệm, thuật toán NSR-ESc có tốc độ hội tụ nhanh nhất
  - Đối với môi trường MountainCar, thuật toán DQN hội tụ nhanh hơn so với các thuật toán khác. Ngoài ra, trong số 4 thuật toán ES thì thuật toán NS-ES có thời gian thực thi lâu nhất
- Mặc dù kết quả đạt được không tốt bằng DQN, tuy nhiên các thuật toán ES đã cho thấy tiềm năng ứng dụng của nó khi giải quyết các bài toán về DRL