Kiến trúc mạng tối ưu đa mục tiêu

Ứng dụng thuật toán NSGA-II tìm kiếm kiến trúc mạng tích chập tối thiểu lỗi và độ trễ

Nhóm 8 - Neural Architecture Search

Ngày 17 tháng 11 năm 2023

Mục lục

Bài toán tối ưu đa mục tiêu

Bài toán tối ưu đa mục tiêu tổng quát có thể phát biểu dưới dạng:

$$\begin{aligned} & \min_{\mathbf{x}\in\Omega} F(\mathbf{x}) \\ \text{với} \quad & g_i(\mathbf{x}) \leq 0, \forall i = \overline{1,m} \\ \text{và} \quad & h_j(\mathbf{x}) = 0, \forall i = \overline{1,n} \end{aligned}$$

Trong đó:

- $\mathbf{x} = [x_1, x_2, \ldots, x_d]$ là biến vector quyết định d chiều.
- $F(\mathbf{x}) = [f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_k(\mathbf{x})]$ gồm k mục tiêu ứng với k hàm số cần cực tiểu hóa.

Trong nhiều trường hợp, việc tìm lời giải cực tiểu toàn cục duy nhất cho tất cả mục tiêu là khó khăn hoặc thậm chí không thể có. Do đó, ta không sử dụng khái niệm tối ưu thông thường.

Tính trội Pareto

Xét bài toán cực tiểu đa mục tiêu, lời giải $\mathbf{x}^{(1)}$ được gọi là **trội hơn** so với lời giải $\mathbf{x}^{(2)}$, kí hiệu $\mathbf{x}^{(1)} \preceq \mathbf{x}^{(2)}$, nếu và chỉ nếu:

$$\begin{aligned} \forall i \in \{1, 2, \ldots, k\} : &\quad f_i(\mathbf{x}^{(1)}) \leq f_i(\mathbf{x}^{(2)}) \\ \exists j \in \{1, 2, \ldots, k\} : &\quad f_j(\mathbf{x}^{(1)}) < f_j(\mathbf{x}^{(2)}) \end{aligned}$$

Nghĩa là $\mathbf{x}^{(1)}$ không tệ hơn $\mathbf{x}^{(2)}$ ở tất cả các hàm mục tiêu và có ít nhất một hàm mục tiêu mà $\mathbf{x}^{(1)}$ tốt hơn hẳn $\mathbf{x}^{(2)}$.

Tập tối ưu Pareto

Với bài toán cực tiểu đa mục tiêu $F(\mathbf{x})$, tập tối ưu Pareto $\mathcal{P}^*$ được định nghĩa là:

$$\mathcal{P}^* = \{\mathbf{x} \in \Omega | \nexists \mathbf{x}' \in \Omega : F(\mathbf{x}') \preceq F(\mathbf{x})\}$$

Nói cách khác, nghiệm tối ưu của bài toán tối ưu đa mục tiêu là một tập các lời giải đôi một không trội hơn nhau và cũng không có lời giải nào khác trội hơn một trong các lời giải này.

Biên Pareto $\mathcal{PF}^*$ được định nghĩa là:

$$\mathcal{PF}^* = \{F(\mathbf{x}) | \mathbf{x} \in \mathcal{P}^*\}$$

Giải thuật tiến hóa cho bài toán tối ưu đa mục tiêu

- Với MOEA, quần thể tại thế hệ thứ t có tập tối ưu Pareto $\mathcal{P}_{\text{current}}(t)$.
- Ta sử dụng *quần thể phụ* lưu trữ các tập tối ưu Pareto tìm được từ thông tin của t thế hệ trước đó, các lời giải không trội hơn lẫn nhau trong $\mathcal{P}_{\text{known}}(t) = \mathcal{P}_{\text{current}}(t) \cup \mathcal{P}_{\text{known}}(t-1)$.
- Tập tối ưu Pareto chính xác $\mathcal{P}_{\text{true}}$ thường không thể tìm được chính xác.

Mục tiêu của MOEA là trả về $\mathcal{P}_{\text{known}} \subseteq \mathcal{P}_{\text{true}}$ hoặc xấp xỉ $\mathcal{P}_{\text{true}}$ bởi $\mathcal{P}_{\text{known}}$.

Thuật toán di truyền sắp xếp không trội

Thuật toán 1: NSGA-II

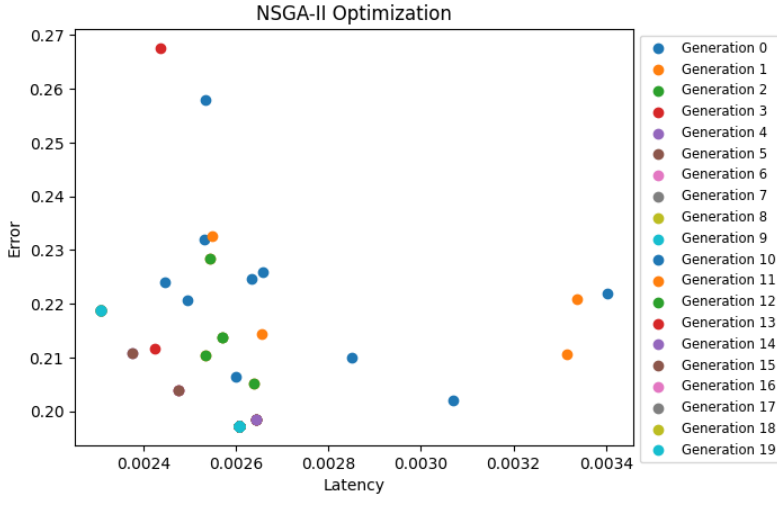
1. Khởi tạo quần thể $\mathbb{P}$ gồm $\mathcal{N}$ cá thể.
2. Tính toán giá trị các hàm mục tiêu với mỗi cá thể.
3. Sắp xếp các cá thể thành các biên Pareto theo hạng.
4. Tạo quần thể con $\mathcal{C}$:
   i. Chọn cha mẹ theo thể thức đấu đôi.
   ii. Lai ghép và đột biến.
5. Vòng lặp thế hệ $i = 1, 2, \ldots, g$ với quần thể $\mathbb{P} \cup \mathbb{C}$:
   i. Tính hạng theo biên Pareto.
   ii. Trên mỗi biên Pareto tính độ đông đúc từng cá thể.
   iii. Lấy ra đủ $\mathcal{N}$ cá thể tốt nhất dựa trên hạng và độ đông đúc để cập nhật quần thể $\mathbb{P}$.
   iv. Tạo quần thể con $\mathbb{C}$.
   v. Bổ sung $\mathbb{C}$ vào $\mathbb{P}$.

Chi tiết triển khai

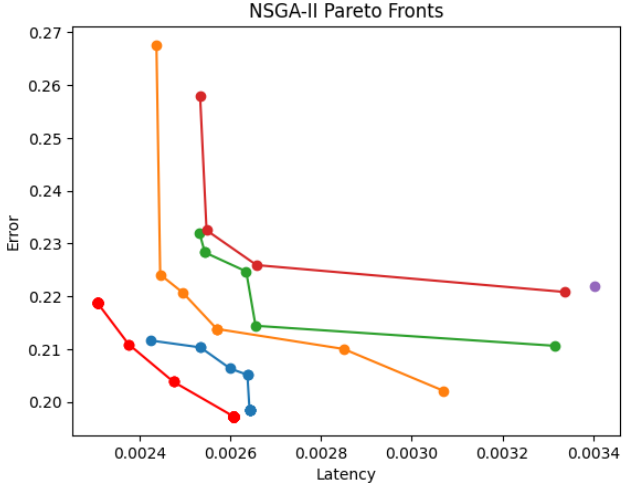
- Bộ dữ liệu CIFAR-10, kích thước lô 128, trình tối ưu ADAM, 5 vòng lặp huấn luyện.
- Mỗi nút là 3 phép toán theo thứ tự: tích chập, chuẩn hóa theo lô, ReLU.
- Kiến trúc mạng có 2 khối, mỗi khối có 5 nút.
- Mã hóa toàn mạng bởi chuỗi nhị phân dài 20, thể hiện kết nối giữa các nút.

Kết quả thu được:

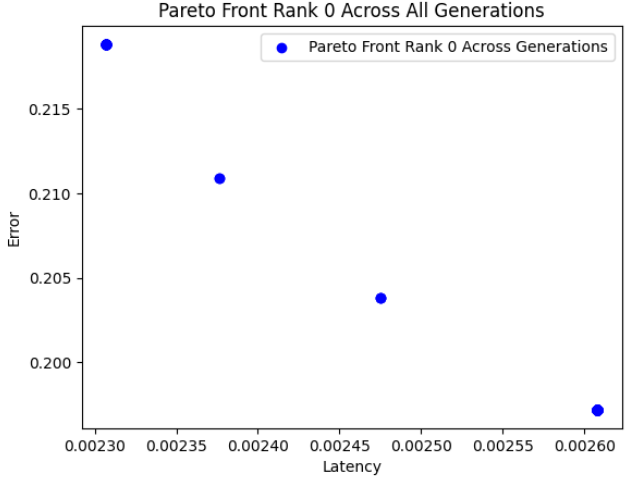
- Thu được 4 lời giải Pareto.
- Thời gian chạy: 5 tiếng ~ 0.2 GPU days.



Hình 1: Kết quả khi sử dụng 10 cá thể, 20 thế hệ



Hình 2: Các biên Pareto tìm được



Hình 3: Biên Pareto hạng 0, lời giải của bài toán

Một số hướng tiếp cận

Một trở ngại lớn với NAS là mỗi lần đánh giá đều tốn kém rất nhiều – thường mất hàng giờ hoặc hàng ngày để huấn luyện hoàn toàn một mạng.

- **NAS Efficiency:** Sử dụng chia sẻ trọng số giữa các mô hình ứng viên để giảm thời gian huấn luyện trong quá trình đánh giá.
- **Pruning:** Giảm số lượng tham số trong một mạng.
  i Bằng cách xác định chỉ số quan trọng (saliency) cho từng tham số, và các tham số ít quan trọng hơn được loại bỏ. Yêu cầu huấn luyện trước khi tính toán tiêu chí quan trọng.
  ii Cách khác là cắt bớt tham số ngay từ thời điểm khởi tạo, không yêu cầu huấn luyện. Một lần truyền xuôi/truyền ngược được sử dụng để tính toán tiêu chí quan trọng.

Các phương pháp tiêu biểu

- **AtomNAS:** Sử dụng cắt tỉa kênh (channel pruning) trong quá trình NAS, giúp phát triển các phép tính chập với nhiều kích thước kernel khác nhau, tạo ra các mô hình mạng tùy chỉnh hiệu quả. Đạt được sự cân bằng giữa hiệu suất và kích thước của mô hình.
- **Blockswap:** Sử dụng thông tin Fisher ngay tại thời điểm khởi tạo để đánh giá các thành phần nhẹ (lightweight primitives). Thông qua việc này, Blockswap có thể chọn lọc và thay thế các phần của mạng, nhằm giảm bớt yêu cầu về tính toán và tài nguyên.
- **NASWOT:** Đề xuất một chỉ số đánh giá mới cho NAS, dựa trên sự tương quan của các Jacobian với các đầu vào khác nhau. Phương pháp này cho phép xếp hạng và so sánh hiệu quả của các mạng mà không cần phải trải qua quá trình huấn luyện dài.

Proxy thông thường: EcoNAS

Trong NAS dựa trên mẫu thông thường, một chế độ huấn luyện proxy thường được sử dụng để dự đoán độ chính xác của một mô hình thay vì huấn luyện đầy đủ.

- Bằng cách tính hệ số tương quan xếp hạng Spearman (Spearman ρ) của một nhiệm vụ proxy với độ chính xác kiểm tra cuối cùng. Hệ số Spearman xem xét liệu có một mối quan hệ đơn điệu giữa xếp hạng của các giá trị hay không. Điều này có nghĩa là nếu giá trị của một biến tăng (nhiệm vụ proxy) thì giá trị của biến kia (độ chính xác) cũng tăng, và ngược lại.
- Hệ số Spearman $\rho \in [-1, 1]$. Một giá trị ρ gần 1 chỉ ra một mối quan hệ tăng đơn điệu mạnh, một giá trị gần -1 chỉ ra một mối quan hệ giảm đơn điệu mạnh, và một giá trị gần 0 chỉ ra rằng không có mối quan hệ đơn điệu rõ ràng nào.
- Proxy được sử dụng ở đây là một phương pháp huấn luyện giảm bớt tính toán, một trong 4 yếu tố sau được giảm xuống: (1) số lượng vòng lặp huấn luyện, (2) số lượng mẫu huấn luyện, (3) độ phân giải đầu vào, hoặc (4) kích thước mô hình (được kiểm soát qua số lượng kênh sau phép tính chập đầu tiên).

Proxy không chi phí

- Sử dụng các proxy thay thế cho độ chính xác của mạng để tăng tốc NAS.
- Một proxy đơn giản có thể sử dụng là `grad_norm` trong đó chúng ta lấy tổng chuẩn Euclid của gradient sau một minibatch dữ liệu huấn luyện.
- Sử dụng thêm các chỉ số khác đã được giới thiệu trước đó trong bối cảnh cắt tỉa (pruning) tham số ở mức độ tham số đơn lẻ - một chỉ số quan trọng được tính toán để xếp hạng các tham số và loại bỏ những tham số ít quan trọng nhất.

Ta sẽ thích nghi các chỉ số này để đánh giá và xếp hạng toàn bộ các mô hình mạng cho NAS.

Các phương pháp cắt tỉa: Snip, Grasp và Synaptic Flow

- **SNIP:** Phương pháp cắt tỉa tham số dựa trên một chỉ số quan trọng (saliency), được tính toán ngay tại thời điểm khởi tạo sử dụng một minibatch dữ liệu. Tiêu chí này ước lượng sự thay đổi trong mất mát khi một tham số cụ thể bị loại bỏ.
- **GRASP:** Một sự cải tiến của SNIP. Nó cố gắng ước lượng sự thay đổi trong độ lớn gradient (thay vì mất mát) khi một tham số bị cắt tỉa, nhằm cải thiện chất lượng của việc lựa chọn tham số để cắt bỏ.

- **SYNAPTIC FLOW:** Một phiên bản được sửa đổi (synflow) giúp tránh sự sụp đổ của các lớp khi thực hiện cắt tỉa tham số. Thay vì sử dụng minibatch dữ liệu huấn luyện và mất mát cross-entropy (như trong snip hoặc grasp), với synflow, tính toán mất mát chỉ đơn giản là tích của tất cả các tham số trong mạng; do đó, không cần dữ liệu để tính toán mất mát này hay chính chỉ số synflow.

$$\texttt{snip}: S_p(\theta) = \left| \frac{\partial \mathcal{L}}{\partial \theta} \odot \theta \right|, \quad \texttt{grasp}: S_p(\theta) = -\left(H \frac{\partial \mathcal{L}}{\partial \theta} \right) \odot \theta, \quad \texttt{synflow}: S_p(\theta) = \frac{\partial \mathcal{L}}{\partial \theta} \odot \theta$$

Trong đó $\mathcal{L}$ là hàm mất mát với tham số θ, H là ma trận Hesse, S_p là độ quan trọng của mỗi tham số và $\odot$ là tích Hadamard.

Phương pháp Fisher

- **FISHER:** Phương pháp Fisher được sử dụng để cắt tỉa kênh bằng cách loại bỏ những kênh kích hoạt (và các tham số liên quan) được ước lượng là có ít ảnh hưởng nhất đến hàm mất mát. Điều này được thực hiện bằng cách tính toán:

$$\texttt{fisher:} S_z(z) = \left(\frac{\partial \mathcal{L}}{\partial z} z\right)^2, \quad S_n = \sum_{i=1}^{M} S(z_i)$$

Trong đó S_z là độ quan trọng của mỗi kích hoạt z và M là chiều dài của bản đồ đặc trưng vector hóa.

Phương pháp Jacobi

- **JACOBIAN COVARIANCE:** Chỉ số Jacobi ghi lại sự tương quan của các kích hoạt trong mạng khi chịu ảnh hưởng của các đầu vào khác nhau trong một minibatch dữ liệu – sự tương quan càng thấp thì mạng dự kiến sẽ hoạt động càng tốt vì nó có thể phân biệt tốt giữa các đầu vào khác nhau.

NAS-Bench-201

- NAS-Bench-201 là một chuẩn mực dành riêng cho việc thử nghiệm các thuật toán NAS, chứa 15625 mô hình CNN từ không gian tìm kiếm dựa trên cell.
- Sử dụng hệ số tương quan xếp hạng Spearman để định lượng mức độ hiệu quả của proxy trong việc xếp hạng các mô hình so với xếp hạng cuối cùng dựa trên độ chính xác kiểm tra.

So sánh EcoNAS proxy với Zero-cost proxies:

- Mặc dù EcoNAS là hiệu quả trong một số trường hợp, nhưng nó có hệ số tương quan Spearman chỉ là 0.61 khi đánh giá trên tất cả 15625 mô hình, thấp hơn so với kết quả 0.87 đạt được trên 50 mô hình trong nghiên cứu EcoNAS.
- Vậy, *synflow* cho thấy kết quả tốt nhất trên cả ba bộ dữ liệu, *jacob_cov* là chỉ số tiếp theo. Một chỉ số mới, gọi là *vote*, đã được thêm vào và hoạt động tốt hơn bất kỳ chỉ số đơn lẻ nào. Cụ thể, chỉ số này hoạt động bằng cách lấy "ý kiến đa số" (majority vote) giữa 3 chỉ số proxy khác nhau: *synflow*, *jacob_cov* và *snip*.

Table 1: Spearman ρ of zero-cost proxies on NAS-Bench-201.

Dataset	grad_norm	snip	grasp	fisher	synflow	jacob_cov	vote
CIFAR-10	0.58	0.58	0.48	0.36	0.74	0.73	0.82
CIFAR-100	0.64	0.63	0.54	0.39	0.76	0.71	0.83
ImageNet16-120	0.58	0.58	0.56	0.33	0.75	0.71	0.82

PyTorch CV

- Thực hành Zero-cost proxies trong cơ sở dữ liệu PyTorchCV, bao gồm các mạng tiên tiến như ResNets, DenseNets, MobileNets, và EfficientNets.
- Đánh giá 50 mô hình cho CIFAR-10, CIFAR-100, SVHN, và 200 mô hình cho ImageNet1k.
- Chỉ số *synflow* thể hiện ưu việt đặc biệt trên SVHN, trong khi chỉ số *grasp* không hiệu quả trong môi trường này. Chỉ số *jaco_cov* cũng không thành công trong môi trường này mặc dù nó hiệu quả trong NAS-Bench-201.

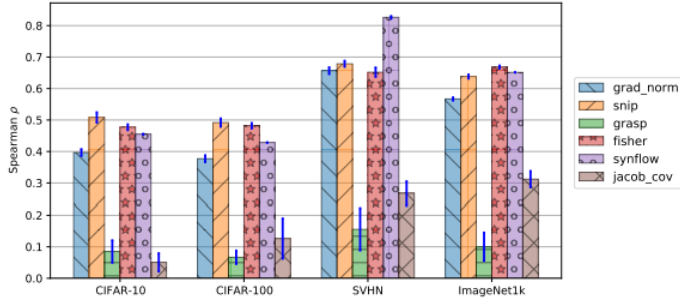


Figure 3: Performance of zero-cost metrics on PyTorchCV models (averaged over 5 seeds).

Các không gian tìm kiếm khác

- Khám phá các chỉ số Zero-cost proxies với các benchmarks NAS khác:
  - i NAS-Bench-101 là chuẩn mực NAS đầu tiên và lớn nhất với hơn 423k mô hình CNN.
  - ii NAS-Bench-NLP nghiên cứu về kiến trúc của 14k recurrent cells trong các nhiệm vụ xử lý ngôn ngữ tự nhiên.
  - iii NAS-Bench-ASR là cơ sở dữ liệu trong nhà cho mô hình nhận dạng giọng nói tự động dựa trên tính toán, đánh giá trên bộ dữ liệu TIMIT.
- So sánh cho thấy chỉ số *synflow* là duy nhất đạt được tính nhất quán trên tất cả các chuẩn mực đã phân tích. Tuy nhiên, hệ số tương quan của *synflow* thấp hơn nhiều so với NAS-Bench-201 (0.3 so với 0.8).

Table 2: Spearman ρ of zero-cost proxies on other NAS search spaces.

	grad_norm	snip	grasp	fisher	**synflow**	jacob_cov
NAS-Bench-101	0.20	0.16	0.45	0.26	**0.37**	0.38
NAS-Bench-NLP	-0.21	-0.19	0.16	–	**0.34**	0.56
NAS-Bench-ASR	0.07	0.01	–	0.02	**0.40**	-0.37

Zero-cost NAS

Phương pháp *NAS không huấn luyện* (NASWOT) sử dụng chỉ số `jacob_cov` để đánh giá và chọn ra mô hình tốt nhất từ một tập hợp các mô hình được chọn ngẫu nhiên. Điểm nổi bật của phương pháp này là sự đơn giản và chi phí tính toán thấp.

Mở rộng phương pháp này một chút:

- Thay vì chỉ huấn luyện mô hình hàng đầu, chúng tôi tiếp tục huấn luyện các mô hình (từ tốt nhất đến tệ nhất theo thứ tự được xếp hạng bởi chỉ số zero-cost) cho đến khi đạt được độ chính xác mong muốn.
- Tuy nhiên, cách tiếp cận này chỉ có thể sản xuất kết quả tốt như chất lượng của chỉ số được sử dụng – không có sự đảm bảo (chỉ có bằng chứng thực nghiệm) rằng các chỉ số này sẽ hoạt động tốt trên tất cả các bộ dữ liệu.

Zero-cost NAS

Do đó, cũng nghiên cứu cách tích hợp các chỉ số không tốn kém vào các thuật toán NAS hiện tại như: Học tăng cường (Reinforcement Learning), Tìm kiếm tiến hóa theo tuổi thọ (Aging Evolution) và Tìm kiếm dựa trên bộ dự đoán (Predictor-based NAS). Cụ thể hơn, chúng ta tiến hành điều tra việc tăng cường các thuật toán tìm kiếm này thông qua:

i **Zero-Cost Warmup Phase.**

- Đây là giai đoạn đầu tiên trong quá trình tìm kiếm, nơi mà các chỉ số không tốn kém được sử dụng để đánh giá nhanh các mô hình mạng nơ-ron.
- Trong giai đoạn này, không cần thực hiện huấn luyện toàn diện cho mỗi mô hình; thay vào đó, các chỉ số được tính toán dựa trên một lượng nhỏ dữ liệu (như một minibatch) để xác định tiềm năng của các mô hình.

ii **Zero-Cost Move Proposal.**

- Trong cách tiếp cận này, các chỉ số không tốn kém được sử dụng để hỗ trợ việc đưa ra quyết định về bước di chuyển tiếp theo trong quá trình tìm kiếm NAS.
- Điều này có nghĩa là sau mỗi bước hoặc vòng lặp trong quá trình tìm kiếm, thay vì dựa vào kết quả từ một quá trình huấn luyện đầy đủ, thuật toán sẽ sử dụng các chỉ số không tốn kém để đánh giá nhanh và quyết định xem có nên chuyển đổi sang một kiến trúc mạng mới hay không.

Zero-cost Warmup Phase

Tham số chính trong khởi động không tốn kém là số lượng mô hình mà chúng ta tính toán và sử dụng chỉ số không tốn kém (N), con số này thường lớn hơn nhiều so với số lượng mô hình mà chúng ta đủ khả năng huấn luyện ($T \ll N$).

- **Aging Evolution:** Đánh giá N mô hình ngẫu nhiên với chỉ số proxy của chúng và chọn những mô hình được xếp hạng cao nhất làm quần thể khởi đầu.

- **Reinforcement Learning:**

i Lấy mẫu N mô hình ngẫu nhiên và sử dụng điểm số không tốn kém của chúng để thưởng cho bộ điều khiển, từ đó thiên vị nó hướng tới việc chọn các kiến trúc có giá trị cao của các chỉ số được chọn.

ii Trong giai đoạn khởi động, phần thưởng cho bộ điều khiển được tính toán bằng cách chuẩn hóa tuyến tính các giá trị trả về bởi các hàm proxy vào khoảng $[-1, 1]$.

- **Binary Predictor:**

i Đây là một mô hình dựa trên Mạng tích chập đồ thị (Graph Convolutional Network - GCN) so sánh hiệu suất tương đối giữa 2 mô hình mạng nơ-ron. Thay vì dựa trên độ chính xác thực tế của mô hình, bộ dự đoán này được huấn luyện để sử dụng các chỉ số không tốn kém.

ii Trong quá trình khởi động, bộ dự đoán được "huấn luyện" bằng cách xem xét hiệu suất tương đối của các cặp mô hình dựa trên chỉ số không tốn kém.

iii Các mô hình được xếp hạng bởi bộ dự đoán sau mỗi vòng huấn luyện (bao gồm cả giai đoạn khởi động).

Zero-cost Move Proposal

Một tham số phổ biến cho các thuật toán đề xuất di chuyển được ký hiệu là R. Tham số này chỉ định số lượng mô hình mà thuật toán có thể đánh giá bằng chỉ số không tốn kém mỗi khi chọn một mô hình để huấn luyện. Có nghĩa là ở mỗi bước chọn mô hình, thuật toán sẽ đánh giá R mô hình khác nhau dựa trên chỉ số không tốn kém trước khi quyết định mô hình nào sẽ được huấn luyện tiếp.

- **Aging Evolution:**

i Trong thuật toán này, "đề xuất di chuyển" được sử dụng để thực hiện các "biến đổi có hướng dẫn".

ii Khi một mô hình được biến đổi, thay vì thực hiện điều này một cách ngẫu nhiên, thuật toán xem xét tất cả các biến đổi có thể từ mô hình hiện tại, đánh giá chúng bằng các proxy không tốn kém và chọn ra biến đổi tốt nhất để thêm vào quần thể mô hình.

- **Reinfocement Learning:**

i Trong trường hợp của Học tăng cường, "đề xuất di chuyển" hoạt động tương tự như trong giai đoạn khởi động.

ii Thay vì chỉ thưởng cho bộ điều khiển trước khi bắt đầu quá trình tìm kiếm, thuật toán xen kẽ giữa thưởng dựa trên độ chính xác và thưởng không tốn kém (dựa trên chỉ số proxy) để hướng dẫn bộ điều khiển chọn lựa các kiến trúc mô hình có khả năng thành công cao.