ĐỀ TÀI LUẬN VĂN

Thuật toán tiến hóa đa nhiệm tự thích ứng

Giáo viên hướng dẫn
Ký và ghi rõ họ tên

Lời cảm ơn

Trước hết, tôi xin được cảm ơn giáo viên hướng dẫn là PGS.TS Huỳnh Thị Thanh Bình. Cô đã luôn luôn giúp đỡ và tạo mọi điều kiện tốt nhất cho tôi hoàn thành khối lượng công việc nghiên cứu của chương trình thạc sĩ khoa học. Tôi cũng xin được cảm ơn học viên cao học Tạ Bảo Thắng và sinh viên Lê Văn Cường, hai đồng tác giả của bài báo khoa học tổng kết đề tài nghiên cứu này.

Trong quá trình học, tôi cũng rất may mắn khi được trường cho phép đi thực tập nghiên cứu tại Viện Tin học Quốc gia Nhật Bản (NII). Tôi cũng xin được cảm ơn Giáo sư Yusheng Ji, giáo viên hướng dẫn tôi tại NII đã bao dung và giúp đỡ tôi trong kỳ thực tập. Cảm ơn nghiên cứu sinh Lê Văn An, người đã làm việc trực tiếp cùng tôi hằng ngày ở NII, tận tay chỉ việc và giúp tôi nâng cao kỹ năng nghiên cứu.

Cảm ơn các bạn thân tôi, Hải, đã cung cấp máy tính để tôi chạy thực nghiệm.

Cảm ơn bố mẹ tôi và bạn gái tôi đã động viên tinh thần tôi để tôi hoàn thành những mục tiêu đã đề ra.

Tóm tắt nội dung luận văn

Thuật toán tiến hóa đa tác vụ hiện tại đang nhận được sự quan tâm của cộng đồng nghiên cứu về tính tóan tiến hóa. Thuật toán được lấy cảm hứng từ khả năng làm nhiều việc cùng một lúc của con người chúng ta. Phỏng lại hiện tượng trên, thuật toán có thể tối ưu nhiều bài toán khác nhau, ngầm chia sẻ thông tin trong lúc giải để kết quả tối ưu từng tác vụ thành phần được cải thiện so với giải riêng từng tác vụ. Thuật toán đã được áp dụng trên nhiều ứng dụng khác nhau từ tối ưu các bài toán định tuyến, phân cụm đồ thị khó đến tối ưu các mạng nơ ron nhỏ. Tiến hóa đa nhiệm được hướng tới triển khai trên môi trường tính toán đám mây với số lượng lớn người dùng gửi yêu cầu giải bài toán của họ về hệ thống để máy chủ trung tâm giải các bài toán của nhiều người dùng khác nhau cùng nhau.

Tuy nhiên, các nghiên cứu về tiền hóa đa nhiệm thường chỉ dừng lại ở tối ưu hai đến ba tác vụ cùng một lúc. Để đạt được yêu cầu triển khai thực tế như trên, thuật toán cần phải đáp ứng được việc giải một lượng lớn các tác vụ khác nhau. Nghiên cứu này đề xuất một thuật toán tiến hóa đa nhiệm có thể xử lý được một lượng lớn các tác vụ, gọi là Many-task Multi-armed Bandit Evolutionary Algorithm (Ma^{2}BEA). Ngoài ra, để giảm thiểu số lượng tham số, thuật toán cần tự động thích nghi với các tập bài toán đầu vào khác nhau, dựa trên nguồn dữ liệu dồi dào sinh ra từ quá trình tối ưu. Ý tưởng chính của Ma^{2}BEA là mô hình hóa lại cách ghép cặp các tác vụ tương đồng để trao đổi thông tin với nhau trên mô hình Multi-Armed Bandits. Cách ghép cặp này là tự động và chỉ dựa trên dữ liệu về giá trị hàm mục tiêu của các tác vụ thành phần, sinh ra trong quá trình tối ưu. Một khung thuật toán mới cho tiến hóa đa nhiệm, phù hợp hơn với trường hợp số lượng tác vụ rất lớn cũng được đề xuất. Thuật toán được đánh giá và so sánh với các thuật toán giải số lượng tác vụ lớn mới nhất, trên các bộ dữ liệu thử nghiệm chung. Ngoài ra, thuật toán cũng được thử nghiệm trên ứng dụng tối ưu mạng nơ-ron cho học tăng cường.

HỌC VIÊN
Ký và ghi rõ họ tên

Tóm tắt

Thuật toán tiến hóa đa tác vụ hiện tại đang nhận được sự quan tâm của cộng đồng nghiên cứu về tính tóan tiến hóa. Thuật toán được lấy cảm hứng từ khả năng làm nhiều việc cùng một lúc của con người chúng ta. Phỏng lại hiện tượng trên, thuật toán có thể tối ưu nhiều bài toán khác nhau, ngầm chia sẻ thông tin trong lúc giải để kết quả tối ưu từng tác vụ thành phần được cải thiện so với giải riêng từng tác vụ. Thuật toán đã được áp dụng trên nhiều ứng dụng khác nhau từ tối ưu các bài toán định tuyến, phân cụm đồ thị khó đến tối ưu các mạng nơ ron nhỏ. Tiến hóa đa nhiệm được hướng tới triển khai trên môi trường tính toán đám mây với số lượng lớn người dùng gửi yêu cầu giải bài toán của họ về hệ thống để máy chủ trung tâm giải các bài toán của nhiều người dùng khác nhau cùng nhau.

Tuy nhiên, các nghiên cứu về tiền hóa đa nhiệm thường chỉ dừng lại ở tối ưu hai đến ba tác vụ cùng một lúc. Để đạt được yêu cầu triển khai thực tế như trên, thuật toán cần phải đáp ứng được việc giải một lượng lớn các tác vụ khác nhau. Nghiên cứu này đề xuất một thuật toán tiến hóa đa nhiệm có thể xử lý được một lượng lớn các tác vụ, gọi là Ma^{2}BEA. Ngoài ra, để giảm thiểu số lượng tham số, thuật toán cần tự động thích nghi với các tập bài toán đầu vào khác nhau, dựa trên nguồn dữ liệu dồi dào sinh ra từ quá trình tối ưu. Ý tưởng chính của Ma^{2}BEA là mô hình hóa lại cách ghép cặp các tác vụ tương đồng để trao đổi thông tin với nhau trên mô hình Multi-Armed Bandits. Cách ghép cặp này là tự động và chỉ dựa trên dữ liệu về giá trị hàm mục tiêu của các tác vụ thành phần, sinh ra trong quá trình tối ưu. Một khung thuật toán mới cho tiến hóa đa nhiệm, phù hợp hơn với trường hợp số lượng tác vụ rất lớn cũng được đề xuất. Thuật toán được đánh giá và so sánh với các thuật toán giải số lượng tác vụ lớn mới nhất, trên các bộ dữ liệu thử nghiệm chung. Ngoài ra, thuật toán cũng được thử nghiệm trên ứng dụng tối ưu mạng nơ-ron cho học tăng cường.

Luận văn được trình bày với các phần như sau:

- **Chương 1** giới thiệu khái quát những khái niệm, nguồn gốc, lý do nghiên cứu về Multifactorial Evolutionary Algorithm (MFEA), đồng thời giới thiệu các ứng dụng của thuật toán trong việc giải quyết nhiều bài toán đồng thời trong nghiên cứu cơ bản và trong thực tế.

- **Chương 2** liệt kê các nghiên cứu liên quan đến nghiên cứu này, cụ thể là các biến thể của thuật toán MFEA có khả năng giải số lượng lớn các bài toán cùng nhau.

- **Chương 3** trình bày vắn tắt những kiến thức cơ sở dẫn dắt đến thuật toán đề xuất tại nghiên cứu này.

- **Chương 4** mô tả đóng góp chính là thuật toán đề xuất.

- **Chương 5** trình bày các thực nghiệm so sánh thuật toán đề xuất với các biến thể khác của thuật toán MFEA trên nhiều bài toán tối ưu khác nhau.

Danh sách viết tắt

Viết tắt	**Tên đầy đủ tiếng Anh**
EA	Evolutionary Algorithm
DE	Differential Evolutionary algorithm
GP	Genetic Programming
ES	Evolution Strategies
PSO	Particle Swarm Optimization
SBX	Simulated Binary Crossover
PM	Polynomial Mutation
GM	Gausian Mutation
rmp	Random Mating Probability
MFEA	Multifactorial Evolutionary Algorithm
MFEA-II	Multifactorial Evolutionary Algorithm - II
GMFEA	Group-based Multifactorial Evolutionary Algorithm
MaTGA	Many-Task archive-based Genetic Algorithm
SBSGA	Evolutionary many-tasking based on Symbiosis in Biocoenosis
GA	Genetic Algorithm
MAB	Multi Armed Bandit
MDP	Markov Decision Process
Ma^{2}BEA	Many-task Multi-armed Bandit Evolutionary Algorithm
UCB	Upper Confident Bound
KL-UCB	Kullback-Leibler UCB
MTO	Multi Task Optimization
MaTO	Many Task Optimization
MaTO-10	Many-task Optimization (10 tasks)
MaTO-50	Many-task Optimization (50 tasks)

Viết tắt	**Giải nghĩa tiếng Việt**
EA	Thuật toán tiến hóa
DE	Thuật toán tiến hóa sai phân
GP	Thuật toán lập trình tiến hóa
ES	Thuật toán thiến lược tiến hóa
PSO	Thuật toán tối ưu bầy đàn
SBX	Phép lai ghép số thực giả lập lai ghép nhị phân
PM	Phép đột biến với phân phối Polynomial
GM	Phép đột biến với phân phối chuẩn
rmp	Xác suất lai ghép khác tác vụ
MFEA	Thuật toán tiến hóa đa nhiệm
MFEA-II	Thuật toán tiến hóa đa nhiệm - Phiên bản 2
GMFEA	Thuật toán tiến hóa đa nhiệm dựa vào phân nhóm
MaTGA	Thuật toán tiến hóa rất nhiều tác vụ dựa vào lưu trữ lịch sử tiến hóa
SBSGA	Thuật toán tiến hóa rất nhiều tác vụ lấy cảm hứng từ cơ chế cộng sinh
GA	Giải thuật di truyền
MAB	Mô hình Multi Armed Bandit
MDP	Quá trình ra quyết định Markov
Ma^{2}BEA	Thuật toán tiến hóa rất nhiều tác vụ dựa trên mô hình Multi Armed Bandit
UCB	Upper Confident Bound
KL-UCB	Kullback-Leibler Upper Confident Bound
MTO	Mô hình tối ưu đa tác vụ
MaTO	Mô hình tối ưu rất nhiều tác vụ
MaTO-10	Bộ dữ liệu tối ưu 10 tác vụ
MaTO-50	Bộ dữ liệu tối ưu 50 tác vụ

Danh sách bảng

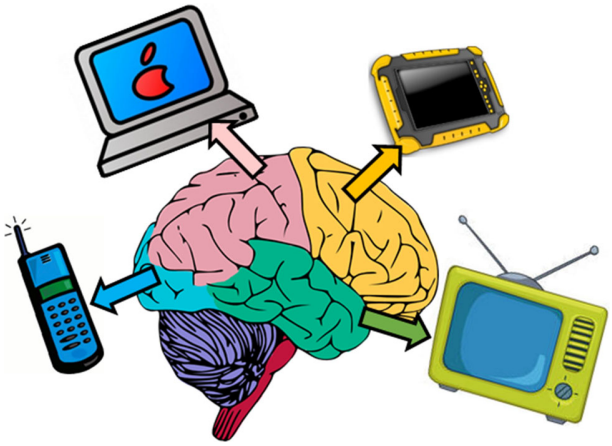
Danh sách hình vẽ

Mục lục

Chương 1

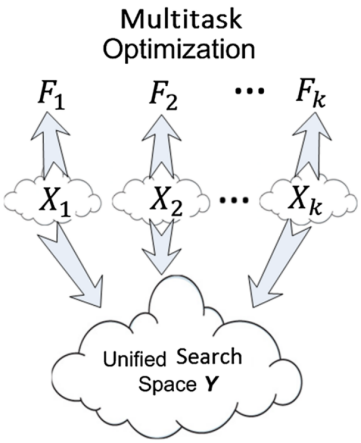
Giới thiệu

Bài toán tối ưu đa tác vụ và thuật toán tiến hóa đa nhiệm hiện tại đang nhận được sự quan tâm của cộng đồng nghiên cứu về tính toán tiến hóa. Những ý tưởng này lấy cảm hứng từ khả năng làm nhiều việc cùng một lúc của con người chúng ta. Một ví dụ điển hình là khả năng tiếp thu thông tin từ nhiều kênh khác nhau như ví dụ trong Hình 1.1. Con người phải có kỹ năng làm nhiều việc cùng một lúc để thích nghi trong thời đại bùng nổ thông tin hiện nay. Mà các mô hình, thuật toán hoặc hệ thống thông minh thường mô phỏng lại cách làm việc của con người. Vậy nên, việc đa nhiệm hóa các thuật toán trở thành một hướng nghiên cứu quan trọng là một điều dễ hiểu. Ngoài ra, nghiên cứu trong ngành tâm lý học đã chỉ ra rằng, nếu con người gộp các tác vụ liên quan mật thiết với nhau để làm cùng nhau tại một thời điểm, thì năng suất làm việc của người đó sẽ được cải thiện đáng kể [1]. Tương tự, các nghiên cứu về tiến hóa đa nhiệm chủ yếu tập trung vào việc trao đổi thông tin giữa các tác vụ tối ưu tương đồng để kết quả tối ưu cuối cùng của từng tác vụ được cải thiện tốt hơn.



Hình 1.1: Ví dụ về cơ chế đa nhiệm của não người khi một người bình thường tương tác với nhiều thiết bị thông tin và truyền thông cùng một lúc [2].

Tiến hóa đa nhiệm được định nghĩa là một khung thuật toán tối ưu cho nhiều bài toán khác nhau, trong đó, các bài toán thành phần là độc lập lẫn nhau (Hình 1.2). Điểm đặc biệt của khung thuật toán này là việc giải một bài toán có thể là một yếu tố hỗ trợ giúp cho việc giải một bài toán khác tốt hơn. Một ứng dụng nổi bật nhất của khung thuật toán tiến hóa đa nhiệm này là ứng dụng trong điện toán đám mây. Ví dụ, ta có thể có một hệ thống tính toán đám mây phục vụ việc giải quyết các bài toán tối ưu cho người dùng. Tại một thời điểm, nhiều người dùng khác nhau có thể yêu cầu máy chủ trung tâm giải bài toán của họ. Hệ thống này có thể cài đặt thuật toán tối ưu đa nhiệm để giải quyết các bài toán này cùng nhau, ngầm chia sẻ tri tức trong quá trình tối ưu để cải thiện lời giải gửi



Hình 1.2: Ví dụ về mô hình tối ưu đa nhiệm [2].

về cho người dùng. Ứng dụng cụ thể của tiến hóa đa nhiệm có thể kể đến là tối ưu nhiều bài toán định tuyến NP-khó trên đồ thị, mà có đồ thị đầu vào khác nhau nhưng cùng hàm mục tiêu [3, 4, 5, 6, 7, 8, 9]. Tiến hóa đa nhiệm cũng đã được ứng dụng trong việc giải quyết việc giải nhiều bài toán tăng tuổi thọ của mạng cảm biến không dây cùng nhau [10]. Ngoài ra, khung thuật toán này cũng đã được áp dụng để tối ưu các mạng nơ ron có cấu trúc gần giống với nhau [11]. Tại các ứng dụng trên, người dùng, tức là các thực thể ở các vị trí khác nhau trong không gian mạng gửi bài toán tối ưu tại vị trí đó về một máy chủ trung tâm, nơi tất cả các bài toán được giải quyết chung với hy vọng kết quả của việc giải chung sẽ tốt hơn giải từng bài toán riêng lẻ. Tuy tiến hóa đa nhiệm đã được nghiên cứu rộng rãi cả về mặt cải tiến giải thuật lẫn đưa vào ứng dụng, số lượng tác vụ mà các thuật toán tiến hóa đa nhiệm giải chỉ dừng lại ở hai hoặc ba tác vụ. Nhưng ngược lại, ứng dụng mà tiến hóa đa nhiệm hướng tới như giải các bài toán tối ưu trên điệm toán đám mây lại có số lượng tác vụ lớn hơn thế rất nhiều. Người dùng của các hệ thống điện toán đám mây phân tán thường lên tới hàng trăm hoặc hàng ngàn người trong một thời điểm. Vậy nên thuật toán tiến hóa đa nhiệm khi áp dụng ra thực tế cần có có khả năng giải quyết rất nhiều bài toán cùng một lúc. Khi số lượng tác vụ tăng lên, có rất nhiều vấn đề phát sinh nảy ra. Ví dụ, thuật toán không thể biết được nên ghép tác vụ nào với tác vụ nào để trao đổi tri thức trong quá trình tối ưu. Nếu có ít tác vụ thì việc lựa chọn ghép cặp một cách tự động sẽ dễ dàng hơn, và đã có nhiều nghiên cứu đào sâu vào giải quyết vấn đề này [12, 13]. Tại thời điểm luận văn này được thực hiện, chưa có nhiều các nghiên cứu chứng minh tính khả thi của thuật toán tối ưu đa nhiệm trên số lượng tác vụ lớn. Trong số 87 công trình đã công bố về tiến hóa đa nhiệm tính đến đầu năm 2021, chỉ có 3 công trình đề xuất thuật toán giải quyết vấn đề này [14, 15, 16]. Cộng đồng nghiên cứu về tiến hóa đa nhiệm hiện nay cũng đã chuyển dần sang nghiên cứu theo hướng này. Minh chứng cụ thể là đề thi các cuộc thi về tiến hóa đa nhiệm trên các hội thảo đầu ngành trước năm 2019 đều chỉ có các bộ dữ liệu gồm hai tác vụ. Các cuộc thi sau năm 2019 đã có thêm các đề thi với số lượng tác vụ là 50 (`http://www.bdsc.site/websites/MTO/index.html`) Vì vậy, tôi đi sâu vào hướng nhỏ này, mong muốn cải thiện được tiến hóa đa nhiệm trên số lượng tác vụ lớn trên các bộ dữ liệu thử nghiệm chuẩn được chia sẻ dùng chung với các nhóm nghiên cứu khác.

Để tiến tới thiết kế một thuật toán tối ưu xử lý được số lượng lớn các tác vụ ta cần phải quan tâm đến vấn đề sau. Thứ nhất, ta không thể thiết kế một thuật toán có nhiều tham số, vì không thể giả định lúc nào người dùng cũng là chuyên gia để tinh chỉnh được các tham số của thuật toán [17]. Cụ thể, ở trường hợp thiết kế thuật toán tối ưu đa nhiệm, các tham số của thuật toán thường nằm ở thành phần lựa chọn ghép cặp xem tác

vụ nào được kết hợp với tác vụ nào. Người dùng không thể tự biết được là các tác vụ đưa vào thuật toán liên quan đến nhau như thế nào để điều chỉnh các tham số trên, vì giả định các bài toán mà tiến hóa đa nhiệm giải đều là các bài toán tối ưu hộp đen khó. Thứ hai, thuật toán tiến hóa đa nhiệm với số lượng lớn tác vụ sinh ra một lượng dữ liệu dồi dào. Qua nhiều thế hệ, dữ liệu như quần thể hay giá trị hàm mục tiêu của rất nhiều tác vụ có thể được lưu lại để phân tích và đưa ra mối quan hệ của các tác vụ một cách tự động. Các kỹ thuật phân tích dữ liệu, học máy có thể được áp dụng để cải thiện thuật toán tiến hóa đa nhiệm. Hơn nữa, bằng việc phân tích dữ liệu một cách tự động như vậy, thuật toán tối ưu này sẽ phụ thuộc vào dữ liệu (data-driven optimization) và giảm thiểu số lượng tham số cho người dùng thuật toán. Trong nghiên cứu này, tôi đề xuất một cách đơn giản nhất đê áp dụng thuật toán học máy giúp thuật toán tiến hóa đa nhiệm tự thích nghi với nhiều trường hợp bài toán đầu vào khác nhau.

Cụ thể là, tôi đề xuất thuật toán Many-task Multi-armed Bandit Evolutionary Algorithm (Ma^{2}BEA) để giải bài toán tối ưu đa tác vụ với số lượng tác vụ lớn. Ở đây, tôi đề xuất một cơ chế ghép cặp tác vụ mới sử dụng mô hình Multi-Armed Bandits. Multi-Armed Bandits là một mô hình mô phỏng lại việc đưa ra quyết định của một tác tử trên một môi trường không biết trước. Tại mỗi vòng, tác tử phải lựa chọn một trong nhiều hành động, và môi trường trả lại cho tác tử một phần thưởng. Thuật toán giải mô hình này phải tìm một chính sách ra quyết định mà tác tử nhận về tổng phần thưởng nhiều nhất. Tương tự, tôi mô hình hóa lại việc lựa chọn tác vụ để ghép cặp trên mô hình Multi-Armed Bandits nếu trên, và ứng dụng phương trình Kullback–Leibler UCB (KL-UCB) để lựa chọn ghép các tác vụ với nhau sao cho giá trị hàm mục tiêu của các tác vụ ghép với nhau được cải thiện nhiều nhất. Tóm lại, đóng góp chính của nghiên cứu này bao gồm:

- Đề xuất thuật toán tiến hóa đa nhiệm mới dựa trên mô hình Multi-Armed Bandits
- Thiết kế một khung mới cho thuật toán tiến hóa đa nhiệm mà phù hợp hơn cho việc tối ưu số lượng lớn các tác vụ
- Đánh giá tính hiệu quả của thuật toán trên nhiều bộ dữ liệu đánh giá chuẩn trong tiến hóa đa tác vụ với số tác vụ lớn, cũng như đánh giá trên tối ưu mạng nơ ron cho học tăng cường.

Chương 2

Cơ sở lý thuyết

Chương này mô tả khái quát về các khái niệm cơ bản liên quan trực tiếp đến thuật toán đề xuất lại luận văn. Trước hết, các khái niệm chung và thuật toán tối ưu hóa đa nhiệm phiên bản gốc được mô tả chi tiết. Sau đó, các kiến thức về mô hình Multi-Armed Bandits được trình bày để dẫn dắt đến việc kết hợp mô hình này vào khung giải thuật tiến hóa đa nhiệm tại chương tiếp theo.

2.1 Thuật toán tiến hóa đa tác vụ

2.1.1 Giới thiệu chung

Một trong những nguyên nhân dẫn đến ý tưởng về các thuật toán tối ưu đa nhiệm là sự hiện diện của nhiều bài toán tối ưu tương đồng trong thực tế. Thật vậy, các bài toán tối ưu hộp đen thường có sự tương đồng cao khi chúng được mô hình hóa cho các vấn đề ở cùng một lĩnh vực. Một ví dụ của các bài toán tối ưu tương đồng là các bài toán đồ thị với cùng một mục tiêu là định tuyến nhưng khác về đồ thị đầu vào (bài toán Miminum Routing Cost Tree [4]). Từ đó, ý tưởng về việc chia sẻ thông tin tối ưu giữa các bài toán khác nhau được hình thành [2]. Việc chia sẻ thông tin giữa các tác vụ tối ưu khác nhau này có thể giúp cho bài toán được giải quyết nhanh chóng hơn, đạt kết quả tốt hơn và tiết kiệm tài nguyên tính toán hơn [13]. Đây cũng là một trong nhưng mục tiêu lâu dài mà cộng đồng nghiên cứu trí tuệ nhân tạo hướng tới [18].

Các nghiên cứu gần đây về tối ưu đa nhiệm thường tập trung vào hai hướng chính: tối ưu đa nhiệm Bayes [19] và tiến hóa đa nhiệm [13]. Tối ưu đa nhiệm Bayes thường được dùng trong bài toán tối ưu tham số nhiều mô hình học máy hoặc học sâu tự động cùng lúc [19]. Phương pháp này sử dụng quá trình Gaussian để mô phỏng lại không gian của từng tác vụ tối ưu, từ đó tự động đo được sự tương đồng giữa các tác vụ, từ đó chuyển giao các nghiệm giữa các tác vụ tương đồng để cho kết quả tối ưu tốt hơn. Tuy nhiên, nhược điểm của tối ưu Bayes là phương pháp này thường chỉ tốt cho các bài toán với số chiều của biến tối ưu nhỏ (cụ thể là khoảng dưới 20 chiều) [20]. Ngược lại, thuật toán tiến hóa Evolutionary Algorithm (EA) có thể giải xấp xỉ được các bài toán tối ưu hộp đen với số chiều lớn hơn. Ngoài biểu diễn số thực, EA cũng có thể giải được nhiều bài toán với biểu diễn khác [4, 3]. Vậy nên, thuật toán tiến hóa đa nhiệm Multifactorial Evolutionary Algorithm (MFEA) với lõi là EA linh động hơn so với tối ưu đa nhiệm Bayes.

Nhìn chung, bài toán tối ưu đa nhiệm tổng quát được phát biểu như sau. Cho K bài toán tối ưu khác nhau cần giải quyết. Ta có thể giả định tất cả K bài toán đều là bài toán tối thiểu hóa, mà không làm mất đi tính tổng quát. Tác vụ thứ k được gọi là T_k. T_k có không gian tìm kiếm là $\mathcal{X}_k$ và có hàm mục tiêu (hay trong tính toán tiến hóa hay gọi là hàm thích nghi) $f_k : \mathcal{X} \to \mathbb{R}$. Ngoài ra, tác vụ T_k có thể bị ràng buộc bởi một số

hàm ràng buộc khác. Tại [21], tác giả định nghĩa tiến hóa đa tác vụ là một khung thuật toán tìm kiếm tiến hóa để giải bài toán tối ưu đa nhiệm. Điểm đặc biệt của khung thuật toán này sử dụng quần thể của các tác vụ khác nhau để gây ảnh hưởng lẫn nhau và tìm được tập hợp nghiệm $\{x_1^*, x_2^*, \ldots, x_{K-1}^*, x_K^*\} = argmin\{f_1(x), f_2(x), \ldots, f_{K-1}(x), f_K(x)\}$ với x_k^* là nghiệm tối ưu toàn cục của T_k. Tại khung thuật toán này, mỗi hàm mục tiêu f_k lại là một yếu tố gây ảnh hưởng, thường ta đều mong muốn là ảnh hưởng tốt, đến quá trình tiến hóa của một tác vụ khác. Hiện nay, bài toán tối ưu đa nhiệm được chia làm hai dạng, Multi Task Optimization (MTO) với số lượng tác vụ nhỏ từ 2 đến 3 tác vụ. Loại thứ hai là tối ưu đa nhiệm với rất nhiều tác vụ Many Task Optimization (MaTO) và số lượng tác vụ thường lớn hơn 10.

2.1.2 Không gian biểu diễn chung

Ở phần này, biểu diễn nghiệm của MFEA sẽ được mô tả. MFEA giải nhiều bài toán khác nhau cùng một lúc, cải thiện kết quả của các tác vụ tối ưu đó bằng cách chia sẻ thông tin giữa chúng. Để việc chia sẻ thông tin được thuận lợi, biểu diễn của các bài toán khác nhau cần được biến đổi lại để đưa vào không gian chung. Từ đó, các tác vụ trong tiến hóa đa nhiệm sẽ chia sẻ thông tin với nhau bằng cách thực hiện lai ghép với tác vụ khác trên không gian chung. Cụ thể, cho K tác vụ, các tác vụ có số chiều lần lượt là $\{D_1, D_2, \ldots, D_K\}$. Biểu diễn chung được định nghĩa là không gian có số chiều là $D_{unified} = max\{D_1, D_2, \ldots, D_K\}$. Việc tối ưu sử dụng các toán tử tiến hóa sẽ được thực hiện trên không gian có chiều $D_{unified}$. Từ đó, việc lai ghép cùng tác vụ và khác tác vụ đều được thực hiện một cách đồng nhất và dễ dàng. Khi đánh giá hàm mục tiêu cho từng cá thể, cá thể thuộc tác vụ T_k sẽ được được giải mã về không gian riêng có số chiều D_k, và thực hiện đánh giá trên hàm mục tiêu $f_k(.)$ như bình thường.

Với các bài toán tối ưu số thực, không gian tìm kiếm tại mỗi chiều thường không được chuẩn hóa. Giả sử ta có biến tối ưu $x^{(k)} \in \mathbb{R}^{D_k}, x^{(k')} \in \mathbb{R}^{D_{k'}}$, thì $min(x^{(k)}) \neq min(x^{(k')})$ và $max(x^{(k)}) \neq max(x^{(k')})$. Vậy nên MFEA quy ước không gian biểu diễn chung của tất các bài toán tối ưu số thực được giới hạn ở khoảng $[0, 1]^{D_{unified}}$. Tương tự như trên, các cá thể được ánh xạ từ nhiều không gian khác nhau về khoảng $[0, 1]^{D_{unified}}$ để dễ dàng trao đổi tri thức với các cá thể ở tác vụ khác. Khi đánh giá, các cá thể lại được giải mã ngược về không gian của từng tác vụ.

2.1.3 Giải thuật MFEA

Tại phần này, các bước thực hiện của thuật toán MFEA phiên bản gốc sẽ được trình bày chi tiết. MFEA khởi tạo và chỉ sử dụng một quần thể duy nhất là quần thể P trên không gian biểu diễn chung để tối ưu K tác vụ $T_1, T_2, \ldots, T_K$ cùng lúc. Mỗi quần thể con tương ứng với tập hợp các cá thể thuộc tác vụ T_k được gọi là P^k. Một số khái niệm khác hỗ trợ cho việc mô tả cụ thể MFEA bao gồm.

Định nghĩa 1 (Skill factor)**.** Skill factor τ_i *của cá thể* i^{th} *là vị trí của tác vụ trong* K *tác vụ, mà cá thể* i *thuộc về. Nói cách khác, cá thể* i^{th} *nằm trong tập hợp* P^{τ_i}*.*

Định nghĩa 2 (Scalar Fitness)**.** Scalar fitness *là giá trị hàm đánh giá được chuẩn hóa lại để không phụ thuộc vào khoảng giá trị của hàm đánh giá của các tác vụ. Cụ thể là* scalar fitness *của cá thể* i^{th} *được tính bằng công thức* $\varphi = \frac{1}{r_{\tau_i}^i}$*, trong đó* $r_{\tau_i}^i$ *là thứ hạng (xếp từ tốt nhất đến tối nhất về mặt giá trị hàm mục tiêu) của cá thể* i^{th} *trong tập hợp* P^{τ_i}*. Bất kể dải giá trị của các hàm đánh giá khác nhau,* scalar fitness *của các hàm đánh giá sẽ có dải giá trị như nhau. Cá thể càng tốt thì thứ hạng* $r_{\tau_i}^i$ *càng nhỏ và* scalar fitness *càng lớn.*

Giải thuật 1 tổng kết ngắn gọn các bước trong thuật toán MFEA. Thủ tục của giải thuật bắt đầu bằng việc khởi tạo giá trị ngẫu nhiên cho một quần thể P gồm $N \times K$ cá thể trên không gian biểu diễn chung. Các tác vụ được phân bố tài nguyên đều nhau, với mỗi tác vụ có N cá thể. Trong quá trình tiến hóa, các cá thể trong cùng một tác vụ (giải cùng một bài toán) được trao đổi vật chất di truyền lẫn nhau bằng phép *lai ghép cùng tác vụ*. Điểm đặc biệt của MFEA là ngoài *lai ghép cùng tác vụ*, MFEA cho phép các tác vụ có thể trao đổi vất chất di truyền lẫn nhau bằng cách *lai ghép khác tác vụ* giữa hai cá thể thuộc tác vụ khác nhau.

Mức độ trao đổi vật chất di truyền giữa các tác vụ khác nhau được điều chỉnh bằng một tham số gọi là *random mating probability* (xác suất lai ghép khác tác vụ), gọi tắt là *rmp*. *rmp* được sử dụng như trong Dòng 14, Giải thuật 1. Nếu hai cá thể được chọn ngẫu nhiên trong quần thể chung P thuộc cùng tác vụ, chúng sẽ luôn luôn được *lai ghép cùng tác vụ*. Tuy nhiên, nếu ta chọn ngẫu nhiên ra hai cá thể thuộc khác tác vụ, chúng sẽ chỉ được *lai ghép khác tác vụ* với xác suất *rmp* ($rand \sim U(0,1)$ mà $rand < rmp$). Trong quá trình *lai ghép khác tác vụ*, một phần của nghiệm của tác vụ này sẽ được gán vào một phần nghiệm của tác vụ khác. Đồng thời, các cá thể con sinh ra sau *lai ghép khác tác vụ* sẽ được gán *skill factor* của cha mẹ, và được đánh giá trên hàm đánh giá của bài toán của một trong cá thể cha hoặc mẹ. Nhờ vậy, thông tin được ngầm trao đổi giữa các tác vụ, làm ảnh hưởng đến kết quả tối ưu của từng tác vụ thành phần trong tối ưu đa nhiệm. Nếu các tác vụ có cùng cực trị toàn cục x^* hoặc cùng hướng đi về cực trị toàn cục x^* thì việc trao đổi này sẽ giúp các tác vụ thành phần đi đến điểm có giá trị hàm đánh giá tốt hơn nhanh hơn. Người dùng thuật toán tự lựa chọn tham số $rmp \in [0,1]$ sao cho phù hợp. Giả sử như các tác vụ trong bộ K tác vụ liên quan mật thiết đến nhau thì nên chọn *rmp* gần với 1 để việc trao đổi thông tin khác tác vụ nhiều hơn, giúp kết quả tốt hơn. Tương tự, nếu biết trước mức độ liên quan giữa các tác vụ là không quá nhiều, người dùng có thể điều chỉnh *rmp* sát với 0 để lượng thông tin trao đổi khác tác vụ giảm xuống.

Tuy nhiên, cơ chế trao đổi thông tin khác tác vụ của thuật toán MFEA phiên bản gốc này có một nhược điểm, đó là việc các tác vụ khác nhau được ghép ngẫu nhiên. Giả sử như các tác vụ trong K tác vụ đều hỗ trợ lẫn nhau thì việc ghép gặp để *lai ghép khác tác vụ* như trên sẽ luôn có ích. Tuy nhiên nếu số lượng tác vụ K lớn, có nhiều tác vụ không hỗ trợ lẫn nhau thì việc ghép ngẫu nhiên như trên có thể làm thuật toán MFEA chạy tồi hơn cả thuật toán tối ưu đơn tác vụ EA. Mục tiếp theo sẽ trình bày một phương pháp học tự thích ứng đơn giản, mô hình Multi-Armed Bandits, một mô hình có tiềm năng giúp cho thuật toán MFEA lựa chọn tác vụ để trao đổi tốt hơn.

Algorithm 1 Khung giả mã của MFEA

Lấy ngẫu nhiên $K \cdot N$ cá thể để tạo quần thể P;
for mỗi cá thể $p_i \in P$ **do**
    Gán *skill factor* $\tau_i = mod(i, K) + 1$, với K là số lượng tác vụ;
    Đánh giá p_i trên tác vụ τ_i;
end for
while thỏa mãn điều kiện kết thúc **do**
    $P \leftarrow$ Chọn $K \cdot N/2$ cá thể tốt nhất từ P;
    Tạo quần thể con $P_c = \emptyset$;
    while số lượng con cho mỗi tác vụ $< N$ **do**
        Lấy ngẫu nhiên hai cá thể cha mẹ p_a và p_b từ P;
        if $\tau_a == \tau_b$ **then**
            $[c_a, c_b] \leftarrow$ *Lai ghép cùng tác vụ* giữa p_a và p_b;
            Gán *skill factor* τ_a cho c_a và c_b
        else if $rand \leq rmp$ **then**;
            $[c_a, c_b] \leftarrow$ *Lai ghép khác tác vụ* giữa p_a và p_b;;
            Gãn ngâu nhiên *skill factor* τ_a or τ_b cho từng con sinh ra;
        else
            $[c_a] \leftarrow$ đột biến p_a;
            Gán *skill factor* τ_a cho c_a ;
            $[c_b] \leftarrow$ đột biến of p_b;
            Gán *skill factor* τ_b cho c_b ;
        end if
        Đánh giá cá thể con $[c_a, c_b]$ trên *skill factor* đã gán;
        $P_c = P_c \cup [c_a, c_b]$;
    end while
    $P = P_c \cup P$;
end while

Algorithm 2 Mô hình Multi-armed bandits

Cho: K lựa chọn, T vòng.
for vòng thứ $t \in \{1, \ldots, T\}$ **do**
    Chọn lựa chọn a_t;
    Nhận về phần thưởng $r_t \in [0, 1]$ cho lựa chọn a_t;
end for

2.2 Mô hình Multi-Armed Bandits

Multi Armed Bandit (MAB) là một lĩnh vực nghiên cứu về trí tuệ nhân tạo và được áp dụng trong nhiều bài toán, nhiều ngành đa dạng. Ý tưởng của MAB bắt nguồn từ năm 1933 [22], và được đẩy mạnh nghiên cứu cả về lý thuyết cơ bản lẫn ứng dụng trong 15 năm gần đây [23]. Mục này giới thiệu tổng quan về cấc thành phần của MAB và hướng ứng dụng của MAB vào một bài toán cụ thể.

Trước tiên, MAB là một mô hình mô phỏng lại việc đưa ra quyết định theo thời gian dưới điều kiện không chắc chắn. Một số ví dụ của bài toán đưa quyết định mà MAB có thể được áp dụng để giải quyết bao gồm:

- Khi một người dùng vào một website, website chọn tiêu đề của một hoặc nhiều tin tức hiển thị. Mục tiêu của website là chọn các tiêu đề của các bài báo mà người dùng nhấn vào xem càng nhiều càng tốt.
- Một cửa hàng điện tử (ví dụ: AppStore, GooglePlay) rao bán các ứng dụng hoặc bài hát. Khi một người dùng tới trang chủ cửa hàng, cửa hàng đưa ra giá của các ứng dụng hoặc bài hát cho người dùng. Người dùng có thể mua hoặc rời cửa hàng. Mục tiêu của cửa hàng là tối ưu lợi nhuận bán ứng dụng và bài hát.
- Mỗi sáng, một nhà đầu tư cá nhân cần chọn một mã cổ phiếu để đầu tư, và đầu tư $50,000$ đồng vào nó. Cuối mỗi ngày, nhà đầu tư theo dõi được chênh lệch. Mục tiêu của nhà đầu tư cá nhân là tối đa hóa lượng tài sản của bản thân.

Mô hình MAB hợp nhất nhiều bài toán đưa quyết định thực tế nêu trên. Trong phiên bản đơn giản nhất của MAB, tại từng thời điểm, tác tử có thể chọn một trong K lựa chọn khác nhau. Thuật toán giải mô hình MAB sẽ được lựa chọn T vòng. Ở mỗi vòng, ngoài việc lựa chọn, thuật toán giải MAB thu thập phần thưởng (reward) cho lựa chọn mà thuật toán đã chọn. Phần thưởng này được sinh từ một phân phối xác suất ngẫu nhiên. Ở đây, thuật toán không biết thông tin nào về phân phối xác suất của phần thưởng.

Giải quyết mô hình MAB khó ở chỗ cần phải cân bằng giữa việc *khai thác* và *khám phá*. Cụ thể là, ở mô hình MAB cơ bản như trên, ở mỗi một vòng lặp, thuật toán giải MAB chỉ biết được phần thưởng khi cho hành động nó đã lựa chọn. Thuật toán không thể biết được phần thưởng của hành động không được chọn. Vậy nên, thuật toán giải MAB phải *khám phá* bằng cách thử càng nhiều các lựa chọn càng tốt. Ngoài ra, nếu thuật toán tập trung quá nhiều vào việc thử các lựa chọn thì tổng phần thưởng của nó đạt được sau T vòng lặp lại thấp. Thuật toán cũng phải *khai thác* các thông tin ví dụ như kỳ vọng của phần thưởng của các lựa chọn nó đã chọn, để chọn hành động cho ra nhiều phần thưởng nhất. Thuật ngữ Multi-Armed Bandits cũng được sinh ra từ một bài toán tương tự trong thực tế, trong đó người chơi ở các sòng bạc lựa chọn một trong K máy đánh bạc khác nhau. Người chơi cũng phải cân bằng giữa chọn máy khác nhau để ước lượng số tiền trung bình sinh ra từ máy, đồng thời phải khai thác các máy cho ra nhiều tiền nhất.

Giao thức của mô hình MAB được mô tả trong Thuật toán 2. Để định lượng được tính hiệu quả của một thuật toán giải MAB, xem thuật toán đó có thực hiện tốt việc cân bằng giữa *khai thác* và *khám phá* hay không, ta sử dụng chỉ số về độ hối tiếc (*regret*). Độ hối tiếc của thuật toán giải MAB sau T vòng được định nghĩa cụ thể như sau:

$$R(T) = \mu^* \times T - \sum_{t=1}^{T} \mu(a_t), \tag{2.1}$$

trong đó, $R(T)$ là tổng mức độ hối tiếc tại vòng thứ T, $\mu(a_t)$ là giá trị trung bình của phần thưởng sinh ra từ các lựa chọn của thuật toán đang xét, và μ^* là giá trị trung bình của hành động tối ưu (giả sử như ta biết được hành động tối ưu là hành động tại một bộ dữ liệu thử nghiệm). Giá trị của mức độ hối tiếc thể hiện mức độ sai lệch giữa chính sách lựa chọn hành động của thuật toán và chính sách lựa chọn tối ưu. Trong nghiên cứu về lý thuyết cho MAB, các thuật toán giải MAB sẽ được phân tích và chứng minh rằng thuật toán đó đạt được giá trị hối tiếc dưới mức tuyến tính. Tức là, nếu $lim_{T\to\infty}\frac{R(T)}{T} = 0$ thì việc *khai thác* và *khám phá* của thuật toán đang xét có hiệu quả, vì hối tiếc không tăng tuyến tính với thời gian. Thường các nghiên cứu lý thuyết cho MAB sẽ chứng minh giá trị hối tiếc của một thuật toán giải MAB có dạng hàm *loga*, và đến một thời gian T đủ lớn, $R(T)$ sẽ tăng không đáng kể nữa, tương đương với việc thuật toán đã học được để khai thác hành động tối ưu.

Chương 3

Các nghiên cứu liên quan

Thuật toán tiến hóa đa nhiệm lần đầu tiên được đề xuất vào năm 2015 [21]. Từ đó đến nay, mô hình thuật toán này đã được nghiên cứu rộng rãi, với kết quả là có tổng cộng 87 công trình nghiên cứu về chủ đề này được công bố trên các hội thảo đầu ngành tính toán tiến hóa như GECCO, CEC, WCCI, hội thảo ngành trí tuệ nhân tạo như AAAI, IJCAI hay các tạp chí có tầm ảnh hướng lớn như IEEE Transactions on Evolutionary Computation, IEEE Transactions on Cybernetics. Có nhiều chủ đề nhỏ có thể khai thác để phát triển thuật toán này. Tuy nhiên, trong phạm vi của luận văn này, tôi chỉ đề cập đến những nghiên cứu liên quan mật thiết nhất đến các nghiên cứu cơ bản cho cơ chế lựa chọn ghép cặp các tác vụ trong MFEA và các nghiên cứu về thuật toán MFEA giải số lượng lớn tác vụ. Chương nghiên cứu liên quan này được chia làm hai phần chính. Phần thứ nhất mô tả các nghiên cứu liên quan có cơ chế ghép cặp tác vụ một cách tự động của MFEA nói chung. Ở phần kế tiếp, các nghiên cứu tập trung vào giải nhiều tác vụ hơn được sẽ được liệt kê và phân tích ưu nhược điểm.

3.1 Bài toán tối ưu đa tác vụ

Như đã nêu trên, thuật toán tối ưu đa tác vụ thu hút được nhiều sự quan tâm của cộng đồng tính toán tiến hóa. Có rất nhiều nghiên cứu liên quan đề xuất nhiều phiên bản khác nhau của thuật toán này trong việc tối ưu một số lượng ít các tác vụ (ví dụ: từ 2 đến 3 tác vụ). Ở đây, tôi chỉ liệt kê trọng tâm hai nghiên cứu của nhóm tác giả gốc của thuật toán tối ưu đa nhiệm, vì hai nghiên cứu này được coi là có ảnh hưởng sâu rộng nhất đến hướng nghiên cứu.

Trước tiên, phải kể đến nghiên cứu đề xuất thuật toán gốc MFEA [21]. Ở nghiên cứu này, nhóm tác giả đã đặt nền móng cho cả hướng nghiên cứu về tiến hóa đa nhiệm. Thuật toán đề xuất MFEA là một khung mẫu đơn giản nhất để chứng minh được giả thiết thuật toán tiến hóa đa nhiệm tốt hơn so với tiến hóa đơn nhiệm. Như đã mô tả ở Phần 2, MFEA khác thuật toán tiến hóa đơn nhiệm ở chỗ MFEA cho phép lai ghép giữa các tác vụ khác nhau. Thông tin mà các tác vụ trao đổi cho nhau giúp đẩy nhanh quá trình tối ưu của từng tác vụ thành phần. Tuy nhiên, MFEA không có cơ chế điều chỉnh và thay đổi lượng thông tin trao đổi giữa các tác vụ. Tất cả các tác vụ có thể trao đổi lẫn nhau với xác suất được định nghĩa bằng thông số *rmp*. Thông số này là một hằng số cho mọi cặp tác vụ, và người dùng thuật toán phải tự điều chỉnh. Nếu người dùng tự biết trước được các tác vụ liên quan đến nhau nhiều thì có thể chỉnh *rmp* tiến tới 1, hoặc ngược lại, nếu các tác vụ ít liên quan thì có thể chỉnh *rmp* sát về 0. Tuy nhiên, thuật toán tiến hóa nói chung giải quyết bài toán tối ưu hộp đen, nên ta không thể biết được tính chất các hàm, huống chi đến việc biết được mối quan hệ giữa các hàm với nhau. Đây là nhược điểm của thuật

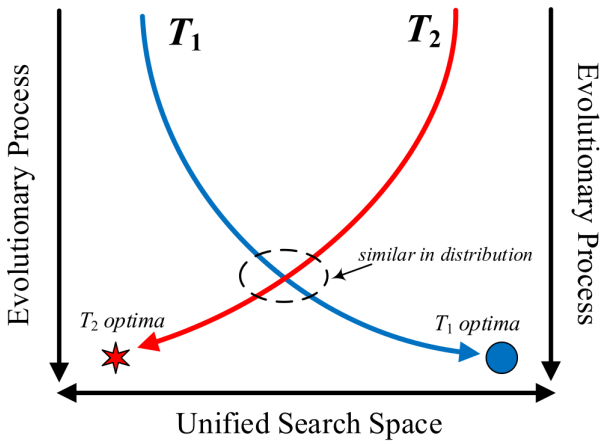
toán MFEA gốc mà được nhiều nghiên cứu sau đó khai thác và cải thiện.

Tiếp đó, nghiên cứu về thuật toán Multifactorial Evolutionary Algorithm - II (MFEA-II) [12] của cùng nhóm nghiên cứu Bali, Ong, Gupta, and Tan đã cải tiến MFEA bằng cách thiết kế một mô-đun học độ tương đồng của các tác vụ để tự điều chỉnh *rmp*. Ý tưởng chính của MFEA-II là coi bài toán điều chỉnh tham số xác suất lai ghép khác tác vụ là một bài toán con, và mô hình hóa để tối ưu bài toán đó song song với bài toán tối ưu đa nhiệm gốc. Giả sử cho K tác vụ, MFEA-II lưu một ma trận xác suất lai ghép khác tác vụ $RMP \in [0,1]^{K\times K}$, để mỗi cặp tác vụ lại có một xác suất lai ghép riêng. Đồng thời, MFEA-II giả định toán tử lai ghép mà thuật toán này sử dụng là toán tử lai ghép $parent - centric$ (tức là sau khi lai ghép, phân phối xác suất sinh ra cá thể con gần với phân bố của cá thể cha mẹ). Nhờ giả định trên, nhóm tác giả phân tích lý thuyết và chứng minh được công thức phân phối xác suất của con sinh ra nếu như nó được lai ghép giữa cặp tác vụ T_j và T_k bất kỳ. Phân phối này phụ thuộc vào hai thành phần chính là tập quần thể cha mẹ của các tác vụ T_j và T_k, và xác suất lai ghép giữa hai tác vụ đó là $RMP_{j,k}$. MFEA-II cũng cho lai ghép, đột biến, đánh giá, chọn lọc và sinh ra quần thể cha mẹ sau chọn lọc. Cuối cùng, MFEA-II điều chỉnh $RMP_{j,k}$ sao cho xác suất tính được của công thức phân phối xác suất của con sinh chứng minh được ở trên trên giống với dữ liệu thật (quần thể đã được chọn lọc) nhất. Vì hàm khoảng cách giữa phân phối dự đoán và dữ liệu thật được chứng minh là hàm lồi nên dễ dàng tối ưu được $RMP_{j,k}$ sử dụng các công cụ tối ưu hàm lồi sẵn có. MFEA-II chỉ có nhược điểm là tốn tài nguyên tính toán khi số lượng tác vụ K tăng. Với K tác vụ, MFEA-II phải giải gần K^2 bài toán tối ưu hàm lồi khác nhau, rất tốn tài nguyên tính toán, nên thuật toán này chỉ phù hợp với việc giải quyết các bài toán với ít số lượng tác vụ.

3.2 Bài toán tối ưu đa tác vụ với số lượng tác vụ lớn

Mặc dù đã có rất nhiều nghiên cứu đề xuất giải thuật mới và phân tích hoạt động của MFEA với giả thiết là số lượng tác vụ nhỏ như 2, hoặc 3 tác vụ, số lượng nghiên cứu về giải quyết một số lượng lớn các tác vụ lại chưa nhiều. Một trong nhưng nghiên cứu đầu tiên trong việc phát triển thuật toán tối ưu đa tác vụ với số lượng tác vụ lớn được công bố vào năm 2018 [14]. Nguyên cứu này đề xuất thuật toán Group-based Multifactorial Evolutionary Algorithm (GMFEA). Ý tưởng chính của thuật toán là sử dụng phương pháp học máy không giám sát để nhóm các tác vụ gần nhau lại thành các cụm nhỏ hơn, sau đó việc *lai ghép khác tác vụ* chỉ diễn ra giữa các tác vụ thuộc cùng một cụm. Cụ thể, GMFEA sử dụng thuật toán phân cụm K-Means. Với mỗi tác vụ, GMFEA chọn một vài cá thể tốt nhất làm đại diện, sau đó ghép với các cá thể đại diện của các tác vụ khác tạo thành tập dữ liệu đầu vào cho K-Means. Thuật toán K-Means thực hiện việc phân cụm các cá thể, và tách các tác vụ thành các cụm nhỏ. Với cách làm này, ở mỗi thế hệ, GMFEA có thể xác định được tác vụ nào có các cá thể tốt nhất gần với tác vụ nào khác, từ đó chỉ cho các tác vụ thuộc cùng một cụm mới được *lai ghép khác tác vụ* với nhau. Tuy nhiên việc ghép cặp chỉ dựa trên khoảng cách quần thể như GMFEA chưa chắc có thể làm cải thiện kết quả tối ưu cho các tác vụ thành phần. Nghiên cứu sau đó [16] chỉ ra một trường hợp minh họa vì sao cách đo và ghép cặp này có thể không hiệu quả (Hình 3.1). Hai tác vụ tuy có cùng phân phối của quần thể tại một thế hệ, nhưng có thể hai tác vụ đó cần phải đi về hai hướng khác nhau để tới được cực trị toàn cục. Khi cho lai ghép hai tác vụ dù gần nhau nhưng cần phải đi hai hướng khác nhau, hai tác vụ đó sẽ giữ chân nhau, và cả hai đều lâu tới được điểm cần đến là cực trị toàn cục riêng.

Vào năm 2019, thuật toán Evolutionary many-tasking based on Symbiosis in Biocoenosis (SBSGA) [15] cũng được đề xuất để giải quyết bài toán tối ưu số lượng lớn tác vụ này.



Hình 3.1: Ví dụ minh họa cho việc hai tác vụ có cùng phân phối của quần thể tại một thời điểm, nhưng hướng để đi đến cực trị toàn cục khác nhau [16].

Ở SBSGA, tác giả đề xuất hai ý tưởng chính để cải thiện so với MFEA hoặc GMFEA. Về ý tưởng thứ nhất, thay vì việc lai ghép khác tác vụ như MFEA, SBSGA trao đổi tri thức bằng cách đưa trực tiếp cá thể của một tác vụ sang tác vụ khác. SBSGA làm như vậy để kết hợp được với các thuật toán tiến hóa đơn nhiệm mà không có phép lai ghép, như thuật toán Evolution Strategies (ES). Tuy nhiên, nhược điểm của cách làm này là việc thay trực tiếp nghiệm của một tác vụ cho tác vụ khác chỉ tốt khi toàn bộ nghiệm của tác vụ truyền đi tốt trên tác vụ nhận về. Vẫn có khả năng tác vụ truyền đi chỉ truyền một phần nghiệm cho tác vụ nhận thì kết quả con sinh ra đánh giá trên tác vụ nhận mới tốt như ví dụ trong Bảng 3.1. Nếu sử dụng cách lai ghép khác tác vụ như MFEA gốc thì phép lai ghép chỉ lấy ngẫu nhiên một phần của nghiệm tác vụ truyền để ghép với tác vụ nhận, phù hợp hơn với các cặp tác vụ như ví dụ này.

Tác vụ	**Tên hàm**	**Điểm tối ưu toàn cục**	**Loại**
T_1	*Sphere* (Công thức 5.1)	$(o_1, o_2, \ldots, o_D) = (80, 80, \ldots, 80)$ $\mathbf{x} \in [-100, 100]^D, D = 50$	Dễ
T_2	*Griewank* (Công thức 5.6)	$\left(o_1, o_2, \ldots, o_{\frac{D}{2}}\right) = (-80, -80, \ldots, -80)$ $\left(O_{\frac{D}{2}+1}, o_2, \ldots, o_D\right) = (80, 80, \ldots, 80)$ $\mathbf{x} \in [-100, 100]^D, D = 50$	Khó

Bảng 3.1: Ví dụ hai tác vụ mà việc trao đổi một phần nghiệm tối ưu cho nhau sẽ cho kết quả tốt hơn là trao đổi toàn bộ nghiệm cho nhau.

Tiếp đó, ý tưởng thứ hai của SBSGA là sử dụng một mô-đun để thống kê lại tần số tráo đổi thành công giữa từng cặp tác vụ. Sau đó, SBSGA định nghĩa một công thức để suy từ tần số trên ra tham số *rmp* (xác suất cho phép tráo đổi khác tác vụ) của từng cặp tác vụ. Trong mỗi thế hệ, SBSGA chỉ cho tráo cá thể từ tác vụ này sang tác vụ kia với xác suất *rmp* đã tính. Tuy nhiên, công thức tính của SBSGA là một công thức heuristic chưa được chứng minh về độ hiệu quả bằng phân tích toán học. Cụ thể, chưa có nghiên cứu nào chứng minh công thức đó đảm bảo được việc cân bằng giữa việc khám phá (thử kết hợp một tác vụ với tác vụ chưa kết hợp bao giờ), và khai thác (khai thác tần số trao đổi thành công đã đo được).

Thuật toán mới nhất về tối ưu đa tác vụ với số lượng tác vụ lớn tính tới thời điểm nghiên của tôi được thực hiện là thuật toán Many-Task archive-based Genetic Algorithm (MaTGA) [16]. Thuật toán này giải quyết được vấn đề của GMFEA, đó là vấn đề thuật toán không tốt khi quần thể của hai tác vụ gần nhau nhưng đi về hai hướng khác nhau

trong không gian tìm kiếm (Hình 3.1). Thay vì việc đo khoảng cách giữa quần thể của các cặp tác vụ tại một thế hệ, MaTGA lưu lại lịch sử quần thể của các tác vụ qua nhiều thế hệ, sau đó đo khoảng cách của các cặp tác vụ trên tập cá thể đã lưu trữ qua nhiều thế hệ đó. Nhờ vậy, MaTGA không chỉ đo được mức độ tương đồng của hai tác vụ tại một thời điểm, mà thuật toán này còn nắm bắt được xem hai tác vụ có dịch chuyển cùng hướng nhau trong không gian tìm kiếm hay không. Ngoài ra, thay vì việc chạy thuật toán K-Means để phân cụm các tác vụ, MaTGA đơn thuần đo khoảng cách các tác vụ sử dụng khoảng cách Kullback-Leibler Divergence. Tuy nhiên, nhược điểm của thuật toán này là nó có quá nhiều tham số. Người dùng phải cấu hình lưu trữ bao nhiêu cá thể trong mỗi thế hệ, giữ bao các cá thể bao nhiêu thế hệ, và nhiều tham số khác để cả hệ thống đo khoảng cách các tác vụ hoạt động một cách hiệu quả. Thuật toán chỉ được kiểm nghiệm trên một tập dữ liệu, nên có thể tham số chỉ được tác giá tinh chỉnh cho một bộ dữ liệu. Tôi đưa ra giả thiết là khi đem cùng bộ tham số này ra kiểm nghiệm trên bộ dữ liệu khác thì kết quả của thuật toán này trên bộ dữ liệu khác sẽ tồi đi rất nhiều, vì chưa được tinh chỉnh tham số. Ý tưởng của thuật toán để giải quyết vấn đề hai tác vụ đi về hai hướng trong không gian tương đối mạch lạc, tuy nhiên ý tưởng này khó có thể triển khai ngoài thực tế vì thuật toán chạy chậm và tốn bộ nhớ. Cụ thể là vì, MaTGA vừa phải lưu cá thể của nhiều tác vụ tại nhiều thời điểm trong quá khứ, vừa phải đo K^2 lần khoảng cách Kullback-Leibler Divergence của tất cả các cặp tác vụ (giả sử số tác vụ là K).

Nắm bắt được những vấn đề của các thuật toán trong các nghiên cứu liên quan, tôi đề xuất thuật toán Ma^{2}BEA vừa giải quyết vấn đề phát hiện sai tác vụ hỗ trợ của GMFEA, vừa nhanh hơn và ít tham số phải tinh chỉnh hơn so với MaTGA, mà phương pháp ghép các tác vụ có nền tảng lý thuyết và có chứng minh hỗ trợ, không như heuristic đề xuất lại thuật toán SBSGA.

Chương 4

Thuật toán đề xuất

Tại chương này, thuật toán đề xuất Ma^{2}BEA sẽ được mô tả chi tiết. Chương chia làm bốn phần chính, ba phần đầu mô tả lần lượt các thành phần và tổng quan các bước của thuật toán. Phần cuối cùng trình bày cách ứng dụng thuật toán tiến hóa đa nhiệm giải nhiều bài toán tối ưu mạng nơ ron.

4.1 Cơ chế lựa chọn tác vụ tương đồng bằng mô hình Multi-Armed Bandits

Mục này trình bày cơ chế lựa chọn tác vụ tương đồng để thực hiện *lai ghép khác tác vụ* của Ma^{2}BEA, mà hiệu quả hơn so với thuật toán MFEA phiên bản gốc. Việc ghép cặp các tác vụ có thể được coi là một bài toán con của bài toán tối ưu đa tác vụ, và được định nghĩa lại như sau. Giả sử ta có cá thể cha p_a được lấy ngẫu nhiên từ một tác vụ T_k bất kỳ. Ma^{2}BEA cần lựa chọn một tác vụ thứ $T_{k'}$ phù hợp. Sau đó, cá thể mẹ p_b được lấy ngẫu nhiên ra từ quần thể $P^{k'}$. p_a được cho lai ghép với p_b để sinh ra cá thể con c. Giả sử các bài toán tối ưu đang xét đều là các bài toán tối thiểu hóa, thì giá trị hàm mục tiêu của con sinh ra càng nhỏ càng tốt. Nếu giá trị hàm mục tiêu của c trên tác vụ T_k nhỏ hơn so với giá trị hàm mục tiêu cao nhất của các cha mẹ trong P^k thì việc trao đổi giữa hai tác vụ k và k' là có ích. Mục tiêu của Ma^{2}BEA là chọn tác vụ k' cho k sao cho sinh ra càng nhiều cá thể con tốt như vậy càng tốt. Bài toán con nêu trên là một bài toán đưa ra quyết định theo thời gian. Các thành phần trong bài toán cũng như yêu cầu của bài toán đều giống như mô tả của mô hình MAB. Ngoài ra, vì lý thuyết trong việc giải mô hình MAB đã được nghiên cứu kỹ lưỡng, nên ở đây, tôi đề xuất định nghĩa lại bài toán này theo khung MAB và sử dụng hướng giải phù hợp. Các thành phần của bài toán ghép tác vụ trên mô hình MAB được định nghĩa như sau:

Định nghĩa 3 (Lựa chọn)**.** *Với mỗi tác vụ T_k, sẽ có $K-1$ lựa chọn, tương ứng với $K-1$ tác vụ $T_{k'}$ mà $k' \in \{1, \ldots, K\}$ và $k' \neq k$.*

Định nghĩa 4 (Phần thưởng)**.** *Sau khi tác vụ $T_{k'}$ được lựa chọn để ghép cặp với tác vụ T_k, phần thưởng của việc chọn tác vụ $T_{k'}$ được định nghĩa như sau:*

$$r(k, k') = \begin{cases} 1 \text{ nếu } f_k(c) < f_k(p), \exists p \in P^k \\ 0 \text{ trong các trường hợp khác.} \end{cases} \tag{4.1}$$

trong đó, c là con sinh ra trong quá trình lai ghép khác tác vụ, $f_k(.)$ là hàm đánh giá của tác vụ T_k.

Vì phần thưởng được định nghĩa ở là biến ngẫu nhiên chỉ chứa hai giá trị 0 và 1, nên ta giả sử phân phối ngẫu nhiên của phần thưởng là một phân phối Bernoulli chưa biết trước. Với dạng phần thưởng như trên, các nghiên cứu tiền nhiệm cho bài toán MAB cơ bản đã đề xuất một thuật toán giải quyết là KL-UCB [23]. Phương pháp này đề xuất phương trình KL-UCB để lựa chọn hành động như sau:

$$k' = \underset{j}{\operatorname{argmax}}\ \mu(j) + \frac{1 + t \times log^2(t)}{N(j)} \tag{4.2}$$

trong đó, $\mu(j)$ là giá trị trung bình ước lượng được của phần thưởng khi lựa chọn j, $N(j)$ là tổng số lần thuật toán đã lựa chọn j, và t là tổng số của tất cả các lần lựa chọn mà thuật toán đã thực hiện tính cho đến thời điểm đang xét. Thành phần $\frac{1+t\times log^2(t)}{N(j)}$ là thành phần thể hiện mức độ thiếu tin cậy của việc ước tính giá trị trung bình của phần thưởng $\mu(j)$. Nếu t tăng thì độ thiếu tin cậy của việc ước tính $\mu(j)$ càng tăng. Nếu lựa chọn j càng được chọn nhiều lần thì $N(j)$ tăng cao dẫn đến thành phần mô tả mức độ thiếu tin cậy nói trên giảm.

Tác vụ $T_{k'}$ được chọn bởi phương trình KL-UCB là tác vụ có giá trị phần thưởng trung bình và tổng mức độ thiếu tự tin về ước lượng giá trị phần thưởng cao nhất. Phương trình này cân bằng giữa việc khai thác các lựa chọn có giá trị phần thưởng cao, và việc khám phá các lựa chọn có độ tin cậy về mặt ước lượng phần thưởng thấp. Phương trình này đã được chứng minh về mặt lý thuyết là đạt được độ hối tiếc dưới tuyến tính [23]. Vậy nên, tôi đề xuất ứng dụng phương pháp này để cải thiện việc ghép cặp các tác vụ trong thuật toán MFEA. Ưu điểm của việc sử dụng phương pháp này là phương pháp đã được chứng minh lý thuyết, không phải heuristic tự thiết kế, và không làm tăng thêm số lượng tham số cho người dùng như [15, 16, 14]

Ngoài ra, bài toán hướng tới giải quyết chính ở đây là bài toán tối ưu đa tác vụ với số lượng tác vụ lớn. Các tác vụ rất nhiều và đa dạng nên ta có thể giả định là với một tác vụ T_k bất kỳ, luôn luôn tồn tại một tác vụ $T_{k'}$ hỗ trợ cho nó. Vậy nên ít có khả năng là không thể tìm được $T_{k'}$, cũng như ít có khả năng tối ưu T_k bằng thuật toán đơn nhiệm lại cho ra kết quả tốt hơn. Ta loại đi trường hợp hành động là không lựa chọn tác vụ nào. Bằng việc tận dụng KL-UCB, Ma^{2}BEA có thể lựa chọn các cặp cha mẹ từ các tác vụ tương đồng và hỗ trợ lẫn nhau. Đồng thời, so với MFEA-II [12], Ma^{2}BEA tuy cũng giải quyết một bài toán con trong bài toán tối ưu đa nhiệm nhưng ít đòi tài nguyên tính toán hơn MFEA-II, vì các lựa chọn là biến nhị phân chứ không phải điều chỉnh ma trận số thực *rmp* như MFEA-II, và Ma^{2}BEA không có lựa chọn là không lai ghép khác tác vụ.

Thuật toán 3 nêu cách tích hợp mô hình MAB đã đề xuất vào thuật toán MFEA. Tại mỗi thế hệ, khi tập trung giải từng tác vụ k, Ma^{2}BEA vẫn thực hiện các bước như thuật toán tiến hóa cơ bản, đó là khởi tạo quần thể con (dòng 1), lai ghép cha mẹ (dòng 3 đến dòng 11), đột biến (dòng 12), chọn lọc cá thể tốt cho thế hệ tiếp theo (dòng 15 đến dòng 17). Như các thuật toán tiến hóa đa nhiệm khác, thuật toán Ma^{2}BEA cũng cho phép việc lai ghép khác tác vụ với xác suất *rmp* (dòng 4 đến dòng 7). Điểm khác biệt ở Ma^{2}BEA so với các thuật toán khác là cơ chế lựa chọn tác vụ có tính hỗ trợ cao cho tác vụ T_k (dòng 5, dòng 6). Sau khi lựa chọn xong tác vụ để trao đổi với T_k, Ma^{2}BEA thực hiện tính toán và cho ra phần thưởng cho lựa chọn trên như trong Đinh nghĩa 4. Ở dòng 14, Ma^{2}BEA cập nhật lại ước lượng của phần thưởng và độ tin cậy của ước lượng phần thưởng để sử dụng tiếp các giá trị này trong những lần lựa chọn tiếp đó.

Algorithm 3 Ma^{2}BEA trong mỗi thế hệ của tác vụ T_k

1: Khởi tạo quần thể con $P^k_{(c)} = \emptyset$;
2: **while** số con sinh ra $< N$ **do**
3: Chọn ngẫu nhiên cá thể cha p_a từ P^k;
4: **if** $rand(0, 1) < rmp$ **then**
5: Chọn tác vụ $T_{k'}$ sử dụng phương trình KL-UCB trong Công thức 4.2;
6: Chọn ngẫu nhiên cá thể mẹ p_b từ $P^{k'}$;
7: $c =$ *Lai ghép khác tác vụ* giữa p_a và p_b;
8: **else**
9: Chọn ngẫu nhiên cá thể mẹ p_b từ P^k;
10: $c =$ *Lai ghép cùng tác vụ* giữa p_a và p_b;
11: **end if**
12: $c =$ Đột biến c;
13: Đánh giá cá thể con c;
14: Cập nhật lại ước lượng $\mu(k')$ và số lượng $N(k')$ cho KL-UCB nếu c được sinh ra từ việc *Lai ghép khác tác vụ*;
15: $P^k_{(c)} = P^k_{(c)} \cup \{c\}$;
16: **end while**
17: $P^k \leftarrow$ Chọn N cá thể tốt nhất từ $P^k \cup P^k_{(c)}$ để tạo lại P^k cho thế hệ tiếp theo;

Algorithm 4 Giả mã của toàn bộ Ma^{2}BEA

1: **for** $k \in \{1, \ldots, K\}$ **do**
2: Khởi tạo ngẫu nhiên $N \sim \mathbb{R}^{D_{unified}}$ cá thể để tạo ra quần thể P^k;
3: Evaluate individual in P^k for task T_k only;
4: **end for**
5: **while** điều kiện kết thúc chưa thỏa mãn **do**
6: **for** $k \in$ hoán vị ngẫu nhiên của $(1, \ldots, K)$ **do**
7: Thực hiện Thuật toán 3 cho tác vụ T_k;
8: **end for**
9: **end while**

4.2 Cấu trúc mới cho thuật toán tiến hóa đa nhiệm

Trong mục này, tôi trình bày đề xuất về cấu trúc mới cho thuật toán tiến hóa đa nhiệm mà phù hợp hơn với các bài toán tối ưu có rất nhiều tác vụ. Ý tưởng của cấu trúc thuật toán mới này được tóm tắt tại thuật toán 4.

Tương tự như MFEA, Ma^{2}BEA hoạt động trên không gian biểu diễn chung như đã mô tả ở Mục 2.1. Cụ thể là các phép lai ghép và đột biến đều được thực hiện trên không gian chung đó, và cá thể chỉ được giải mã ra các không gian tìm kiếm riêng của từng tác vụ tối ưu trong bước đánh giá độ tốt của cá thể đó. Mặc dù biểu diễn ở không gian chung nhưng Ma^{2}BEA lưu các cá thể của các tác vụ khác nhau trong các tập quần thể con $P^k, \forall k \in \{1, \ldots, K\}$ riêng. Việc lưu trữ như vậy giúp cho các bước như chọn một cá thể ngẫu nhiên trong tác vụ k (dòng 3, dòng 6, Thuật toán 3) trở nên dễ dàng hơn.

Ở bước đầu của giải thuật, các quần thể con của từng tác vụ P^k sẽ được khởi tạo. Tiếp đó, Ma^{2}BEA thực hiện lặp đi lặp lại việc tiến hóa các quần thể con trên nhiều thế hệ. Ở mỗi thế hệ, Ma^{2}BEA cải thiện từng quần thể của từng tác vụ một cách tuần tự. Bên cạnh đó, các thuật toán tiến hóa đa nhiệm trước đây thường đợi cho tất cả các tác vụ thực hiện hết tất cả các công việc như lai ghép, đột biến, đánh giá rồi sau đó mới cập

nhật toàn bộ quần thể P của tất cả các tác vụ. Ma^{2}BEA thì không cần đợi tất cả các tác vụ phải hoàn thiện mới thực hiện cập nhật. Thay vào đó, Ma^{2}BEA thực hiện phép chọn lọc tự nhiên và cập nhật ngay sau bước đánh giá con sinh ra của từng tác vụ. Bằng cách này, tại Ma^{2}BEA có thể có những tác vụ đi trước và cập nhật trước để dò đường trên không gian tìm kiếm. Các tác vụ được giải quyết sau đó trong cùng một thế hệ sẽ được trao đổi tri thức với các tác vụ đi trước đó. Giả sử KL-UCB ghép đúng cặp hai tác vụ đi trước và đi sau, mà hai tác vụ có sự tương đồng cao về địa hình của hàm mục tiêu (highly ordinal correlation in fitness landscape). Tác vụ đi trước ở đây đóng vai trò như một quần thể chạy nhanh hơn và đi trước trong không gian tìm kiếm, truyền thông tin về địa hình tương lai cho tác vụ đi sau. Việc này giúp các nghiệm tốt hơn của tác vụ đi trước sẽ được truyền luôn cho tác vụ tương đồng đi sau, giúp làm giảm số lượng lần đánh giá để khám phá không gian tìm kiếm của tác vụ đi sau. Vậy nên, tác vụ đi sau sẽ hội tụ về cực trị nhanh hơn.

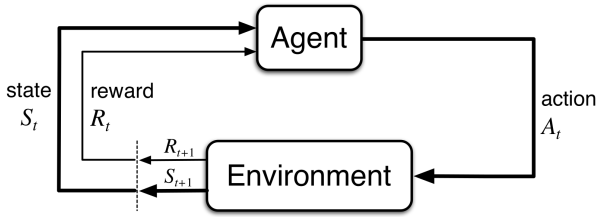
Để đảm bảo việc các tác vụ đều được có lợi, thứ tự tối ưu của các tác vụ trong một thế hệ được hoán đổi ngẫu nhiên (dòng 6, Thuật toán 4). Nhờ đó, tác vụ nào cũng được đi sau như nhau để hưởng lợi từ việc các tác vụ tương đồng với nó đi trước để quét trước không gian tìm kiếm xem khu vực nào tốt hơn.

Ý tưởng này cũng lấy cảm hứng từ thuật toán Look-ahead optimizer [24] trong tối ưu hóa của các mô hình học sâu. Ngoài ra, cấu trúc mới này cũng được lấy cảm hứng từ cơ chế làm việc đa nhiệm của não người. Như trong nghiên cứu trong ngành tâm lý học [25], việc gia tăng số lượng công việc mà một người có thể làm cùng không có mối quan hệ gì đáng kể với khả năng não người xử lý các việc đó một cách song song, mà chỉ liên quan đến khả năng con người thực hiện chuyển đổi từ làm việc này sang việc khác một cách nhanh chóng. Nói cách khác, não người luôn luôn xử lý một việc trong một thời điểm. Cấu trúc mới của Ma^{2}BEA mô phỏng lại việc xử lý từng việc một lúc này. Hơn nữa, khi số lượng công việc nhiều lên, và các công việc không thống nhất và không liên quan đến nhau, con người thường không hoàn thành tốt các công việc đó. Ta thường tập trung vào một nhóm nhỏ các việc quan trọng, rồi rời sang một nhóm nhỏ các việc quan trọng khác. Ma^{2}BEA với cơ chế nhóm tác vụ tương đồng mô phỏng lại việc con người tập trung làm các việc giống nhau thay vì các việc khác nhau cùng lúc. Ngoài ra, nếu tác vụ làm trước tương đồng với tác vụ làm sau, con người thường dùng lại kinh nghiệm đã làm để áp cho công việc phía sau. Đây cũng là một trong những nguyên lý tôi dựa vào để thiết kế ra Ma^{2}BEA. Với tất cả những đề xuất trên, tôi đặt ra giả thiết Ma^{2}BEA có thể giải các bài toán tối ưu đa nhiệm có rất nhiều tác vụ tốt hơn.

4.3 Tổng quan về mô hình đề xuất

Tổng kết lại, Ma^{2}BEA có hai điểm mới nổi bật so với các thuật toán tiến hóa đa nhiệm khác. Một là cơ chế ghép cặp để trao đổi tri thức giữa các tác vụ tương đồng. Cơ chế này dựa trên giá trị hàm đánh giá của con sinh ra sau những lần *lai ghép khác tác vụ* để ước lượng mức độ tương hỗ của các cặp tác vụ. Việc cân bằng giữa lựa chọn các tác vụ mới để trao đổi và trao đổi với các tác vụ đã tương đồng sẵn được xử lý bởi thuật toán KL-UCB [23]. Ngoài ra, cấu trúc mới của Ma^{2}BEA, khi thực hiện tiến hóa tuần tự từng tác vụ cũng khiến cho thuật toán này phù hợp hơn với tiến hóa đa nhiệm với số lượng lớn các tác vụ.

Khi so sánh với MFEA, ta thấy cả ba thuật toán MaTGA, SBSGA và Ma^{2}BEA đều trội hơn ở cơ chế trao đổi khác tác vụ tự thích nghi. Tuy nhiên, cơ chế trao đổi của MaTGA có phần cồng kềnh hơn khi có tương đối nhiều tham số để định nghĩa xem nên lưu bao nhiêu cá thể con mỗi thế hệ và lưu lại lịch sử bao nhiêu thế hệ. Ngoài ra, MaTGA



Hình 4.1: Tương tác giữa tác tử (Agent) và môi trường (Environment) trong mô hình MDP [26].

cũng tốn chi phí tính toán hơn vì nó đo khoảng cách giữa các tác vụ trên tập cá thể trong quá khứ, thay vì ước lượng độ hỗ trợ trên giá trị hàm đánh giá như Ma^{2}BEA hay SBSGA. SBSGA tuy cũng đo trên giá trị hàm đánh giá nhưng phương pháp chưa có chứng minh như KL-UCB của Ma^{2}BEA. Nếu kiểu biểu diễn của cá thể thay đổi, ví dụ như đổi từ biểu diễn số thực sang biểu diễn hoán vị hay nhị phân thì hàm đo khoảng cách của MaTGA cũng cần phải thiết kế lại cho phù hợp, còn Ma^{2}BEA hay SBSGA thì vẫn dùng được ngay.

Ta cũng không thể phủ nhận là việc đo độ tương đồng của các tác vụ trên giá trị hàm đánh giá là không tốt bằng ước lượng trên quần thể. Biểu diễn quần thể là tập dữ liệu lớn hơn, có nhiều thông tin đáng giá hơn so với giá trị hàm đánh giá. Vậy nên, một đóng góp nữa của Ma^{2}BEA đó là làm một thuật toán cơ sở để so sánh với các thuật toán tối ưu đa tác vụ tự thích nghi học trên dữ liệu quần thể. Các thuật toán tiến hóa đa nhiệm tự thích nghi mới cần tốt hơn một cách đáng kể so với phiên bản tối ưu đa nhiệm tự thích nghi đơn giản và ít tham số như Ma^{2}BEA.

4.4 Ứng dụng của tiến hóa đa nhiệm cho tối ưu mạng nơ ron trong học tăng cường

Mục này trình bày một trong những ứng dụng của tiến hóa đa nhiệm là tối ưu nhiều mạng nơ ron cho học tăng cường. Hình 4.1 minh họa cho mô hình Markov Decision Process (MDP) cho một trong những mô hình của quá trình học tăng cường [26]. Mô hình MDP là mô hình mở rộng hơn của mô hình MAB đã đề cập ở Mục 2.2. Tại mỗi vòng ra quyết định, tác tử không chỉ nhận về phần thưởng R_t mà còn nhận về trạng thái của môi trường S_t. Việc ra quyết định của tác tử ở mô hình MDP có thể được mô hình hóa lại bằng một mạng nơ-ron có tham số ánh xạ từ không gian trạng thái sang không gian hành động. Trong thực tế, có thể có nhiều môi trường học tăng cường có sự tương đồng cao, ví dụ như điều khiển các robot có hình dạng hoặc trọng lượng chênh lệch [27]. Cụ thể, mục tiêu của bài toán học tăng cường đa tác vụ này là:

$$\underset{\theta_k}{\text{maximize}}\ F(\theta_k; E_k) = \sum_{t=1}^{T} r_t \tag{4.3}$$

trong đó θ_k là tập hợp tham số của mạng nơ-ron điều khiển tác tử thứ k, mà ta cần tối ưu. E_k là môi trường thứ k tương ứng. $F(\theta_k; E_k)$ là hàm đánh giá, tính tổng phần thưởng nhận được khi sử dụng mạng nơ-ron với tham số θ_k điều khiển tác tử trong môi trường E_k.

Thuật toán tiến hóa đa nhiệm có thể tối ưu được nhiều mạng nơ-ron điều khiển nhiều tác tử hoạt động trong các môi trường tương đồng. θ_k là biến tối ưu sẽ được biến đổi bằng các phép lai ghép và đột biến của thuật toán tiến hóa. Ngoài ra, θ_k cũng được *lai ghép khác tác vụ* với $\theta_{k'}, \forall k' \in \{1, 2, \ldots, K-1, K\} | k \neq k'\}$. Ma^{2}BEA sẽ ghép cặp và thực

hiện *lai ghép khác tác vụ* cho các mạng nơ ron của các môi trường có độ tương đồng cao. Bằng việc chỉa sẻ thông tin tối ưu giữa các tác vụ một cách hiệu quả, Ma^{2}BEA có thể giúp tìm được tập tham số θ_k tốt hơn và nhanh hơn so với EA hay MFEA.

Chương 5

Kết quả thực nghiệm

Trong chương này, thuật toán đề xuất Ma^{2}BEA sẽ được đánh giá chi tiết trên nhiều bộ dữ liệu thực nghiệm của bài toán MaTO. Ma^{2}BEA cũng sẽ được so sánh với các thuật toán tiến hóa đa nhiệm mới nhất trong việc tối ưu rất nhiều tác vụ cùng một thời điểm.

5.1 Bài toán tối ưu với 10 tác vụ, biết trước mối quan hệ giữa các tác vụ

5.1.1 Mô tả bộ dữ liệu

Bộ dữ liệu Many-task Optimization (10 tasks) (MaTO-10) (bộ dữ liệu tối ưu 10 tác vụ) đã được đề xuất tại [16]. Bộ dữ liệu này được sinh ra để đánh giá và so sánh mức độ hiệu quả của các thuật toán trong việc giải bài toán MaTO. Bộ dữ liệu bao gồm mô tả của 10 hàm số nhiều biến trên không gian số thực, tạo ra bởi 7 hàm cơ bản như sau:

1. Sphere:
$$F(z) = \sum_{i=1}^{D} z_i^2 \tag{5.1}$$

2. Weierstrass:
$$\begin{aligned} F(z) = &\sum_{i=1}^{D} \left(\sum_{k=0}^{k_{\max}} \left[a^k \cos\left(2\pi b^k \left(z_i + 0.5\right)\right)\right] \right) \\ &- D \sum_{k=0}^{k_{\max}} \left[a^k \cos\left(2\pi b^k * 0.5\right)\right] \\ &a = 0.5, b = 3, k_{\max} = 20 \end{aligned} \tag{5.2}$$

3. Rosenbrock:
$$F(z) = \sum_{i=1}^{D-1} \left(100 \left(z_i^2 - z_{i+1} \right)^2 + \left(z_i - 1 \right)^2 \right); \tag{5.3}$$

4. Ackley:
$$\begin{aligned} F(z) = &-20 \exp\left(-0.2 \sqrt{\frac{1}{D} \sum_{i=1}^{D} z_i^2} \right) \\ &- \exp\left(\frac{1}{D} \sum_{i=1}^{D} \cos\left(2\pi z_i\right) \right) + 20 + e \end{aligned} \tag{5.4}$$

5. Schwefel:

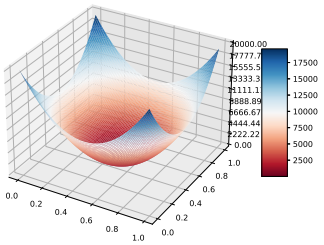
$$F(z) = 418.9829 \times D - \sum_{i=1}^{D} z_i \sin\left(|z_i|^{\frac{1}{2}}\right) \tag{5.5}$$

6. Griewank:

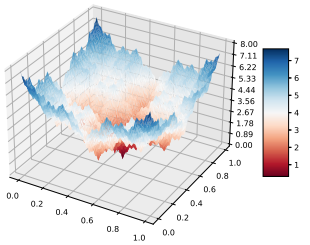
$$F(z) = 1 + \frac{1}{4000}\sum_{i=1}^{D} z_i^2 - \prod_{i=1}^{D} \cos\left(\frac{z_i}{\sqrt{i}}\right) \tag{5.6}$$

7. Rastrigin:

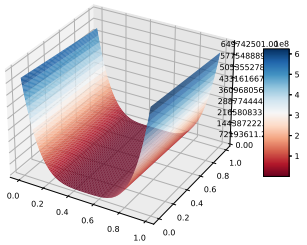
$$F(z) = \sum_{i=1}^{D} \left(z_i^2 - 10\cos\left(2\pi z_i\right) + 10\right) \tag{5.7}$$



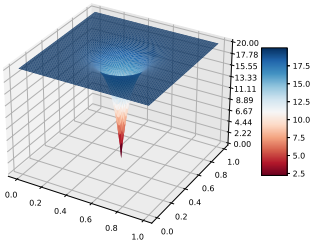
(a) Sphere



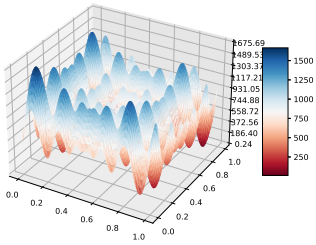
(b) Weierstrass



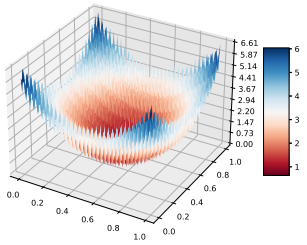
(c) Rosenbrock



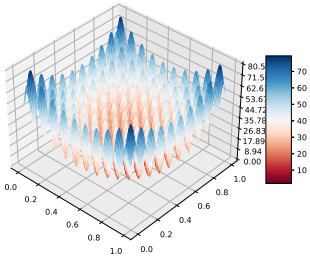
(d) Ackley



(e) Schwefel



(f) Griewank



(g) Rastrigin

Hình 5.1: Minh họa của các hàm cơ bản trong bộ dữ liệu thử nghiệm trên không gian tìm kiếm 2 chiều

Trong mỗi hàm, $z\ \forall i \in \{1, \ldots, D\}$ là đầu vào cho các hàm cơ bản nêu trên, mà đã được biến đổi qua các phép dịch và phép chiếu. Các hàm cơ bản nêu trên được lựa chọn sao cho tính chất và độ khó của các hàm đa dạng.

Bảng 5.1 nêu rõ 10 tác vụ cấu thành bộ dữ liệu thử nghiệm MaTO-10. Với mỗi tác vụ, ta có một hàm cơ bản và một vector dịch O. Tất cả các hàm cơ bản đều có cực trị toàn cục tại $(o_1, o_2, \ldots, o_D) = (0, 0, \ldots, 0)$, và vector dịch O dời cực trị toàn cục của hàm cơ bản đó đến O. Nói cách khác, đầu vào của các hàm cơ bản là $z_i = x_i - o_i\ \forall i \in \{1, \ldots, D\}$. Các tác vụ có thể được phân làm 2 loại: khó và dễ. Các tác vụ khó như $T_5, \ldots, T_{10}$ là

Tác vụ	Tên hàm	Vection dịch (O)	Loại	Tác vụ hỗ trợ
T_1	$Sphere_1$	$(o_1, o_2, \ldots, o_D) = (0, 0, \ldots, 0)$ $\mathbf{x} \in [-100, 100]^D, D = 50$	Dễ	Không có
T_2	$Sphere_2$	$(o_1, o_2, \ldots, o_D) = (80, 80, \ldots, 80)$ $\mathbf{x} \in [-100, 100]^D, D = 50$	Dễ	Không có
T_3	$Sphere_3$	$(o_1, o_2, \ldots, o_D) = (-80, -80, \ldots, -80)$ $\mathbf{x} \in [-100, 100]^D, D = 50$	Dễ	Không có
T_4	$Weierstrass_{25D}$	$(o_1, o_2, \ldots, o_D) = (-0.4, -0.4, \ldots, -0.4)$ $\mathbf{x} \in [-0.5, 0.5]^D, D = 25$	Dễ	Không có
T_5	$Rosenbrock$	$(o_1, o_2, \ldots, o_D) = (0, 0, \ldots, 0)$ $\mathbf{x} \in [-50, 50]^D, D = 50$	Khó	T_1
T_6	$Ackley$	$(o_1, o_2, \ldots, o_D) = (40, 40, \ldots, 40)$ $\mathbf{x} \in [-50, 50]^D, D = 50$	Khó	T_2
T_7	$Weierstrass_{50D}$	$(o_1, o_2, \ldots, o_D) = (-0.4, -0.4, \ldots, -0.4)$ $\mathbf{x} \in [-0.5, 0.5]^D, D = 50$	Khó	T_3, T_4
T_8	$Schwefel$	$(o_1, o_2, \ldots, o_D) = (420.9, 420.9, \ldots, 420.9)$ $\mathbf{x} \in [-500, 500]^D, D = 50$	Khó	Không có
T_9	$Griewank$	$\left(o_1, o_2, \ldots, o_{\frac{D}{2}}\right) = (-80, -80, \ldots, -80)$ $\left(O_{\frac{D}{2}+1}, o_2, \ldots, o_D\right) = (80, 80, \ldots, 80)$ $\mathbf{x} \in [-100, 100]^D, D = 50$	Khó	T_4
T_{10}	$Rastrigin$	$\left(o_1, o_2, \ldots, o_{\frac{D}{2}}\right) = (40, 40, \ldots, 40)$ $\left(o_{\frac{D}{2}+1}, o_2, \ldots, o_D\right) = (-40, -40, \ldots, -40)$ $\mathbf{x} \in [-50, 50]^D, D = 50$	Khó	Không có

Bảng 5.1: Mô tả chi tiết của các hàm trong bộ dữ liệu thử nghiệm MaTO-10

những tác vụ với hàm mục tiêu có nhiều cục bộ địa phương. Khi giải quyết các tác vụ này, các thuật toán tiến hóa thông thường như EA sẽ dễ dàng bị rơi vào các điểm cục bộ địa phương và cần nhiều thế hệ để tìm được nghiệm tốt hơn. Mặt khác, các tác vụ dễ được cấu thành bởi hoặc là hàm lồi (T_1, T_2, T_3), hoặc có số lượng biến tối ưu nhỏ (T_4), nên EA dễ dàng có thể tìm được cực trị toàn cục một cách dễ dàng.

Trong bộ tập hàm đánh giá MaTO-10 này, các tác vụ dễ được ghép cặp với các tác vụ khó, bằng cách thiết kế để cực trị toàn cục của các tác vụ khó trùng với cực trị toàn cục của các tác vụ dễ. Ta có thể tham khảo ví dụ như trong Bảng 5.1, cực trị toàn cục của tác vụ T_5 trùng với tác vụ T_1. Trong trường hợp này, khi quần thể nghiệm của thuật toán EA khi giải tác vụ dễ đi đến khu vực cực trị toàn cực chung, thì những nghiệm tốt này cũng có thể có ảnh hướng tích cực đến việc tìm kiếm của tác vụ khó. Ngoài ra, các tác vụ khó như T_6, T_7, T_9 cũng được thiết kế để có cùng cực trị toàn cục với các tác vụ dễ. Các tác vụ dễ được thiết kế với cực trị toàn cục ở xa nhau, cho nên ta hy vọng thuật toán ghép cặp tác tác vụ sẽ không ghép các tác vụ dễ với nhau. Tác vụ khó như T_8, T_{10} cũng không có cùng cực trị toàn cục với bất kỳ một tác vụ nào khác. Bộ dữ liệu này giả lập môi trường tiến hóa đa nhiệm với số lượng lớn tác vụ, trong đó có rất nhiều dạng quan hệ (tương đồng hoặc không tương đồng) tồn tại cùng lúc.

Mục tiêu chính khi lựa chọn tập hàm đánh giá MaTO-10 để đánh giá thuật toán tối ưu đa tác vụ tự thích ứng như Ma^{2}BEA là kiểm chứng xem thuật toán học mối quan hệ giữa các tác vụ của thuật toán có học đúng hay không. Vì các hàm tối ưu trong bộ hàm đánh giá này là hàm tối ưu hộp đen nên thuật toán tối ưu không có thông tin gì về các hàm và mối liên hệ giữa các hàm. Điều này đồng thời dẫn đến mối quan hệ giữa các tác vụ cũng không có sẵn, mà phải được xác định dần trong quá trình tối ưu chúng bằng một thành phần học mối quan hệ giữa các tác vụ. Ví dụ, nếu thành phần học mối quan hệ giữa các tác vụ xác định đúng được là cần phải ghép và thực hiện trao đổi vật chất di truyền giữa các tác vụ T_5 với T_1, thì thành phần học mối quan hệ đã hoàn thành được

yêu cầu đặt ra. Ngoài ra, với các tác vụ không tương thích như T_1, T_2 và T_3, thuật toán học mối quan hệ phải phát hiện và hạn chế trao đổi kết hợp gen giữa các tác vụ này. Nếu làm được điều đó tức là thuật toán hoạt động tốt và đã giải mã đúng mối quan hệ giữa các hàm tối ưu.

5.1.2 Cài đặt thực nghiệm

Thực nghiệm trên bộ dữ liệu MaTO-10 được thiết và cài đặt như sau. Thuật toán đề xuất Ma^{2}BEA sẽ được chạy và so sánh cùng thuật toán tối ưu đa nhiệm phiên bản gốc MFEA [21]. Hai thuật toán mới nhất đề xuất cho bài toán MaTO, bao gồm MaTGA [16] và SBSGA [15] cũng được dùng làm cơ sở để so sánh. Ở đây, GMFEA không được đưa vào làm cơ sở so sánh do thực nghiệm của các bài báo đề xuất MaTGA [16] và SBSGA [15] đều chỉ ra rằng GMFEA đều cho kết quả kém hơn hai thuật toán mới này.

Về cơ bản, các thuật toán tối ưu đa nhiệm đều là một khung thuật toán và có thể kết hợp nhiều thuật toán tối ưu đơn nhiệm như Genetic Algorithm (GA), Differential Evolutionary algorithm (DE), ES, Genetic Programming (GP) hay Particle Swarm Optimization (PSO) vào các khung thuật toán đó. Ví dụ như thuật toán MFEA đã được đề xuất đề kết hợp với cả 5 biến thể của thuật toán tiến hóa nói trên [21, 28, 15, 29, 30]. Thuật toán MaTGA cũng đã được đề xuất để kết hợp với GA hoặc DE [16]. Tương tự, thuật toán SBSGA cũng được đề xuất để kết hợp với ES [15]. Việc chọn thuật toán tối ưu đơn nhiệm khác nhau để ghép cặp với khung thuật toán tối ưu đa nhiệm sẽ làm ảnh hưởng đáng kể đến kết quả tối ưu. Vậy nên, để việc so sánh các thuật toán tối ưu đa nhiệm như Ma^{2}BEA, MFEA, MaTGA, hay SBSGA được khách quan và công bằng, các thuật toán trên sử dụng cùng một thuật toán tối ưu đơn nhiệm GA với cùng bộ tham số.

Các tham số chung được cài đặt như sau. Trước hết, kích cỡ quần thể để giải mỗi tác vụ được cài đặt chung cho các thuật toán là $N = 100$. Ở bộ MaTO-10 này, có 10 tác vụ nên tổng số lượng cá thể trong quần thể chung của thuật toán tối ưu đa nhiệm là 1000. Các thuật toán đều được chạy $T = 1000$ thế hệ. Phép toán lai ghép trên không gian số thực được dùng là phép Simulated Binary Crossover (SBX) (tạm dịch là phép lai ghép giả lập nhị phân) [31]. Tham số của phép lai ghép SBX được dùng chung là giá trị phân phối $\eta_c = 2$. Phép toán đột biến trên không gian số thực được dùng chung cho các thuật toán tối ưu đa nhiệm là Polynomial Mutation (PM) [32]. Tham số của phép đột biến PM được dùng chung là giá trị phân phối $\eta_m = 5$. Riêng mình thuật toán MaTGA sử dụng phép đột biến Gausian Mutation (GM) [33], vì khi thử sử dụng phép đột biến PM cho MaTGA thì thuật toán đo khoảng các giữa các tác vụ của MaTGA bị đo sai. Còn khi MaTGA sử dụng đột biến GM thì khoảng cách giữa các tác vụ được đo đúng hơn và kết quả cạnh tranh hơn. Hai thuật toán MFEA và Ma^{2}BEA cần cài đặt tham số Random Mating Probability (rmp), là xác suất để hai tác vụ khác nhau có thể trao đổi vật chất di truyền cho nhau. Tham số này được cài đặt là $rmp = 0.3$. Lý do lựa chọn bộ tham số $(T, N, \eta_c, \eta_m, rmp) = (1000, 100, 2, 5, 0.3)$ là vì đây là bộ tham số đã được sử dụng lại nhiều trong các thực nghiệm cho các thuật toán tiến hóa đa nhiệm [21].

Thuật toán Ma^{2}BEA sẽ được phân tích và so sánh với các thuật toán khác dưới nhiều góc độ như sau. Trước hết, các thuật toán sẽ được so sánh với nhau trên chỉ tiêu là giá trị trung bình của 10 hàm đánh giá của bộ dữ liệu MaTO-10 sau khi tối ưu. Vì tính ngẫu nghiên của các thuật toán, tất cả đều được chạy 30, sau đó các thông số của 30 lần chạy được thu thập để đánh giá. Phép kiểm định phi tham số Man Whitney U [34] được sử dụng để đảm bảo việc so sánh có ý nghĩa thống kê. Trong thực nghiệm này, một thuật toán được khẳng định là tốt nhất khi và chỉ khi giá trị trung bình của hàm đánh giá của thuật toán đó sau 30 lần chạy cao hơn tất cả các thuật toán khác, với trị số của phép thử Man Whitney U là $p_{value} < 0.05$.

Thuật toán đề xuất được cài đặt trên ngôn ngữ Python. Vì việc cài đặt lại các thuật toán so sánh như MaTGA và SBSGA tương đối phức tạp nên hai thuật toán này được chạy và lấy kết quả trực tiếp từ cài đặt của các tác giả trên ngôn ngữ Java. Tôi và các thành viên trong nhóm nghiên cứu cũng đã cài đặt thuật toán gốc MFEA trên cả hai ngôn ngữ. Thời gian chạy của các thuật toán sẽ được chuẩn hóa và so sánh dựa trên thời gian chạy của thuật toán đó so với thời gian chạy của thuật toán MFEA cài đặt trên cùng ngôn ngữ và cùng môi trường chạy. Thực nghiệm được chạy trên môi trường hệ điều hành Ubuntu 20.04, với CPU là Intel(R) Core(TM) i9-9900K CPU @ 3.60GHz, với 32GB RAM DDR4.

Ngoài các góc độ trên, việc học cách ghép cặp các tác vụ hỗ trợ lẫn nhau cũng được thống kê và minh họa. Số lần trung bình một tác vụ được ghép cặp với một tác vụ khác nhờ Ma^{2}BEA sẽ được thống kê và minh họa, nhằm kiểm chứng xem mô hình MAB đề xuất có thực sự hoạt động tốt hay không.

5.1.3 Phân tích kết quả thực nghiệm

Bảng 5.2 so sánh thuật toán tôi đề xuất với các thuật toán khác được đề xuất gần đây. Ma^{2}BEA cho kết quả tốt nhất trên 6/10 tác vụ của bộ dữ liệu thực nghiệm MaTO-10. Cụ thể, Ma^{2}BEA giải tốt nhất trên 4 tác vụ dễ và 2 tác vụ khó. Ma^{2}BEA giải tốt hơn tất cả các thuật toán khác trên các tác vụ dễ do nó có khung thuật toán khác các thuật toán còn lại (Mục 4.2). Cấu trúc này giúp các tác vụ có nhiều cơ hội để cùng nhau khám phá không gian tìm kiếm của các hàm mục tiêu hơn, giúp tìm được cực trị toàn cục của các hàm đánh giá có một cực trị một cách nhanh hơn. Bên cạnh đó, xét các tác vụ khó, Ma^{2}BEA hầu như luôn tốt hơn MFEA đáng kể, trừ trường hợp của một hàm đánh giá khó, có nhiều cực trị địa phương như hàm *Rastrigin* của tác vụ T_{10}. Điều này chứng tỏ thuật toán đề xuất với cấu trúc mới và phương pháp ghép cặp tác vụ tương đồng sử dụng MAB đã cải thiện độ hiệu quả một cách đáng kể việc trao đổi kiến thức giữa các tác vụ so với MFEA.

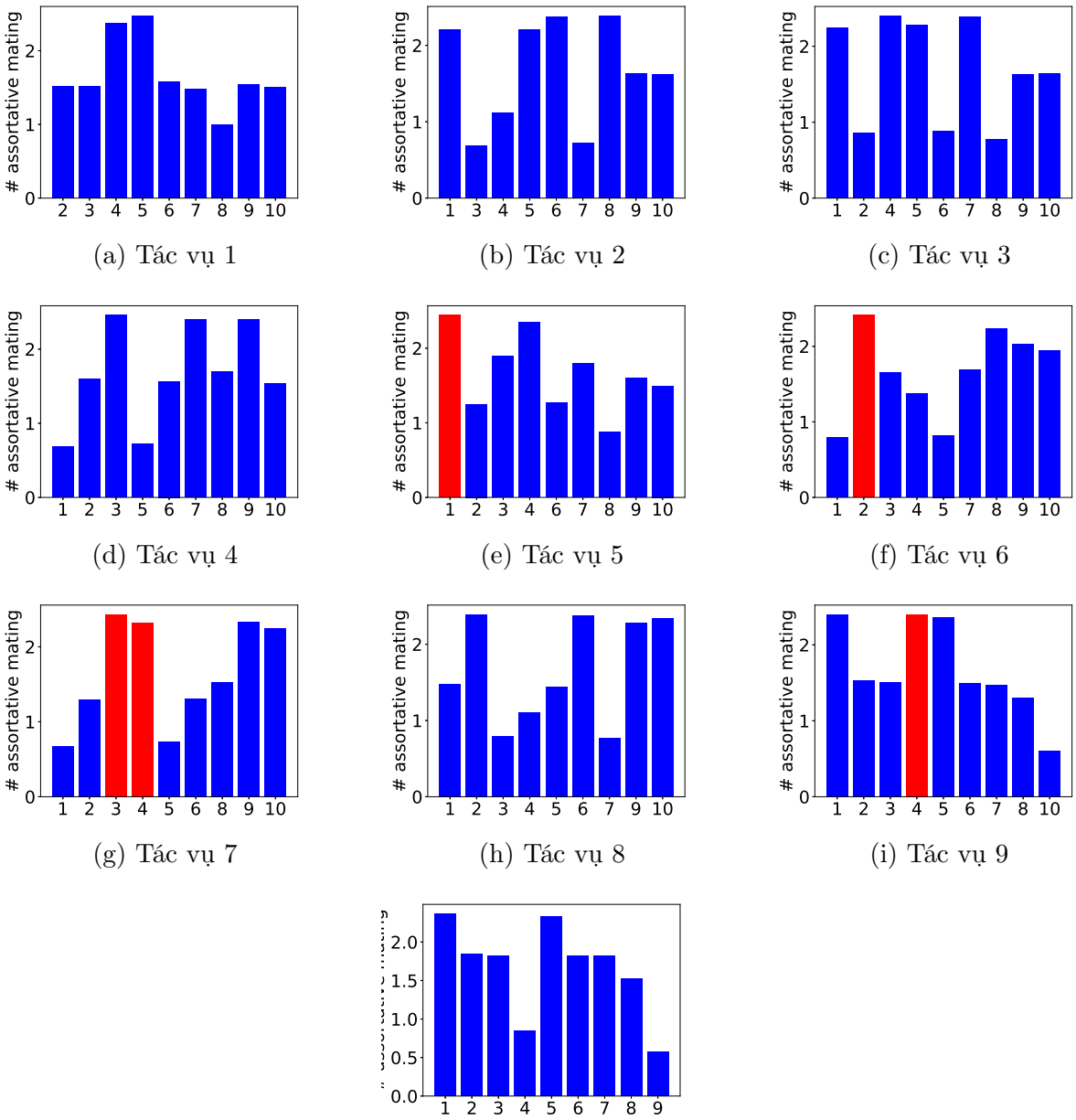
Ngoài ra, Ma^{2}BEA cũng tốt hơn so với MaTGA và SBSGA trên 2 tác vụ khó. Thuật toán xếp thứ nhì sau Ma^{2}BEA là MaTGA, với 4/10 tác vụ tối ưu đạt kết quả tốt nhất. Tuy nhiên, như đã giải thích ở Mục 3.2, MaTGA có tương đối nhiều tham số, bộ tham số tác giả đưa ra đã được tinh chỉnh cho bộ dữ liệu thử nghiệm MaTO-10 này nên kết quả của MaTGA trên bộ dữ liệu này cũng tương đối tốt. Ngoài ra, hai thuật toán còn lại là MFEA và SBSGA cũng tốt nhất ở 1/10 tác vụ.

Ở bộ dữ liệu này, các tác vụ có độ tương đồng cao được biết trước như trong định nghĩa ở Bảng 5.1. Ví dụ, ta biết chắc chắn được là nếu ghép tác vụ T_5 (Khó) với tác vụ T_1 (Dễ) thì tác vụ T_5 sẽ được lợi. Tác vụ T_1 dễ hơn sẽ đi đến cực trị toàn cục của T_1 nhanh hơn, mà T_5 có địa hình hàm mục tiêu của hàm *Rosenbrock* (Công thức 5.3 và Hình 5.1c) tương đồng với hàm mục tiêu *Sphere* (Công thức 5.1 và Hình 5.1a). Ngoài ra, T_1 lại có cùng cực trị toàn cục với T_5, nên nó sẽ giúp T_5 tìm được cực trị toàn cục nhanh hơn. Hình 5.2, minh họa số lần một tác vụ lựa chọn lai ghép với các tác vụ khác.

Ở hình này, số lần trung bình các tác vụ mà hỗ trợ được ghép với tác vụ đang xét được hiển thị bằng màu đỏ. Xét các tác vụ mà có tác vụ khác hỗ trợ như T_5, T_6, T_7 và T_9, tác vụ hỗ trợ cho các tác vụ đó luôn được Ma^{2}BEA chọn nhiều nhất. Xét các tác vụ chắc chắn không hỗ trợ nhau như T_1, T_2 và T_3, có thể thấy T_1 có cực trị toàn cục tại $\{(o_1, o_2, \ldots, o_D) = (0, 0, \ldots, 0), \mathbf{x} \in [-100, 100]^D, D = 50\}$ ở xa so với cực trị toàn cục của T_2 tại $\{(o_1, o_2, \ldots, o_D) = (80, 80, \ldots, 80), \mathbf{x} \in [-100, 100]^D, D = 50\}$ và xa cực trị toàn cục của T_3 tại $\{(o_1, o_2, \ldots, o_D) = (-80, -80, \ldots, -80), \mathbf{x} \in [-100, 100]^D, D = 50\}$ Vậy nên, số lần T_1 ghép với T_2 và T_3 tại mỗi thế hệ đều không cao (Hình 5.2a). Hơn nữa, vì cực trị toàn cục của T_2 và T_3 ở hai phía khác nhau của không gian tìm kiếm nên ở

Hình 5.2b, Ma^{2}BEA ghép T_2 với T_3 ít nhất. Hiện tượng tương tự cũng diễn ra với T_3, khi Ma^{2}BEA ghép T_2 cho T_3 ít nhất.

Tuy nhiên, thuật toán ghép cặp sai tác vụ $T_{10}(Rastrigin)$ với $T_1(Sphere)$ và $T_5(Rosenbrock)$. Nguyên nhân là vì hàm *Rastrigin* của T_{10} nhiều cực bộ địa phương, và khi các cá thể của quần thể giải tác vụ khó T_{10} kết hợp với cá thể giải tác vụ dễ hơn như T_1 hoặc T_5, thì thường các cá thể của T_1 hoặc T_5 đã về cực trị toàn cục. Mà một phần nghiệm tối ưu toàn cục của T_1 hoặc T_5 lại nằm đúng những vị trí cực trị địa phương của T_{10}, vậy nên T_{10} giải bởi Ma^{2}BEA kém hơn so với các thuật toán khác. Nhưng, nhìn chung thì việc ghép cặp của Ma^{2}BEA trong các trường hợp thông thường khác là đúng như theo kỳ vọng định nghĩa bởi bộ dữ liệu MaTO-10.



(a) Tác vụ 1 (b) Tác vụ 2 (c) Tác vụ 3

(d) Tác vụ 4 (e) Tác vụ 5 (f) Tác vụ 6

(g) Tác vụ 7 (h) Tác vụ 8 (i) Tác vụ 9

(j) Tác vụ 10

Hình 5.2: Số lần trung bình một tác vụ ghép cặp với tác vụ khác trên mỗi mỗi thế hệ, sử dụng cơ chế ghép cặp đề xuất tại Ma^{2}BEA. Tác vụ để ghép cặp lý tưởng được vẽ với màu đỏ

5.2 Bài toán tối ưu đa tác vụ với 50 tác vụ, không biết trước mối quan hệ giữa các tác vụ

5.2.1 Mô tả bộ dữ liệu

Bộ dữ liệu Many-task Optimization (50 tasks) (MaTO-50) là bộ dữ liệu cung cấp 10 tập hàm đánh giá với độ khó đa dạng. Mỗi tập hàm đánh giá chứa định nghĩa của 50 tác vụ. Mỗi tác vụ là một hàm đánh giá đơn mục tiêu, được cấu thành từ các hàm cơ bản từ Công thức 5.1 đến Công thức 5.7. Các hàm sẽ được hiện diện đều trong 50 tác vụ. Các tác vụ cũng có cực trị toàn cục khác nhau một cách ngẫu nhiên, và cực trị toàn cục đó được định nghĩa sẵn trong bộ dữ liệu (`http://www.bdsc.site/websites/MTO/WCCI-SO-Manytask-Benchmarks.rar`). Bộ dữ liệu này được đề xuất tại cuộc thi về tiến hóa đa nhiệm tại hai hội thảo đầu ngành tính toán tiến hóa vào năm 2020 là WCCI 2020 và GECCO 2020.

Như trong Bảng 5.3, bộ dữ liệu có thể được chia làm ba nhóm chính. Nhóm thứ nhất bao gồm ba bộ hàm đánh giá B_1, B_2, B_3, mỗi bộ này đều chứa 50 tác vụ có cũng một dạng hàm đánh giá, chỉ khác nhau mỗi cực trị toàn cục. Tương tự, nhóm thứ hai bao gồm bộ B_4, B_5, B_6, B_7, mỗi bộ chứa ba dạng hàm cơ bản, chia đều cho 50 (tức có từ 16 đến 17 tác vụ ứng với một trong ba hàm). Nhóm thứ ba gồm các bộ còn lại, mỗi bộ chứa nhiều hơn năm hàm cơ bản. Như vậy, càng về các bộ dữ liệu sau, độ tương đồng về không gian tìm kiếm của các tác vụ càng lớn, yêu cầu thuật toán tiến hóa đa nhiệm phải phân nhóm cho các tác vụ một cách tốt hơn.

5.2.2 Cài đặt thực nghiệm

Việc cài đặt thực nghiệm và đánh giá kết quả được thực hiện tương tự với Mục 5.1.2, chỉ khác mỗi các hàm đánh giá. Tuy nhiên, vì bộ dữ liệu MaTO-50 chứa nhiều tác vụ hơn, và ta không biết trước mối quan hệ giữa các tác vụ như bộ dữ liệu MaTO-10 nên phần phân tích về thành phần ghép cặp tác vụ sẽ được bỏ qua. Thời gian chạy của các thuật toán tại hai bộ dữ liệu sẽ được gộp lại trình bày và so sánh tại mục phân tích kết quả bên dưới.

5.2.3 Phân tích kết quả thực nghiệm

Ở mục này, kết quả của việc sử dụng 4 thuật toán tiến hóa đa nhiệm để giải 10 bộ dữ liệu trong tập dữ liệu thử nghiệm MaTO-50 sẽ được trình bày chi tiết. Bảng 5.4 diễn giải số lượng tác vụ mà mỗi thuật toán giải được nghiệm có giá trị hàm mục tiêu tốt nhất, và giải tốt nhất một cách đáng kể so với các thuật toán khác. Nhìn chung, thuật toán đề xuất giải tốt hơn các thuật toán khác một cách đáng kể ở 6/10 bộ dữ liệu thử nghiệm. Tại cả 10 bộ dữ liệu thử nghiệm, Ma^{2}BEA đều có các tác vụ giải tốt hơn các thuật toán khác.

Ở nhóm thứ nhất, nhóm mà gồm ba bộ dữ liệu tạo thành từ một hàm tối ưu duy nhất, thuật toán đề xuất chạy tốt nhất trên bộ B_1 và B_2. Cấu trúc mới giúp cho việc khai phá không gian tìm kiếm nhanh hơn của Ma^{2}BEA đã giúp thuật toán này giải tối ưu tuyệt đối cho các hàm đơn cực trị như *Sphere* (bộ B_1) hoặc *Rosenbrock* (bộ B_2). Với hàm có nhiều cục bộ địa phương như *Rastrigin*, thuật toán có thể mắc lỗi tương tự như ở tác vụ T_{10} bộ MaTO-10, là nghiệm tối ưu của một tác vụ cho đi lại rơi đúng vào cục bộ địa phương của tác vụ nhận về. Tuy nhiên, Ma^{2}BEA vẫn có giá trị hàm mục tiêu trung bình thuộc háng tốt nhất ở 8/50 tác vụ ở bộ B_3 này.

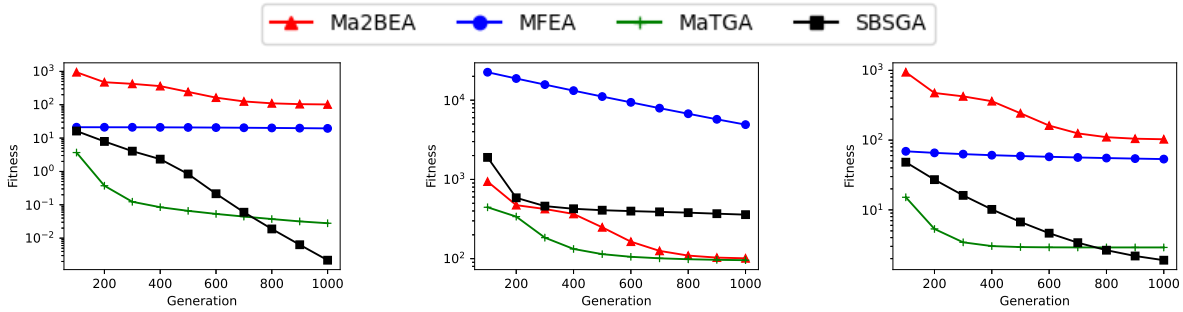
Tác vụ	Ma^{2}BEA	MFEA	MaTGA	SBSGA
T_1	**3.72E-05**	1.31E+00 (−)	2.45E-04 (−)	5.58E-04 (−)
T_2	**6.48E-06**	1.27E+00 (−)	4.75E-04 (−)	6.10E-05 (−)
T_3	**4.35E-07**	1.17E+00 (−)	**0.00E+00** (≈)	2.00E-06 (−)
T_4	**7.09E-13**	3.37E+00 (−)	1.00E-06 (−)	1.60E-05 (−)
T_5	1.74E+01	8.15E+02 (−)	**2.16E-02** (+)	3.29E-02 (+)
T_6	**7.03E-04**	1.99E+01 (−)	3.61E-03 (−)	**8.59E-04** (≈)
T_7	1.00E-02	1.05E+01 (−)	**5.57E-04** (+)	1.60E-03 (+)
T_8	1.89E+02	2.34E+03 (−)	**3.40E-02** (−)	4.73E-02 (−)
T_9	**4.00E-07**	4.11E+02 (−)	4.52E-03 (−)	4.58E-03 (−)
T_{10}	2.74E+01	**1.54E+01** (+)	1.57E+01 (+)	3.41E+01 (−)

Bảng 5.2: Giá trị hàm đánh giá của các thuật toán tiến hóa đa nhiệm trên từng tác vụ, lấy trung bình sau 30 lần chạy. $(-, +, and \approx$ đặt cạnh thuật toán nào thể hiện là thuật toán đó lần lượt kém đáng kể, tốt đáng kể hoặc tương đương với Ma^{2}BEA, sau khi kiểm định bằng phép thống kê phi tham số Wilcoxon signed-rank test với độ tin cậy tại $\alpha = 0.05$)

Bộ hàm đánh giá	**Danh sách hàm cơ bản**
B_1	*Sphere*
B_2	*Rosenbrock*
B_3	*Rastrigin*
B_4	*Sphere*, *Rosenbrock*, *Ackley*
B_5	*Rastrigin*, *Griewank*, *Weierstrass*
B_6	*Rosenbrock*, *Griewank*, *Schwefel*
B_7	*Ackley*, *Rastrigin*, *Weierstrass*
B_8	*Rosenbrock*, *Ackley*, *Rastrigin*, *Griewank*, *Weierstrass*
B_9	*Rosenbrock*, *Ackley*, *Rastrigin*, *Griewank*, *Weierstrass*, *Schwefel*
B_{10}	*Ackley*, *Rastrigin*, *Griewank*, *Weierstrass*, *Schwefel*

Bảng 5.3: Danh sách các hàm ở mỗi bộ dữ liệu đánh giá

Ở hai nhóm còn lại, Ma^{2}BEA đều là thuật toán tốt nhất trên hai bộ dữ liệu. Hình 5.3 minh họa trường hợp xấu nhất khi cả ba hàm tối ưu tại bộ B_7 đều là hàm khó. Hàm *Ackley* có một cực trị toàn cục nhưng phần lớn không gian tìm kiếm là dị biệt, có đạo hàm bằng 0 và không cho thuật toán tối ưu hướng để dò về khu vực nhỏ ở xung quanh cực trị toàn cục. Hai hàm còn lại là *Rastrigin* và *Weierstrass* thì có nhiều cực trị địa phương. Vậy nên không có tác vụ nào dễ đi trước để cải thiện các tác vụ khỏ, như giả định của Ma^{2}BEA. Ba bộ dữ liệu còn lại của nhóm hai đều có các hàm dễ như *Sphere*, *Rosenbrock* hoặc có độ khó trung bình như *Griewank*, nên kết quả của Ma^{2}BEA tốt hơn. Riêng B_4, Ma^{2}BEA tốt chỉ sau SBSGA. Xét nhóm thứ ba, Hình 5.4 minh họa một trong những trường hợp tốt nhất của Ma^{2}BEA. Trong bộ dữ liệu này, kể từ sau thế hệ thứ 500, MaTGA đã dần hội tụ về cục bộ, còn Ma^{2}BEA vẫn tiếp tục cho kết quả tốt hơn với tốc độ duy trì nhanh như ban đầu. Ma^{2}BEA cũng giải tốt hơn đáng kể so với SBSGA, trừ mỗi hàm *Griewank*. Ngoài ra, Ma^{2}BEA tốt hơn rất nhiều so với MFEA. MFEA ghép cặp các tác vụ một cách ngẫu nhiên, nên giá trị hàm đánh giá thường đi ngang và không giảm nhiều.

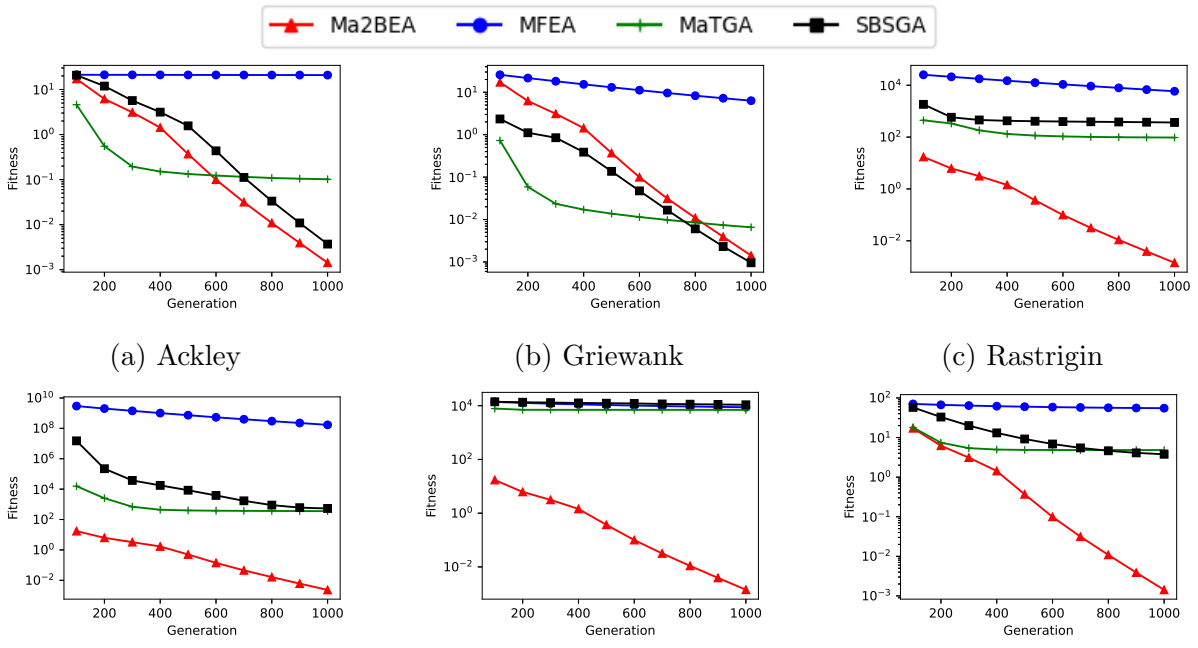


(a) Ackley (b) Rastrigin (c) Weierstrass

Hình 5.3: Biểu đồ hội tụ của 4 thuật toán trên bộ hàm đánh giá B_7, thể hiện giá trị hàm đánh giá trung bình trong quá trình tiến hóa trải dài 1000 thế hệ. B_7 là một trong những bộ mà Ma^{2}BEA không phải là thuật toán tốt nhất.

Benchmark	Ma^{2}BEA	MFEA	MaTGA	SBSGA
B_1	**50 (50)**	0 (0)	0 (0)	0 (0)
B_2	**50 (50)**	0 (0)	0 (0)	0 (0)
B_3	8 (0)	0 (0)	**42 (8)**	0 (0)
B_4	17 (9)	0 (0)	0 (0)	**33 (33)**
B_5	**50 (50)**	0 (0)	0 (0)	0 (0)
B_6	**50 (50)**	0 (0)	0 (0)	0 (0)
B_7	3 (0)	0 (0)	14 (2)	**33 (28)**
B_8	**30 (28)**	0 (0)	0 (0)	20 (20)
B_9	**42 (42)**	0 (0)	0 (0)	8 (8)
B_{10}	8 (6)	2 (0)	11 (10)	**29 (23)**

Bảng 5.4: Bảng so sánh độ hiệu quả của các thuật toán trên 10 tập hàm đánh giá, mỗi tập có 50 hàm đánh giá của bộ dữ liệu MaTO-50, thống kê lại sau 30 lần chạy. Giá trị tại mỗi ô thể hiện số lần thuật toán tại cột đó tốt hơn tất cả các thuật toán khác trên bộ dữ liệu tại hàng đó. Giá trị trong ngoặc đơn thể hiện số lần thuật toán tại đó được kiểm định là tốt hơn đáng kể sử dụng phép thống kê phi tham số Wilcoxon signed-rank test với độ tin cậy tại $\alpha = 0.05$.

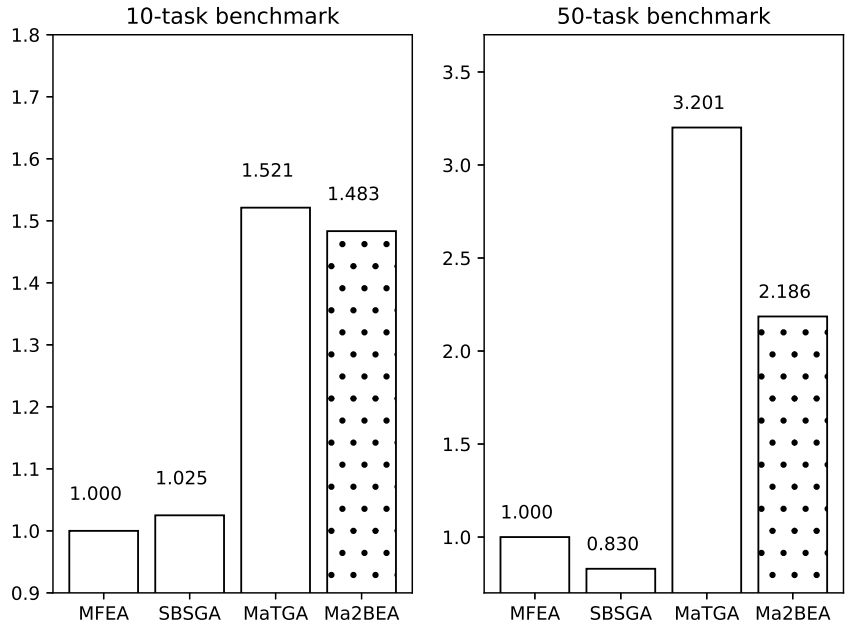


(a) Ackley (b) Griewank (c) Rastrigin

(d) Rosenbrock (e) Schwefel (f) Weierstrass

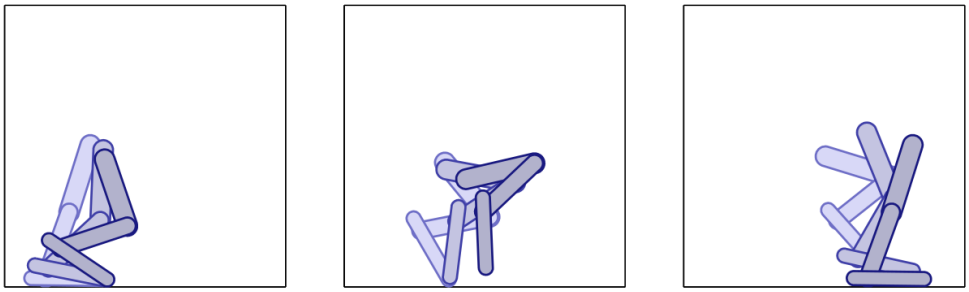
Hình 5.4: Biểu đồ hội tụ của 4 thuật toán trên bộ hàm đánh giá B_9, thể hiện giá trị hàm đánh giá trung bình trong quá trình tiến hóa trải dài 1000 thế hệ. B_9 là một trong những bộ mà Ma^{2}BEA là thuật toán tốt nhất.

Như đã trình bày ở Chương 3, MaTGA có quá nhiều tham số cần phải tinh chỉnh. Bộ tham số của MaTGA chỉ chạy tốt trên MaTO-10, còn đem nguyên bộ tham số đó áp vào bộ MaTO-50 này thì kết quả thiếu cạnh tranh hơn hẳn, khi MaTGA chỉ tốt nhất ở 1/10. Ma^{2}BEA không có tham số nào khác ngoài $rmp = 0.3$, tự thích nghi dựa vào dữ liệu và luôn duy trì được tốt nhất 6/10 ở cả hai bộ dữ liệu.



Hình 5.5: Thời gian chạy của các thuật toán tiến hóa đa nhiệm, lấy thời gian chạy của MFEA trên cũng một một trường thực nghiệm, cùng ngôn ngữ lập trình làm chuẩn

Hình 5.5 so sánh thời gian chạy giữa các thuật toán trên hai bộ dữ liệu MaTO-10 và MaTO-50. Thời gian chạy của SBSGA là tương đồng với MFEA, trong khi thời gian chạy của Ma^{2}BEA và MaTGA tăng đáng kể khi số lượng tác vụ tăng từ 10 đến 50. Tuy nhiên thời gian chạy của Ma^{2}BEA tăng chậm hơn so với MaTGA. Đó là vì MaTGA đo khoảng K^2 cặp khoảng cách Kullback Leibler Divergence của K quần thể với nhau, trong khi Ma^{2}BEA chỉ cập nhật K^2 phân phối phần thưởng. Hơn nữa, Ma^{2}BEA tính toán trên giá trị hàm mục tiêu chứ không tính toán trên cả quần thể như MaTGA. Nói tóm lại, Ma^{2}BEA cho kết quả tối ưu đa tác vụ với số lượng tác vụ lớn tương đối cạnh tranh so với các thuật toán mới nhất, với thời gian có thể chấp nhận được.



Hình 5.6: Minh họa cách hoạt động của robot Hopper [35].

5.3 Kết quả ứng dụng trên tối ưu mạng nơ ron cho học tăng cường trên môi trường Mujoco.

5.3.1 Mô tả bộ dữ liệu

Tại thí nghiệm này, tôi sử dụng bộ dữ liệu học Mujoco Multitask [27]. Trong bộ dữ liệu này có nhiều tập môi trường thử nghiệm, và tôi chọn hai tập môi trường là $Hopper - gravity$ và $Hopper - size$ để thử nghiệm các thuật toán. $Hopper$ là một robot đơn giản mô phỏng chân người với ba khớp nối như trong hình 5.6. Với mỗi tác vụ tối ưu ở mỗi môi trường thử nghiệm này, mạng nơ-ron nhận vào trạng thái $s \in \mathbb{R}^{11}$ và đưa ra hành động là $a \in \mathbb{R}^3$, tương đương với lực ở ba mối nối. Ở mỗi tập môi trương, có nhiều môi trường tương đương với nhau, cụ thể như sau:

- $Hopper - gravity$: gia tốc trọng trường của các môi trường khác nhau
    1. HopperGravityHalf-v0
    2. HopperGravityThreeQuarters-v0
    3. Hopper-v1
    4. HopperGravityOneAndQuarter-v0
    5. HopperGravityOneAndHalf-v0
- $Hopper - size$: robot có kích cỡ khác nhau
    1. HopperSmallFoot-v0
    2. HopperSmallLeg-v0
    3. HopperSmallThigh-v0
    4. HopperSmallTorso-v0
    5. HopperBigFoot-v0
    6. HopperBigLeg-v0
    7. HopperBigThigh-v0
    8. HopperBigTorso-v0
    9. Hopper-v1

5.3.2 Cài đặt thực nghiệm

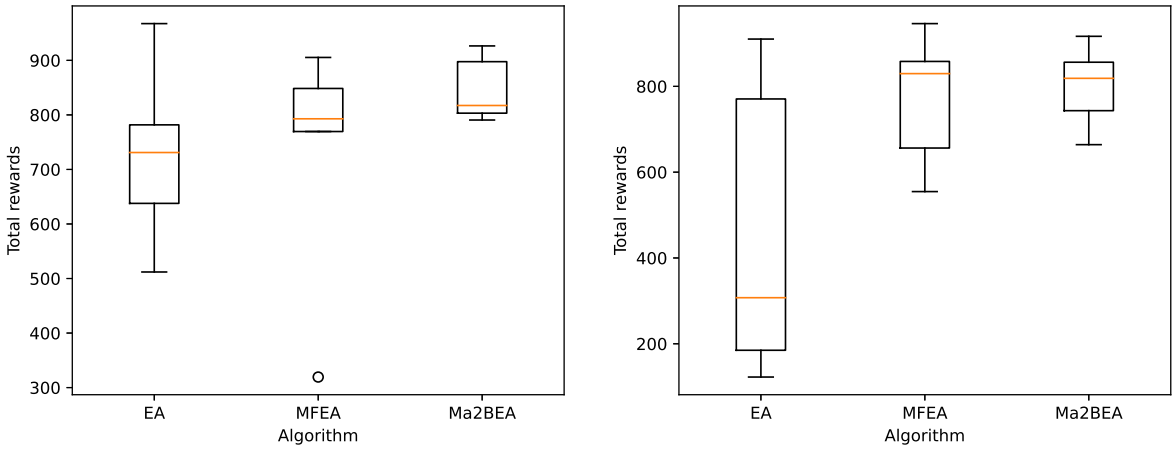
Phần lớn việc cài đặt thực nghiệm và đánh giá kết quả được thực hiện tương tự với Mục 5.1.2. Tuy nhiên, ở bộ dữ liệu Mujoco Multitask này, vì mất nhiều thời gian giả lập môi trường học tăng cường nên thời gian thực nghiệm kéo dài hơn so với MaTO-10 và MaTO-50, Ma^{2}BEA sẽ chỉ được so sánh với EA và MFEA. Không như MaTO-10 và MaTO-50, MFEA chưa được chứng minh là tốt hơn so với EA trên bộ dữ liệu Mujoco Multitask nên EA cũng được chạy thêm để so sánh.

Cả ba thuật toán đều tối ưu mạng nơ ron một lớp ẩn, với đầu vào là 11 chiều, lớp ẩn chứa 16 nút và đầu ra là 3 nút. Tổng số tham số là 243 tham số. Các mạng nơ-ron đều được mã hóa vào không gian biểu diễn chung là $[0, 1]^{243}$. Khi mang đi đánh giá, các mạng nơ-ron được giải mã lại, và dãn ra từ khoảng $[0, 1]$ đến $[-5, 5]$. Ngoài ra, vì thời gian chạy lâu nên các tham số như số thế hệ và kích thước quần thể chọn nhỏ hơn so với hai bộ dữ liệu trước là $(T, N) = (100, 10)$

5.3.3 Phân tích kết quả thực nghiệm

Phân phối của giá trị hàm đánh giá của ba thuật toán EA, MFEA, và Ma^{2}BEA được thể hiện ở hình 5.7. Dễ thấy, ở ứng dụng này, cả hai thuật toán tối ưu đa nhiệm đều cho kết quả cao hơn so với EA. Với khả năng trao đổi thông tin giữa nhiều tác vụ, giá trị tối thiểu của tổng phần thưởng trên mọi môi trường giải bằng hai thuật toán tiến hóa đa nhiệm đều cao hơn giá trị tối thiểu của tổng phần thưởng của EA. Ở EA, không có cơ chế trao đổi thông tin nên còn một vài tác vụ kém, chưa thoát khỏi vùng cục bộ với tổng phần thưởng khoảng 500 như trong bộ $Hopper - gravity$ hoặc vùng cục bộ với giá trị hàm đánh giá chỉ ở mức 200 như ở bộ $Hopper - size$.

Ta cũng dễ thấy được Ma^{2}BEA với cơ chế ghép cặp tự thích nghi vẫn cho kết quả nhỉnh hơn so với MFEA. Với cơ chế ghép cặp ngẫu nhiên thô sơ, MFEA vẫn để một tác vụ tại bộ $Hopper - gravity$ tắc ở vùng cục bộ với tổng phần thưởng chỉ ở mức 300, còn Ma^{2}BEA luôn đều đặn cho kết quả cao hơn 800 ở bộ $Hopper - gravity$, và cao hơn 600 ở bộ $Hopper - size$



(a) $Hopper - gravity$ (b) $Hopper - size$

Hình 5.7: Biểu đồ hộp (minh hoạ các giá trị nhỏ nhất, quartile thứ nhất, median, quartile thứ ba, giá trị lớn nhất) của các hàm mục tiêu của bộ Mujoco Multitask, giải bởi EA, MFEA, và Ma^{2}BEA. Giá trị được thống kê ở biểu đồ hộp là giá trị hàm mục tiêu tại thế hệ cuối cùng của các tác vụ khác nhau, giải bởi từng thuật toán trên từng bộ dữ liệu.

Kết luận

Trong luận văn này, tôi đã đề xuất một thuật toán tiến hóa đa nhiệm mới tên là Ma^{2}BEA, tự thích nghi với dữ liệu tối ưu. Giải thuật có hai điểm mới nổi bật là cơ chế ghép cặp tự động, không có tham số sử dụng mô hình Multi-Armed Bandits, và cấu trúc mới chuyển giao tri thức một cách tuần tự.

Thực nghiệm chỉ ra rằng Ma^{2}BEA với các tham số mặc định đã tự ứng biến được trên nhiền bộ dữ liệu thực nghiệm khác nhau. Ma^{2}BEA phần nhiều tốt nhất trên ba bộ dữ liệu thử nghiệm, cả tối ưu hàm số thực lẫn tối ưu mạng nơ-ron.

Bài báo trong quá trình học

- Le Tien Thanh, La Van Cuong, Ta Bao Thang, and Huynh Thi Thanh Binh. "Multi-Armed Bandits for Many-task Evolutionary Optimization". In: *2021 IEEE Congress on Evolutionary Computation (CEC)*. IEEE. 2021, pp. 1–8
- Le Van An, Le Tien Thanh, Nguyen Phi Le, Huynh Thi Thanh Binh, Akerkar Rajendra, and Yusheng Ji. "GCRINT: Network Traffic Imputation Using Graph Convolutional Recurrent Neural Network". In: *2021 IEEE International Conference on Communications*. IEEE. 2021, pp. 1–8
- Le Van An, Le Tien Thanh, Nguyen Phi Le, Huynh Thi Thanh Binh, and Yusheng Ji. "Multi-time-step Segment Routing based Traffic Engineering Leveraging Traffic Prediction". In: *2021 IFIP/IEEE International Symposium on Integrated Network Management*. IEEE. 2021, pp. 1–8

Tài liệu tham khảo

[1] Cora M Dzubak et al. "Multitasking: The good, the bad, and the unknown". In: *The Journal of the Association for the Tutoring Profession* 1.2 (2008), pp. 1–12.

[2] Yew-Soon Ong and Abhishek Gupta. "Evolutionary multitasking: a computer science view of cognitive multitasking". In: *Cognitive Computation* 8.2 (2016), pp. 125–142.

[3] Huynh Thi Thanh Binh, Pham Dinh Thanh, Tran Ba Trung, and Le Phuong Thao. "Effective multifactorial evolutionary algorithm for solving the cluster shortest path tree problem". In: *2018 IEEE Congress on Evolutionary Computation (CEC)*. IEEE. 2018, pp. 1–8.

[4] Tran Ba Trung, Le Tien Thanh, Ly Trung Hieu, Pham Dinh Thanh, and Huynh Thi Thanh Binh. "Multifactorial evolutionary algorithm for clustered minimum routing cost problem". In: *Proceedings of the Tenth International Symposium on Information and Communication Technology*. 2019, pp. 170–177.

[5] Huynh Thi Thanh Binh, Ta Bao Thangy, Nguyen Binh Long, Ngo Viet Hoang, and Pham Dinh Thanh. "Multifactorial Evolutionary Algorithm for Inter-Domain Path Computation under Domain Uniqueness Constraint". In: *2020 IEEE Congress on Evolutionary Computation (CEC)*. IEEE. 2020, pp. 1–8.

[6] Pham Dinh Thanh, Huynh Thi Thanh Binh, and Tran Ba Trung. "An efficient strategy for using multifactorial optimization to solve the clustered shortest path tree problem". In: *Applied Intelligence* 50.4 (2020), pp. 1233–1258.

[7] Huynh Thi Thanh Binh, Ta Bao Thang, Nguyen Duc Thai, and Pham Dinh Thanh. "A bi-level encoding scheme for the clustered shortest-path tree problem in multifactorial optimization". In: *Engineering Applications of Artificial Intelligence* 100 (2021), p. 104187.

[8] Phan Thi Hong Hanh, Pham Dinh Thanh, and Huynh Thi Thanh Binh. "Evolutionary algorithm and multifactorial evolutionary algorithm on clustered shortest-path tree problem". In: *Information Sciences* 553 (2021), pp. 280–304.

[9] Ta Bao Thang, Nguyen Binh Long, Ngo Viet Hoang, and Huynh Thi Thanh Binh. "Adaptive Knowledge Transfer in Multifactorial Evolutionary Algorithm for the Clustered Minimum Routing Cost Problem". In: *Applied Soft Computing* (2021), p. 107253.

[10] Nguyen Thi Tam, Tran Quang Tuan, Huynh Thi Thanh Binh, and Ananthram Swami. "Multifactorial evolutionary optimization for maximizing data aggregation tree lifetime in wireless sensor networks". In: *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications II*. Vol. 11413. International Society for Optics and Photonics. 2020, 114130Z.

[11] Rohitash Chandra, Abhishek Gupta, Yew-Soon Ong, and Chi-Keong Goh. "Evolutionary multi-task learning for modular knowledge representation in neural networks". In: *Neural Processing Letters* 47.3 (2018), pp. 993–1009.

[12] Kavitesh Kumar Bali, Yew-Soon Ong, Abhishek Gupta, and Puay Siew Tan. "Multifactorial evolutionary algorithm with online transfer parameter estimation: MFEA-II". In: *IEEE Transactions on Evolutionary Computation* 24.1 (2019), pp. 69–83.

[13] Abhishek Gupta and Yew-Soon Ong. *Memetic computation: the mainspring of knowledge transfer in a data-driven optimization era*. Vol. 21. Springer, 2018.

[14] Jing Tang, Yingke Chen, Zixuan Deng, Yanping Xiang, and Colin Paul Joy. "A Group-based Approach to Improve Multifactorial Evolutionary Algorithm." In: *IJCAI*. 2018, pp. 3870–3876.

[15] Rung-Tzuo Liaw and Chuan-Kang Ting. "Evolutionary manytasking optimization based on symbiosis in biocoenosis". In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 33. 01. 2019, pp. 4295–4303.

[16] Yongliang Chen, Jinghui Zhong, Liang Feng, and Jun Zhang. "An adaptive archive-based evolutionary framework for many-task optimization". In: *IEEE Transactions on Emerging Topics in Computational Intelligence* 4.3 (2019), pp. 369–384.

[17] Abhishek Gupta and Yew-Soon Ong. "Multitask Knowledge Transfer Across Problems". In: *Memetic Computation*. Springer, 2019, pp. 83–92.

[18] Serkan Cabi, Sergio Gómez Colmenarejo, Matthew W Hoffman, Misha Denil, Ziyu Wang, and Nando Freitas. "The intentional unintentional agent: Learning to solve many continuous control tasks simultaneously". In: *Conference on Robot Learning*. PMLR. 2017, pp. 207–216.

[19] Kevin Swersky, Jasper Snoek, and Ryan Prescott Adams. "Multi-task bayesian optimization". In: (2013).

[20] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P Adams, and Nando De Freitas. "Taking the human out of the loop: A review of Bayesian optimization". In: *Proceedings of the IEEE* 104.1 (2015), pp. 148–175.

[21] Abhishek Gupta, Yew-Soon Ong, and Liang Feng. "Multifactorial evolution: toward evolutionary multitasking". In: *IEEE Transactions on Evolutionary Computation* 20.3 (2015), pp. 343–357.

[22] William R Thompson. "On the likelihood that one unknown probability exceeds another in view of the evidence of two samples". In: *Biometrika* 25.3/4 (1933), pp. 285–294.

[23] Tor Lattimore and Csaba Szepesvári. *Bandit algorithms*. Cambridge University Press, 2020.

[24] Michael R Zhang, James Lucas, Geoffrey Hinton, and Jimmy Ba. "Lookahead optimizer: k steps forward, 1 step back". In: *arXiv preprint arXiv:1907.08610* (2019).

[25] Reem Alzahabi and Mark W Becker. "The association between media multitasking, task-switching, and dual-task performance." In: *Journal of Experimental Psychology: Human Perception and Performance* 39.5 (2013), p. 1485.

[26] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[27] Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. "Deep reinforcement learning that matters". In: *Thirty-Second AAAI Conference on Artificial Intelligence*. 2018.

[28] Nguyen Quoc Tuan, Ta Duy Hoang, and Huynh Thi Thanh Binh. "A guided differential evolutionary multi-tasking with powell search method for solving multi-objective continuous optimization". In: *2018 IEEE Congress on Evolutionary Computation (CEC)*. IEEE. 2018, pp. 1–8.

[29] Jinghui Zhong, Liang Feng, Wentong Cai, and Yew-Soon Ong. "Multifactorial genetic programming for symbolic regression problems". In: *IEEE transactions on systems, man, and cybernetics: systems* 50.11 (2018), pp. 4492–4505.

[30] Zedong Tang and Maoguo Gong. "Adaptive multifactorial particle swarm optimisation". In: *CAAI Transactions on Intelligence Technology* 4.1 (2019), pp. 37–46.

[31] Kalyanmoy Deb, Ram Bhushan Agrawal, et al. "Simulated binary crossover for continuous search space". In: *Complex systems* 9.2 (1995), pp. 115–148.

[32] Kalyanmoy Deb and Debayan Deb. "Analysing mutation schemes for real-parameter genetic algorithms". In: *International Journal of Artificial Intelligence and Soft Computing* 4.1 (2014), pp. 1–28.

[33] Robert Hinterding. "Gaussian mutation and self-adaption for numeric genetic algorithms". In: *Proceedings of 1995 IEEE International Conference on Evolutionary Computation*. Vol. 1. IEEE. 1995, p. 384.

[34] Donald W Zimmerman. "Comparative power of Student t test and Mann-Whitney U test for unequal sample sizes and variances". In: *The Journal of Experimental Education* 55.3 (1987), pp. 171–174.

[35] Tom Erez, Yuval Tassa, and Emanuel Todorov. "Infinite horizon model predictive control for nonlinear periodic tasks". In: *Manuscript under review* 4 (2011).