Reinforcement Learning – Học Tăng Cường

Nguyễn Quản Bá Hồng*

Ngày 22 tháng 9 năm 2025

Tóm tắt nội dung

This text is a part of the series *Some Topics in Advanced STEM & Beyond*:
URL: https://nqbh.github.io/advanced_STEM/.
Latest version:

- *Reinforcement Learning – Học Tăng Cường.*
  PDF: URL: .pdf.
  TEX: URL: .tex.
- .
  PDF: URL: .pdf.
  TEX: URL: .tex.

Mục lục

1 Basic Reinforcement Learning

1.1 Richard S. Sutton, Andrew G. Barto. Reinforcement Learning: An Introduction. 2e. 2020

- Preface to 2e. 20 years since publication of 1e of this book have seen tremendous progress in AI, propelled in large part by advances in ML, including advances in RL. Although impressive computational power that became available is responsible for some of these advances, new developments in theory & algorithms have been driving forces as well. In face of this progress, a 2e of 1998 book was long overdue, & finally began project in 2012. Goal for 2e was same as goal for 1e: provide a clear & simple account of key ideas & algorithms of RL that is accessible to readers in all related disciplines. Edition remains an introduction, & retain a focus on core, online learning algorithms. This edition includes some new topics that rose to importance over intervening years, & expanded coverage of topics that we now understand better. But made no attempt to provide comprehensive coverage of field, which has exploded in many different directions. Apologize for having to leave out all but a handful of these contributions.

  – 20 năm kể từ khi xuất bản ấn bản đầu tiên của cuốn sách này, chúng ta đã chứng kiến những tiến bộ vượt bậc trong lĩnh vực AI, phần lớn nhờ vào những tiến bộ trong Học máy (ML), bao gồm cả những tiến bộ trong Học máy (RL). Mặc dù sức mạnh tính toán ấn tượng đã góp phần tạo nên 1 số tiến bộ này, nhưng những phát triển mới về lý thuyết & thuật toán cũng là động lực thúc đẩy. Trước những tiến bộ này, ấn bản thứ 2 của cuốn sách năm 1998 đã bị trì hoãn từ lâu, & cuối cùng đã được khởi động dự án vào năm 2012. Mục tiêu của ấn bản thứ 2 cũng giống như mục tiêu của ấn bản thứ 1: cung cấp 1 bản tóm tắt rõ ràng & đơn giản về những ý tưởng chính & thuật toán của Học máy (RL), dễ hiểu cho độc giả trong mọi lĩnh vực liên quan. Ấn bản này vẫn là phần giới thiệu, & tập trung vào các thuật toán học tập trực tuyến cốt lõi. Ấn bản này bao gồm 1 số chủ đề mới đã trở nên quan trọng trong những năm tiếp theo, & mở rộng phạm vi bao quát các chủ đề mà giờ đây chúng ta đã hiểu rõ hơn. Tuy nhiên, ấn bản này không cố gắng cung cấp phạm vi bao quát toàn diện về lĩnh vực này, vốn đã bùng nổ theo nhiều hướng khác nhau. Xin lỗi vì đã phải bỏ qua hầu hết những đóng góp này.

*A scientist- & creative artist wannabe, a mathematics & computer science lecturer of Department of Artificial Intelligence & Data Science (AIDS), School of Technology (SOT), UMT Trường Đại học Quản lý & Công nghệ TP.HCM, Hồ Chí Minh City, Việt Nam.
E-mail: nguyenquanbahong@gmail.com & hong.nguyenquanba@umt.edu.vn. Website: https://nqbh.github.io/. GitHub: https://github.com/NQBH.

As in 1e, chose not to produce a rigorous formal treatment of RL, or to formulate it in most general terms. However, our deeper understanding of some topics since 1e required a bit more mathematics to explain; have set off more mathematical parts in shaded boxes that non-mathematically-inclined may choose to skip. Also use a slightly different notation used in 1e. In teaching, have found: new notation helps to address some common points of confusion. It emphasizes difference between random variables, denoted with capital letters, & their instantiations, denoted in lower case. E.g., state, action, & reward at time step t are denoted S_t, A_t, R_t, while their possible values might be denoted s, a, r. Along with this, natural to use lower case for value functions (e.g., v_π) & restrict capitals to their tabular estimates (e.g., $Q_t(s, a)$). Approximate value functions are deterministic functions of random parameters & are thus also in lower case (e.g., $\hat{v}(s, \mathbf{w}_t) \approx v_\pi(s)$). Vectors, e.g. weight vector $\mathbf{w}_t$ (formerly $\boldsymbol{\theta}_t$) & feature vector $\mathbf{x}_t$ (formerly $\boldsymbol{\phi}_t$), are bold & written in lowercase even if they are random variables. Uppercase bold is reserved for matrices. In 1e used special notations, $\mathcal{P}^a_{ss'}, \mathcal{R}^a_{ss'}$, for transition probabilities & expected rewards. 1 weakness of that notation: it still did not fully characterize dynamics of rewards, giving only their expectations, which is sufficient for dynamic programming but not for RL. Another weakness is excess of subscripts & superscripts. In this edition, use explicit notation of $p(s', r|s, a)$ for joint probability for next state & reward given current state & action. All changes in notation are summarized in a table on p. 6.

– Giống như trong 1e, chúng tôi đã chọn không đưa ra 1 cách xử lý chính thức nghiêm ngặt về RL, hoặc xây dựng nó theo những thuật ngữ chung nhất. Tuy nhiên, sự hiểu biết sâu sắc hơn của chúng tôi về 1 số chủ đề kể từ 1e đòi hỏi nhiều kiến thức toán học hơn để giải thích; đã đặt nhiều phần toán học hơn trong các ô tô đậm mà những người không có thiên hướng toán học có thể chọn bỏ qua. Ngoài ra, chúng tôi cũng sử dụng 1 ký hiệu hơi khác được sử dụng trong 1e. Trong giảng dạy, chúng tôi đã tìm thấy: ký hiệu mới giúp giải quyết 1 số điểm dễ nhầm lẫn phổ biến. Nó nhấn mạnh sự khác biệt giữa các biến ngẫu nhiên, được ký hiệu bằng chữ in hoa, & các thể hiện của chúng, được ký hiệu bằng chữ thường. Ví dụ: trạng thái, hành động, & phần thưởng tại bước thời gian t được ký hiệu là S_t, A_t, R_t, trong khi các giá trị khả dĩ của chúng có thể được ký hiệu là s, a, r. Cùng với điều này, việc sử dụng chữ thường cho các hàm giá trị (ví dụ: v_π) & giới hạn chữ hoa trong các ước lượng dạng bảng của chúng là điều tự nhiên (ví dụ: $Q_t(s, a)$). Hàm giá trị gần đúng là hàm xác định của các tham số ngẫu nhiên & do đó cũng được viết thường (ví dụ: $\hat{v}(s, \mathbf{w}_t) \approx v_\pi(s)$). Các vectơ, ví dụ: vectơ trọng số $\mathbf{w}_t$ (trước đây là $\boldsymbol{\theta}_t$) & vectơ đặc trưng $\mathbf{x}_t$ (trước đây là $\boldsymbol{\phi}_t$), được viết đậm & bằng chữ thường ngay cả khi chúng là các biến ngẫu nhiên. Chữ in hoa đậm được dành riêng cho các ma trận. Trong 1e đã sử dụng các ký hiệu đặc biệt, $\mathcal{P}^a_{ss'}, \mathcal{R}^a_{ss'}$, cho xác suất chuyển đổi & phần thưởng mong đợi. 1 điểm yếu của ký hiệu đó: nó vẫn không mô tả đầy đủ động lực của phần thưởng, chỉ đưa ra kỳ vọng của chúng, đủ cho lập trình động nhưng không đủ cho RL. Một điểm yếu khác là có quá nhiều chỉ số dưới & chỉ số trên. Trong phiên bản này, hãy sử dụng ký hiệu rõ ràng $p(s', r|s, a)$ cho xác suất kết hợp cho trạng thái tiếp theo & phần thưởng dựa trên trạng thái hiện tại & hành động. Tất cả các thay đổi về ký hiệu được tóm tắt trong bảng ở trang 6.

2e is significantly expanded, & its top-level organization has been changed. After introductory 1st chap, 2e is divided into 3 new parts. 1st part (Chaps. 2–8) treats as much of RL as possible without going beyond tabular case for which exact solutions can be found. Cover both learning & planning methods for tabular case, as well as their unification in n-step methods & in Dyna. Many algorithms presented in this part are new to 2e, including UCB, Expected Sarsa, Double learning, tree-backup, $Q(\sigma)$, RTDP, & MCTS. Doing tabular case 1st, & thoroughly, enables core ideas to be developed in simplest possible setting. 2nd part of book (Chaps. 9–13) is then devoted to extending ideas to function approximation. It has new sects on ANNs, Fourier basis, LSTD, kernel-based methods, Gradient-TD & Emphatic-TD methods, average-reward methods, true online TD(λ), & policy-gardient methods. 2e significantly expands treatment of off-policy learning, 1st for tabular case in Chaps. 5–7, then with function approximation in Chaps. 11–12. Another change: 2e separates forward-view idea of n-step boostrapping (now treated more fully in Chap. 7) from backward-view idea of eligibility traces (now treated independently in Chap. 12). 3rd part of book has large new chaps on RL's relationships to psychology (Chap. 14) & neuroscience (Chap. 15), as well as an updated case-studies chap including Atari game playing, Watson's wagering strategy, & Go playing programs AlphaGo & AlphaGo Zero (Chap. 16). Still, out of necessity we have included only a small subset of all that has been done in field. Our choices reflect our long-standing interests in inexpensive model-free methods that should scale well to large applications. Final chap now includes a discussion of future societal impacts of RL. For better or worst, 2e is about twice as large as 1e.

– 2e is significantly expanded, & its top-level organization has been changed. After introductory 1st chap, 2e is divided into 3 new parts. 1st part (Chaps. 2–8) treats as much of RL as possible without going beyond tabular case for which exact solutions can be found. Cover both learning & planning methods for tabular case, as well as their unification in n-step methods & in Dyna. Many algorithms presented in this part are new to 2e, including UCB, Expected Sarsa, Double learning, tree-backup, $Q(\sigma)$, RTDP, & MCTS. Doing tabular case 1st, & thoroughly, enables core ideas to be developed in simplest possible setting. 2nd part of book (Chaps. 9–13) is then devoted to extending ideas to function approximation. It has new sects on ANNs, Fourier basis, LSTD, kernel-based methods, Gradient-TD & Emphatic-TD methods, average-reward methods, true online TD(λ), & policy-gardient methods. 2e significantly expands treatment of off-policy learning, 1st for tabular case in Chaps. 5–7, then with function approximation in Chaps. 11–12. Another change: 2e separates forward-view idea of n-step boostrapping (now treated more fully in Chap. 7) from backward-view idea of eligibility traces (now treated independently in Chap. 12). 3rd part of book has large new chaps on RL's relationships to psychology (Chap. 14) & neuroscience (Chap. 15), as well as an updated case-studies chap including Atari game playing, Watson's wagering strategy, & Go playing programs AlphaGo & AlphaGo Zero (Chap. 16). Still, out of necessity we have included only a small subset of all that has been done in field. Our choices reflect our long-standing interests in inexpensive model-free methods that should scale well to large applications. Final chap now includes a discussion of future societal impacts of RL. For better or worst, 2e is about twice as large as 1e.

This book is designed to be used as primary text for a 1- or 2-semester course on RL. For a 1-semester course, 1st 10 chaps should be covered in order & form a good core, to which can be added material from other chaps, from other books e.g. Bertsekas & Tsitsiklis (1996), Wiering & van Otterlo (2012), & Szepesvári (2010), or from literature, according to taste.

Depending of students' background, some additional material on online supervised learning may be helpful. Ideas of options & option models are a natural addition (Sutton, Precup & Singh, 1999). A 2-semester course can cover all chaps as well as supplementary material. Book can also be used as part of broader courses on Ml, AI, or neural networks. In this case, it may be desirable to cover only a subset of material. Recommend covering Chap. 1 for a brief overview, Chap. 2 through Sect. 2.4, Chap. 3, & then selecting sections from remaining chaps according to time & interests. Chap. 6 is most important for subset & for rest of book. A course focusing on ML or neural networks should cover Chaps. 9–10, & a course focusing on AI or planning should cover Chap. 8. Throughout book, sects & chaps that are more difficult & not essential to rest of book are marked with a *. These can be omitted on 1st reading without creating problems later on. Some exercises are also marked with a * to indicate: they are more advanced & not essential to understanding basic material of chap.

– Cuốn sách này được thiết kế để sử dụng làm tài liệu chính cho khóa học 1 hoặc 2 học kỳ về RL. Đối với khóa học 1 học kỳ, 10 chương đầu tiên nên được học theo thứ tự & tạo thành 1 cốt lõi tốt, có thể thêm tài liệu từ các chương khác, từ các cuốn sách khác, ví dụ như Bertsekas & Tsitsiklis (1996), Wiering & van Otterlo (2012) & Szepesvári (2010) hoặc từ tài liệu tham khảo, tùy theo sở thích. Tùy thuộc vào nền tảng của sinh viên, 1 số tài liệu bổ sung về học tập có giám sát trực tuyến có thể hữu ích. Ý tưởng về các tùy chọn & mô hình tùy chọn là 1 sự bổ sung tự nhiên (Sutton, Precup & Singh, 1999). Một khóa học 2 học kỳ có thể bao gồm tất cả các chương cũng như tài liệu bổ sung. Sách cũng có thể được sử dụng như 1 phần của các khóa học rộng hơn về ML, AI hoặc mạng nơ-ron. Trong trường hợp này, có thể chỉ nên học 1 phần nhỏ tài liệu. Đề xuất học Chương 1 để có cái nhìn tổng quan ngắn gọn, Chương Từ Chương 2 đến Chương 2.4, Chương 3, & sau đó chọn các phần từ các chương còn lại theo thời gian & sở thích. Chương 6 là quan trọng nhất đối với tập con & cho phần còn lại của sách. Một khóa học tập trung vào Học máy hoặc mạng nơ-ron nên bao gồm các Chương 9–10, & 1 khóa học tập trung vào Trí tuệ nhân tạo hoặc Lập kế hoạch nên bao gồm Chương 8. Xuyên suốt sách, các chương & khó hơn & không cần thiết cho phần còn lại của sách được đánh dấu bằng dấu *. Những chương này có thể được bỏ qua khi đọc lần đầu mà không gây ra vấn đề gì sau này. Một số bài tập cũng được đánh dấu bằng dấu * để chỉ ra: chúng nâng cao hơn & không cần thiết cho việc hiểu nội dung cơ bản của chương.

- Preface to 1e. 1st came to focus on what is now known as RL in late 1979. Both at University of Massachusetts, work on 1 of earliest projects to revive idea that networks of neuronlike adaptive elements might prove to be a promising approach to artificial adaptive intelligence. Project explored "heterostatic theory of adaptive systems" developed by A. Harry Klopf. Harry's work was a rich source of ideas, & we were permitted to explore them critically & compare them with long history of prior work in adaptive systems. Our task become 1 of teasing ideas apart & understanding their relationships & relative importance. This continues today, but in 1979 we came to realize: perhaps simplest of ideas, which had long been taken for granted, had received surprisingly little attention from a computational perspective. This was simply idea of a learning system that *wants* something, that adapts its behavior in order to maximize a special signal from its environment. This was idea of a "hedonistic" learning system, or, as we would say now, idea of RL.

  – Đầu tiên, chúng tôi tập trung vào cái mà ngày nay được gọi là RL vào cuối năm 1979. Cả hai đều tại Đại học Massachusetts, làm việc trên 1 trong những dự án đầu tiên nhằm khôi phục ý tưởng rằng mạng lưới các yếu tố thích ứng giống nơ-ron có thể là 1 phương pháp tiếp cận đầy hứa hẹn cho trí tuệ nhân tạo thích ứng. Dự án đã khám phá "lý thuyết dị biệt về các hệ thống thích ứng" do A. Harry Klopf phát triển. Công trình của Harry là 1 nguồn ý tưởng phong phú, & chúng tôi được phép khám phá chúng 1 cách phê phán & so sánh chúng với lịch sử lâu dài của các công trình về hệ thống thích ứng trước đây. Nhiệm vụ của chúng tôi là phân tích các ý tưởng & hiểu mối quan hệ & tầm quan trọng tương đối của chúng. Điều này vẫn tiếp tục cho đến ngày nay, nhưng vào năm 1979, chúng tôi nhận ra: có lẽ những ý tưởng đơn giản nhất, vốn từ lâu đã được coi là hiển nhiên, lại nhận được rất ít sự chú ý từ góc độ tính toán. Đây chỉ đơn giản là ý tưởng về 1 hệ thống học tập *nó muốn* 1 thứ gì đó, điều chỉnh hành vi của nó để tối đa hóa 1 tín hiệu đặc biệt từ môi trường của nó. Đây là ý tưởng về 1 hệ thống học tập "hưởng thụ", hay như chúng ta thường nói bây giờ, ý tưởng về RL.

  Like others, had a sense: RL had been thoroughly explored in early days of cybernetics & AI. On closer inspection, though, found: it had been explored only slightly. While RL had clearly motivated some of earliest computational studies of learning, most of these researchers had gone on to other things, e.g. pattern classification, supervised learning, & adaptive control, or they had abandoned study of learning altogether. As a result, special issues involved in learning how to get something from environment received relatively little attention. In retrospect, focusing on this idea was critical step that set this branch of research in motion. Little progress could be made in computational study of RL until it was recognized that such a fundamental idea had not yet been thoroughly explored.

  – Giống như những người khác, tôi có cảm giác: RL đã được khám phá kỹ lưỡng trong những ngày đầu của ngành điều khiển học & AI. Tuy nhiên, khi xem xét kỹ hơn, tôi thấy rằng: nó chỉ được khám phá 1 chút. Trong khi RL rõ ràng đã thúc đẩy 1 số nghiên cứu tính toán sớm nhất về học tập, hầu hết các nhà nghiên cứu này đã chuyển sang những thứ khác, ví dụ như phân loại mẫu, học có giám sát, & điều khiển thích ứng, hoặc họ đã từ bỏ hoàn toàn việc nghiên cứu về học tập. Kết quả là, các vấn đề đặc biệt liên quan đến việc học cách lấy thứ gì đó từ môi trường nhận được tương đối ít sự chú ý. Nhìn lại, việc tập trung vào ý tưởng này là bước quan trọng đưa nhánh nghiên cứu này vào hoạt động. Nghiên cứu tính toán về RL chỉ có thể đạt được rất ít tiến bộ cho đến khi người ta nhận ra rằng 1 ý tưởng cơ bản như vậy vẫn chưa được khám phá kỹ lưỡng.

  Field has come a long way since then, evolving & maturing in several directions. RL has gradually become 1 of most active research areas in ML, AI, & neural network research. Field has developed strong mathematical foundations & impressive applications. Computational study of RL is now a large field, with hundreds of active researchers around world in diverse disciplines e.g. psychology, control theory, AI, & neuroscience. Particularly important have been contributions establishing & developing relationships to theory of optimal control & dynamic programming. Overall problem of learning from interaction to

achieve goals is still far from being solved, but our understanding of it has improved significantly. Can now place component ideas, e.g. temporal-difference learning, dynamic programming, & function approximation, within a coherent perspective w.r.t. overall problem.

– Lĩnh vực này đã đi 1 chặng đường dài kể từ đó, phát triển & trưởng thành theo nhiều hướng. RL đã dần trở thành 1 trong những lĩnh vực nghiên cứu tích cực nhất trong ML, AI, & nghiên cứu mạng nơ-ron. Lĩnh vực này đã phát triển nền tảng toán học vững chắc & các ứng dụng ấn tượng. Nghiên cứu tính toán của RL hiện là 1 lĩnh vực lớn, với hàng trăm nhà nghiên cứu tích cực trên khắp thế giới trong nhiều lĩnh vực khác nhau, ví dụ như tâm lý học, lý thuyết điều khiển, AI, & khoa học thần kinh. Đặc biệt quan trọng là những đóng góp thiết lập & phát triển mối quan hệ với lý thuyết điều khiển tối ưu & lập trình động. Vấn đề chung về học hỏi từ tương tác để đạt được mục tiêu vẫn còn lâu mới được giải quyết, nhưng sự hiểu biết của chúng ta về nó đã được cải thiện đáng kể. Giờ đây, có thể đặt các ý tưởng thành phần, ví dụ như học chênh lệch thời gian, lập trình động, & xấp xỉ hàm, trong 1 quan điểm mạch lạc đối với vấn đề tổng thể.

In some sense we have been working toward this book for 30 years, & have lots of people to thank.

- Summary of notation. Capital letters are used for random variables, whereas lower case letters are used for values of random variable & for scalar functions. Quantities that are required to be real-valued vectors are written in bold & in lower case (even if random variables). Matrices are bold capitals.

- 1. Introduction. Idea that we learn by interacting with our environment is probably the 1st to occur to us when we think about nature of learning. When an infant plays, waves its arms, or looks about, it has not explicit teacher, but it does have gave a direct sensorimotor connection to its environment. Exercising this connection produces a wealth of information about cause & effect, about consequences of actions, & about what to do in order to achieve goals. Throughout our lives, such interactions are undoubtedly a major source of knowledge about our environment & ourselves. Whether we are learning to drive a car or to hold a conversation, we are acutely aware of how our environment responds to what we do, & we seek to influence what happens through our behavior. Learning from interaction is a foundational idea underlying nearly all theories of learning & intelligence.

  – Ý tưởng rằng chúng ta học bằng cách tương tác với môi trường có lẽ là ý tưởng đầu tiên xuất hiện khi chúng ta nghĩ về bản chất của việc học. Khi 1 đứa trẻ sơ sinh chơi đùa, vẫy tay hoặc nhìn xung quanh, nó không có giáo viên rõ ràng, nhưng nó đã tạo ra 1 kết nối cảm biến vận động trực tiếp với môi trường của mình. Việc thực hiện kết nối này tạo ra rất nhiều thông tin về nguyên nhân & kết quả, về hậu quả của hành động, & về những việc cần làm để đạt được mục tiêu. Trong suốt cuộc đời của chúng ta, những tương tác như vậy chắc chắn là 1 nguồn kiến thức chính về môi trường & bản thân chúng ta. Cho dù chúng ta đang học lái xe hay trò chuyện, chúng ta đều nhận thức sâu sắc về cách môi trường của chúng ta phản ứng với những gì chúng ta làm, & chúng ta tìm cách ảnh hưởng đến những gì xảy ra thông qua hành vi của mình. Học hỏi từ tương tác là 1 ý tưởng nền tảng làm nền tảng cho hầu hết mọi lý thuyết về học tập & trí thông minh.

  In this book, explore a *computational* approach to learning from interaction. Rather than directly theorizing about how people or animals learn, primarily explore idealized learning situations & evaluate effectiveness of various learning methods. I.e., adopt perspective of an AI researcher or engineer. Explore designs for machines that are effective in solving learning problems of scientific or economic interest, evaluating designs through mathematical analysis of computational experiments. Approach we explore, called *reinforcement learning*, is much more focused on goal-directed learning from interaction than other approaches to ML.

  – Trong cuốn sách này, chúng ta sẽ khám phá phương pháp tiếp cận *computation* để học từ tương tác. Thay vì trực tiếp lý thuyết hóa cách con người hoặc động vật học tập, chúng ta sẽ chủ yếu khám phá các tình huống học tập lý tưởng & đánh giá hiệu quả của các phương pháp học tập khác nhau. Ví dụ, hãy áp dụng góc nhìn của 1 nhà nghiên cứu hoặc kỹ sư AI. Khám phá các thiết kế máy móc hiệu quả trong việc giải quyết các vấn đề học tập mang tính khoa học hoặc kinh tế, đánh giá các thiết kế thông qua phân tích toán học các thí nghiệm tính toán. Phương pháp chúng tôi khám phá, được gọi là *reinforcement learning*, tập trung nhiều hơn vào việc học tập hướng mục tiêu từ tương tác so với các phương pháp tiếp cận ML khác.

  - 1.1. RL. RL is learning what to do – how to map situations to actions – so as to maximize a numerical reward signal. Learner is not told which actions to take, but instead must discover which actions yield most reward by trying them. In most interesting & challenging cases, actions may affect not only immediate reward but also next situation &, through that, all subsequent rewards. These 2 characteristics – trial-&-error search & delayed reward – are 2 most important distinguishing features of RL.

    – RL đang học cách làm gì – cách liên kết tình huống với hành động – để tối đa hóa tín hiệu phần thưởng số. Người học không được chỉ dẫn phải thực hiện hành động nào, mà thay vào đó, phải tự khám phá hành động nào mang lại phần thưởng cao nhất bằng cách thử nghiệm chúng. Trong hầu hết các trường hợp & thử thách thú vị, hành động có thể ảnh hưởng không chỉ đến phần thưởng tức thời mà còn cả tình huống tiếp theo &, qua đó, tất cả các phần thưởng tiếp theo. Hai đặc điểm này – tìm kiếm thử nghiệm & lỗi & phần thưởng trì hoãn – là 2 đặc điểm phân biệt quan trọng nhất của RL.

    RL, like many topics whose names end with "ing", e.g. ML & mountaineering, is simultaneously a problem, a class of solution methods that work well on problem, & field that studies this problem & its solution methods. Convenient to use a single name for all 3 things, but at same time essential to keep 3 conceptually separate. In particular, distinction between problems & solution methods is very important in RL; failing to make this distinction is source of many confusions.

    – RL, giống như nhiều chủ đề có tên kết thúc bằng "ing", ví dụ: ML & mountaineering, đồng thời là 1 bài toán, 1 lớp các phương pháp giải quyết hiệu quả cho bài toán, & lĩnh vực nghiên cứu bài toán này & các phương pháp giải quyết của nó. Việc sử dụng 1 tên gọi duy nhất cho cả 3 lĩnh vực rất tiện lợi, nhưng đồng thời cũng cần thiết phải tách biệt 3 khái niệm

này về mặt khái niệm. Đặc biệt, việc phân biệt giữa bài toán & phương pháp giải quyết là rất quan trọng trong RL; việc không phân biệt được điều này là nguyên nhân gây ra nhiều nhầm lẫn.

Formalize problem of RL using ideas from dynamical systems theory, specifically, as optimal control of incompletely-known Markov decision processes. Details of this formalization must wait until Chap. 3, but basic idea is simply to capture most important aspects of real problem facing a learning agent interacting over time with its environment to achieve a goal. A learning agent must be able to sense state of its environment to some extent & must be able to take actions that affect state. Agent also must have a goal or goals relating to state of environment. Markov decision processes are intended to include just these 3 aspects – sensation, action, & goal – in their simplest possible forms without trivializing any of them. Any method that is well suited to solving such problems we consider to be RL method.

– Chính thức hóa bài toán RL bằng các ý tưởng từ lý thuyết hệ thống động, cụ thể là điều khiển tối ưu các quá trình quyết định Markov chưa được biết đầy đủ. Chi tiết về chính thức hóa này phải chờ đến Chương 3, nhưng ý tưởng cơ bản chỉ đơn giản là nắm bắt hầu hết các khía cạnh quan trọng của vấn đề thực tế mà 1 tác nhân học tập tương tác theo thời gian với môi trường của nó để đạt được mục tiêu. Một tác nhân học tập phải có khả năng cảm nhận trạng thái của môi trường ở 1 mức độ nào đó & phải có khả năng thực hiện các hành động ảnh hưởng đến trạng thái. Tác nhân cũng phải có 1 hoặc nhiều mục tiêu liên quan đến trạng thái của môi trường. Các quá trình quyết định Markov được thiết kế để chỉ bao gồm 3 khía cạnh này – cảm giác, hành động, & mục tiêu – ở dạng đơn giản nhất có thể mà không tầm thường hóa bất kỳ khía cạnh nào. Bất kỳ phương pháp nào phù hợp để giải quyết các vấn đề như vậy, chúng tôi coi là phương pháp RL.

RL is different from *supervised learning*, kind of learning studied in most current research in field of ML. Supervised learning is learning from a training set of labeled examples provided by a knowledgeable external supervisor. Each example is a description of a situation together with a specification – label – of correct action system should take in that situation, which is often to identify a category to which situation belongs. Object of this kind of learning is for system to extrapolate, or generalize, its responses so that it acts correctly in situations not present in training set. This is an important kind of learning, but alone it is not adequate for learning from interaction. In interactive problems, often impractical to obtain examples of desired behavior that are both correct & representative of all situations in which agent has to act. In uncharted territory – where one would expect learning to be most beneficial – an agent must be able to learn from its own experience.

– RL khác với *học có giám sát*, loại học được nghiên cứu trong hầu hết các nghiên cứu hiện tại trong lĩnh vực ML. Học có giám sát là học từ 1 tập huấn luyện các ví dụ được gắn nhãn do 1 người giám sát bên ngoài có hiểu biết cung cấp. Mỗi ví dụ là 1 mô tả về 1 tình huống cùng với 1 thông số kỹ thuật – nhãn – về hành động đúng mà hệ thống nên thực hiện trong tình huống đó, thường là để xác định 1 danh mục mà tình huống đó thuộc về. Mục tiêu của loại học này là để hệ thống suy rộng hoặc khái quát hóa các phản ứng của nó để nó hành động chính xác trong các tình huống không có trong tập huấn luyện. Đây là 1 loại học quan trọng, nhưng chỉ riêng nó thì không đủ để học từ tương tác. Trong các vấn đề tương tác, thường không thực tế để có được các ví dụ về hành vi mong muốn vừa chính xác & đại diện cho tất cả các tình huống mà tác nhân phải hành động. Trong lãnh thổ chưa được khám phá – nơi mà người ta mong đợi việc học có lợi nhất – 1 tác nhân phải có khả năng học hỏi từ kinh nghiệm của chính mình.

RL is also different from what ML researchers call *unsupervised learning*, which is typically about finding structure hidden in collections of unlabeled data. Terms supervised learning & unsupervised learning would seem to exhaustively classify ML paradigms, but they do not. Although one might be tempted to think of RL as a kind of unsupervised learning because it does not rely on examples of correct behavior, RL is trying to maximize a reward signal instead of trying to find hidden structure. Uncovering structure in an agent's experience can certainly be useful in RL, but by itself does not address RL problem of maximizing a reward signal. Therefore consider RL to be a 3rd ML paradigm, alongside supervised learning & unsupervised learning & perhaps other paradigms.

– RL cũng khác với cái mà các nhà nghiên cứu ML gọi là *unsupervised learning*, thường là về việc tìm kiếm cấu trúc ẩn trong các tập hợp dữ liệu chưa được gắn nhãn. Các thuật ngữ học có giám sát & unsupervised learning dường như phân loại đầy đủ các mô hình ML, nhưng chúng không phải vậy. Mặc dù người ta có thể bị cám dỗ nghĩ về RL như 1 loại học không giám sát vì nó không dựa trên các ví dụ về hành vi đúng, RL đang cố gắng tối đa hóa tín hiệu phần thưởng thay vì cố gắng tìm kiếm cấu trúc ẩn. Việc khám phá cấu trúc trong trải nghiệm của 1 tác nhân chắc chắn có thể hữu ích trong RL, nhưng bản thân nó không giải quyết được vấn đề RL về việc tối đa hóa tín hiệu phần thưởng. Do đó, hãy coi RL là mô hình ML thứ 3, cùng với học có giám sát & unsupervised learning & có lẽ là các mô hình khác.

1 of challenges that arise in RL, & not in other kinds of learning, is trade-off between exploration & exploitation. To obtain a lot of reward, a RL agent must prefer actions that it has tried in past & found to be effective in producing reward. But to discover such actions, it has to try actions that it has not selected before. Agent has to *exploit* what it has already experienced in order to obtain reward, but it also has to *explore* in order to make better action selections in future. Dilemma: neither exploration nor exploitation can be pursued exclusively without failing at task. Agent must try a variety of action & progressively favor those that appear to be best. On a stochastic task, each action must be tried many times to gain a reliable estimate of its expected reward. Exploration–exploitation dilemma has been intensively studied by mathematicians for many decades, yet remains unresolved. For now, simply note: entire issue of balancing exploration & exploitation does not even arise in supervised & unsupervised learning, at least in purest forms of these paradigms.

– Một trong những thách thức nảy sinh trong RL, & không phải trong các loại hình học tập khác, là sự đánh đổi giữa khám phá & khai thác. Để nhận được nhiều phần thưởng, 1 tác nhân RL phải ưu tiên các hành động mà nó đã thử trong quá khứ & thấy hiệu quả trong việc tạo ra phần thưởng. Nhưng để khám phá ra những hành động như vậy, nó phải thử các hành động mà nó chưa từng chọn trước đây. Tác nhân phải *nó khai thác* những gì nó đã trải nghiệm để nhận được phần thưởng, nhưng nó cũng phải *nó khám phá* để đưa ra lựa chọn hành động tốt hơn trong tương lai. Thế tiến thoái lưỡng nan: cả khám phá lẫn khai thác đều không thể được theo đuổi độc quyền mà không thất bại trong nhiệm vụ. Tác nhân phải thử

nhiều hành động & dần dần ưu tiên những hành động có vẻ tốt nhất. Trong 1 nhiệm vụ ngẫu nhiên, mỗi hành động phải được thử nhiều lần để có được ước tính đáng tin cậy về phần thưởng dự kiến của nó. Thế tiến thoái lưỡng nan giữa khám phá & khai thác đã được các nhà toán học nghiên cứu sâu rộng trong nhiều thập kỷ, nhưng vẫn chưa được giải quyết. Hiện tại, chỉ cần lưu ý: toàn bộ vấn đề cân bằng giữa khám phá & khai thác thậm chí không phát sinh trong học tập có giám sát & không giám sát, ít nhất là ở dạng tinh khiết nhất của các mô hình này.

Another key feature of RL: it explicitly considers *whole* problem of a goal-directed agent interacting with an uncertain environment. This is in contrast to many approaches that consider subproblems without addressing how they might fit into a larger picture. E.g., have mentioned: many ML researchers have studied supervised learning without specifying how such an ability would ultimately be useful. Other researchers have developed theories of planning with general goals, but without considering planning's role in real-time decision making, or question of where predictive models necessary for planning would come from. Although these approaches have yielded many useful results, their focus on isolated subproblems is a significant limitation.

– Một đặc điểm quan trọng khác của RL: nó xem xét 1 cách rõ ràng *toàn bộ* vấn đề của 1 tác nhân hướng đến mục tiêu tương tác với 1 môi trường không chắc chắn. Điều này trái ngược với nhiều phương pháp tiếp cận chỉ xem xét các bài toán con mà không giải quyết cách chúng có thể phù hợp với bức tranh tổng thể. Ví dụ, đã đề cập: nhiều nhà nghiên cứu ML đã nghiên cứu học có giám sát mà không chỉ rõ khả năng đó cuối cùng sẽ hữu ích như thế nào. Các nhà nghiên cứu khác đã phát triển các lý thuyết về lập kế hoạch với các mục tiêu chung, nhưng lại không xem xét vai trò của lập kế hoạch trong việc ra quyết định theo thời gian thực, hoặc câu hỏi về việc các mô hình dự đoán cần thiết cho việc lập kế hoạch sẽ đến từ đâu. Mặc dù các phương pháp tiếp cận này đã mang lại nhiều kết quả hữu ích, nhưng việc tập trung vào các bài toán con riêng lẻ là 1 hạn chế đáng kể.

RL takes opposite tack, starting with a complete, interactive, goal-seeking agent. All RL agents have explicit goals, can sense aspects of their environments, & can choose actions to influence their environments. Moreover, usually assumed from beginning: agent has to operate despite significant uncertainty about environment it faces. When RL involves planning, it has to address interplay between planning & real-time action selection, as well as question of how environment models are acquired & improved. When RL involves supervised learning, it does so for specific reasons that determine which capabilities are critical & which are not. For learning research to make progress, important subproblems have to be isolated & studied, but they should be subproblems that play clear roles in complete, interactive, goal-seeking agents, even if all details of complete agent cannot yet be filled in.

– RL có hướng đi ngược lại, bắt đầu với 1 tác nhân hoàn chỉnh, tương tác & tìm kiếm mục tiêu. Tất cả các tác nhân RL đều có mục tiêu rõ ràng, có thể cảm nhận các khía cạnh của môi trường của chúng, & có thể chọn hành động để tác động đến môi trường của chúng. Hơn nữa, thường được giả định ngay từ đầu: tác nhân phải hoạt động bất chấp sự không chắc chắn đáng kể về môi trường mà nó phải đối mặt. Khi RL liên quan đến việc lập kế hoạch, nó phải giải quyết sự tương tác giữa việc lập kế hoạch & lựa chọn hành động theo thời gian thực, cũng như câu hỏi về cách các mô hình môi trường được tiếp thu & cải thiện. Khi RL liên quan đến việc học có giám sát, nó làm như vậy vì những lý do cụ thể để xác định khả năng nào là quan trọng & khả năng nào không. Để nghiên cứu về học tập đạt được tiến bộ, các vấn đề phụ quan trọng phải được phân lập & nghiên cứu, nhưng chúng phải là những vấn đề phụ đóng vai trò rõ ràng trong các tác nhân hoàn chỉnh, tương tác & tìm kiếm mục tiêu, ngay cả khi tất cả các chi tiết của tác nhân hoàn chỉnh vẫn chưa thể được điền đầy đủ.

By a complete, interactive, goal-seeking agent we do not always mean something like a complete organism or robot. These are clearly examples, but a complete, interactive, goal-seeking agent can also be a component of a larger behaving system. In this case, agent directly interacts with rest of larger system & indirectly interacts with larger system's environment. A simple example is an agent that monitors charge level of robot's battery & sends commands to robot's control architecture. This agent's environment is rest of robot together with robot's environment. Important to look beyond most obvious examples of agents & their environments to appreciate generality of RL framework.

– Khi nói đến 1 tác nhân hoàn chỉnh, tương tác & tìm kiếm mục tiêu, chúng ta không phải lúc nào cũng muốn nói đến 1 sinh vật hay robot hoàn chỉnh. Đây rõ ràng là những ví dụ, nhưng 1 tác nhân hoàn chỉnh, tương tác & tìm kiếm mục tiêu cũng có thể là 1 thành phần của 1 hệ thống hoạt động lớn hơn. Trong trường hợp này, tác nhân tương tác trực tiếp với phần còn lại của hệ thống lớn hơn & gián tiếp tương tác với môi trường của hệ thống lớn hơn. Một ví dụ đơn giản là 1 tác nhân theo dõi mức sạc pin của robot & gửi lệnh đến kiến trúc điều khiển của robot. Môi trường của tác nhân này là phần còn lại của robot cùng với môi trường của robot. Điều quan trọng là phải xem xét kỹ hơn hầu hết các ví dụ rõ ràng về tác nhân & môi trường của chúng để đánh giá được tính tổng quát của khuôn khổ RL.

1 of most exciting aspects of modern RL is its substantive & fruitful interactions with other engineering & scientific disciplines. RL is part of a decades-long trend within AI & ML toward greater integration with statistics, optimization, & other mathematical subjects. E.g., ability of some RL methods to learn with parameterized approximators addresses classical "curse of dimensionality" in operations research & control theory. More distinctively, RL has also interacted strongly with psychology & neuroscience, with substantial benefits going both ways. Of all forms of ML, RL is closest to kind of learning that humans & other animals do, & many of core algorithms of RL were originally inspired by biological learning systems. RL has also given back, both through a psychological model of animal learning that better matches some of empirical data, & through an influential model of parts of brain's reward system. Body of this book develops ideas of RL that pertain to engineering & AI, with connections to psychology & neuroscience summarized in Chaps. 14–15.

– Một trong những khía cạnh thú vị nhất của Học máy (RL) hiện đại là những tương tác thiết thực & hiệu quả của nó với các ngành kỹ thuật & khoa học khác. RL là 1 phần của xu hướng kéo dài hàng thập kỷ trong AI & ML hướng tới sự tích hợp sâu hơn với thống kê, tối ưu hóa & các môn toán học khác. Ví dụ, khả năng học với các bộ xấp xỉ tham số của 1 số phương pháp RL giải quyết "lời nguyền đa chiều" cổ điển trong nghiên cứu vận hành & lý thuyết điều khiển. Đặc biệt hơn,

RL cũng tương tác mạnh mẽ với tâm lý học & khoa học thần kinh, mang lại những lợi ích đáng kể cho cả hai bên. Trong tất cả các dạng ML, RL gần nhất với loại hình học tập mà con người & các loài động vật khác thực hiện, & nhiều thuật toán cốt lõi của RL ban đầu được lấy cảm hứng từ các hệ thống học tập sinh học. RL cũng đã đóng góp trở lại, thông qua 1 mô hình tâm lý học về học tập của động vật phù hợp hơn với 1 số dữ liệu thực nghiệm, & thông qua 1 mô hình có ảnh hưởng về các bộ phận của hệ thống khen thưởng của não. Nội dung cuốn sách này phát triển các ý tưởng RL liên quan đến kỹ thuật & AI, với mối liên hệ với tâm lý học & khoa học thần kinh được tóm tắt trong các Chương 14–15.

Finally, RL is also part of a larger trend in AI back toward simple general principles. Since late 1960s, many AI researchers presumed: there are no general principles to be discovered, that intelligence is instead due to possession of a vast number of special purpose tricks, procedures, & heuristics. It was sometimes said: if we could just get enough relevant facts into a machine, say 1 million, or 1 billion, then it would become intelligent. Methods based on general principles, e.g. search or learning, were characterized as "weak methods", whereas those based on specific knowledge were called "strong methods". This view is uncommon today. From our point of view, it was premature: too little effort had been put into search for general principles to conclude that there were none. Modern AI now includes much research looking for general principles of learning, search, & decision making. Not clear how far back pendulum will swing, but RL research is certainly part of swing back toward simpler & fewer general principles of AI.

– Cuối cùng, RL cũng là 1 phần của xu hướng lớn hơn trong AI hướng về các nguyên lý chung đơn giản. Từ cuối những năm 1960, nhiều nhà nghiên cứu AI đã cho rằng: không có nguyên lý chung nào cần được khám phá, mà trí thông minh thực chất là do sở hữu vô số các thủ thuật, quy trình, & phương pháp tìm kiếm chuyên biệt. Đôi khi người ta nói: nếu chúng ta có thể đưa đủ dữ liệu liên quan vào 1 cỗ máy, chẳng hạn như 1 triệu, hoặc 1 tỷ, thì nó sẽ trở nên thông minh. Các phương pháp dựa trên các nguyên lý chung, ví dụ như tìm kiếm hoặc học tập, được gọi là "phương pháp yếu", trong khi các phương pháp dựa trên kiến thức cụ thể được gọi là "phương pháp mạnh". Quan điểm này không phổ biến ngày nay. Theo quan điểm của chúng tôi, điều đó là quá sớm: quá ít nỗ lực được bỏ ra để tìm kiếm các nguyên lý chung để kết luận rằng không có nguyên lý nào cả. AI hiện đại ngày nay bao gồm nhiều nghiên cứu tìm kiếm các nguyên lý chung về học tập, tìm kiếm, & ra quyết định. Không rõ con lắc sẽ quay ngược lại bao xa, nhưng nghiên cứu RL chắc chắn là 1 phần của quá trình quay trở lại với các nguyên lý chung đơn giản hơn & ít nguyên lý chung hơn của AI.

- 1.2. Examples. A good way to understand RL: consider some of examples & possible applications that have guided its development.
  - A master chess player makes a move. Choice is informed both by planning – anticipating possible replies & counterreplies – & by immediate, intuitive judgments of desirability of particular positions & moves.
  - An adaptive controller adjusts parameters of a petroleum refinery's operation in real time. Controller optimizes yield/cost/quality trade-off on basis of specified marginal costs without sticking strictly to set points originally suggested by engineers.
  - A gazelle calf struggles to its feet minutes after being born. Half an hour later it is running at 20 miles per hour.
  - A mobile robot decides whether it should enter a new room in search of more trash to collect or start trying to find its way back to its battery recharging station. It makes its decision based on current charge level of its battery & how quickly & easily it has been able to find recharger in past.
  - Phil prepares his breakfast. Closely examined, even this apparently mundane activity reveals a complex web of conditional behavior & interlocking goal-subgoal relationships: walking to cupboard, opening it, selecting a cereal box, then reaching for, grasping, & retrieving box. Other complex, tuned, interactive sequences of behavior are required to obtain a bowl, spoon, & milk carton. Each step involves a series of eye movements to obtain information & to guide reaching & locomotion. Rapid judgments are continually made about how to carry objects or whether it is better to ferry some of them to dining table before obtaining others. Each step is guided by goals, e.g. grasping a spoon or getting to refrigerator, & is in service of other goals, e.g. having spoon to eat with once cereal is prepared & ultimately obtaining nourishment. Whether he is aware of it or not, Phil is accessing information about state of his body that determines his nutritional needs, level of hunger, & food preferences.

– Một cách hay để hiểu RL: hãy xem xét 1 số ví dụ & các ứng dụng khả thi đã định hướng cho sự phát triển của nó.

- Một kỳ thủ cờ vua lão luyện thực hiện 1 nước đi. Lựa chọn được đưa ra dựa trên cả việc lập kế hoạch – dự đoán các câu trả lời khả thi & các câu trả lời ngược lại – & bằng những phán đoán trực quan, tức thời về mức độ mong muốn của các vị trí & nước đi cụ thể.
- Một bộ điều khiển thích ứng điều chỉnh các thông số vận hành của 1 nhà máy lọc dầu theo thời gian thực. Bộ điều khiển tối ưu hóa sự cân bằng giữa năng suất/chi phí/chất lượng dựa trên chi phí cận biên cụ thể mà không tuân thủ nghiêm ngặt các điểm đặt ban đầu do các kỹ sư đề xuất.
- Một con linh dương con đang cố gắng đứng dậy sau khi sinh ra vài phút. Nửa giờ sau, nó chạy với tốc độ 20 dặm 1 giờ.
- Một robot di động quyết định liệu nó nên đi vào 1 căn phòng mới để tìm thêm rác để thu gom hay bắt đầu tìm đường quay trở lại trạm sạc pin. Nó đưa ra quyết định dựa trên mức sạc pin hiện tại & tốc độ & dễ dàng mà nó đã tìm thấy bộ sạc trong quá khứ.
- Phil chuẩn bị bữa sáng. Khi xem xét kỹ lưỡng, ngay cả hoạt động tưởng chừng như tầm thường này cũng hé lộ 1 mạng lưới phức tạp của các hành vi có điều kiện & các mối quan hệ mục tiêu-mục tiêu phụ đan xen: đi đến tủ bếp, mở tủ, chọn hộp ngũ cốc, rồi với lấy, nắm lấy, & lấy hộp. Các chuỗi hành vi phức tạp, được điều chỉnh & tương tác khác cũng cần thiết để lấy được bát, thìa, & hộp sữa. Mỗi bước bao gồm 1 loạt chuyển động mắt để thu thập thông tin & hướng dẫn việc với & di chuyển. Những phán đoán nhanh chóng liên tục được đưa ra về cách mang đồ vật hoặc liệu có nên mang 1 số đồ vật đến bàn ăn trước khi lấy những đồ vật khác hay không. Mỗi bước đều được hướng dẫn bởi các mục tiêu, ví dụ:

cầm thìa hay đến tủ lạnh, & phục vụ cho các mục tiêu khác, ví dụ: có thìa để ăn sau khi ngũ cốc đã được chuẩn bị & cuối cùng là có được dinh dưỡng. Dù có nhận thức được điều đó hay không, Phil đang tiếp cận thông tin về trạng thái cơ thể của mình, từ đó quyết định nhu cầu dinh dưỡng, mức độ đói, & sở thích ăn uống của cậu bé.

These examples share features that are so basic that they are easy to overlook. All involve *interaction* between an active decision-making agent & its environment, within which agent seeks to achieve a *goal* despite *uncertainty* about its environment. Agent's actions are permitted to affect future state of environment (e.g., next chess position, level of reservoirs of refinery, robot's next location & future charge level of its battery), thereby affecting actions & opportunities available to agent at later times. Correct choice requires taking into account indirect, delayed consequences of actions, & thus may require foresight or planning.

– These examples share features that are so basic that they are easy to overlook. All involve *interaction* between an active decision-making agent & its environment, within which agent seeks to achieve a *goal* despite *uncertainty* about its environment. Agent's actions are permitted to affect future state of environment (e.g., next chess position, level of reservoirs of refinery, robot's next location & future charge level of its battery), thereby affecting actions & opportunities available to agent at later times. Correct choice requires taking into account indirect, delayed consequences of actions, & thus may require foresight or planning.

At same time, in all of these examples effects of actions cannot be fully predicted; thus agent must monitor its environment frequently & react appropriately. E.g., Phil must watch milk he pours into his cereal bowl to keep it from overflowing. All these examples involve goals that are explicit in sense: agent can judge progress toward its goal based on what it can sense directly. Chess player knows whether or not he wins, refinery controller knows how much petroleum is being produced, gazelle calf knows when it falls, mobile robot knows when its batteries run down, & Phil knows whether or not he is enjoying his breakfast.

– Đồng thời, trong tất cả các ví dụ này, tác động của hành động không thể được dự đoán đầy đủ; do đó, tác nhân phải thường xuyên theo dõi môi trường xung quanh & phản ứng phù hợp. Ví dụ, Phil phải để ý sữa anh ta đổ vào bát ngũ cốc của mình để tránh bị tràn. Tất cả các ví dụ này đều liên quan đến các mục tiêu rõ ràng: tác nhân có thể đánh giá tiến độ đạt được mục tiêu dựa trên những gì nó có thể cảm nhận trực tiếp. Người chơi cờ vua biết mình thắng hay thua, người quản lý nhà máy lọc dầu biết lượng dầu đang được sản xuất, con linh dương con biết khi nào nó ngã, robot di động biết khi nào pin của nó hết, & Phil biết mình có đang thưởng thức bữa sáng hay không.

In all of these examples agent can use its experience to improve its performance over time. Chess player refines intuition he uses to evaluate positions, thereby improving his play; gazelle calf improves efficiency with which it can run; Phil learns to streamline making his breakfast. Knowledge agent brings to task at start – either from previous experience with related tasks or built into it by design or evolution – influences what is useful or easy to learn, but interaction with environment is essential for adjusting behavior to exploit specific features of task.

– Trong tất cả các ví dụ này, tác nhân có thể sử dụng kinh nghiệm của mình để cải thiện hiệu suất theo thời gian. Người chơi cờ vua tinh chỉnh trực giác mà anh ta sử dụng để đánh giá vị trí, từ đó cải thiện lối chơi của mình; con linh dương Gazelle cải thiện hiệu suất chạy; Phil học cách đơn giản hóa việc chuẩn bị bữa sáng. Kiến thức mà tác nhân mang đến cho nhiệm vụ ngay từ đầu – từ kinh nghiệm trước đó với các nhiệm vụ liên quan hoặc được tích hợp sẵn trong đó theo thiết kế hoặc quá trình tiến hóa – sẽ quyết định những gì hữu ích hoặc dễ học, nhưng tương tác với môi trường là điều cần thiết để điều chỉnh hành vi nhằm khai thác các đặc điểm cụ thể của nhiệm vụ.

- 1.3. Elements of RL. Beyond agent & environment, one can identify 4 main subelements of a RL system: a *policy*, a *reward signal*, a *value function*, &, optionally, a *model* of environment.

  – Ngoài tác nhân & môi trường, người ta có thể xác định 4 yếu tố phụ chính của hệ thống RL: *chính sách*, *tín hiệu phần thưởng*, *hàm giá trị*, &, tùy chọn, *mô hình* của môi trường.

  A *policy* defines learning agent's way of behaving at a given time. Roughly speaking, a policy is a mapping from perceived states of environment to actions to be taken when in those states. It corresponds to what in psychology would be called a set of stimulus–response rules or associations. In some cases policy may be a simple function or lookup table, whereas in others it may involve extensive computation e.g. a search process. Policy is core of a RL agent in sense: it alone is sufficient to determine behavior. In general, policies may be stochastic, specifying probabilities for each action.

  – Chính sách học tập (*policy) xác định cách thức hành xử của tác nhân học tập tại 1 thời điểm nhất định. Nói 1 cách đại khái, chính sách là 1 phép ánh xạ từ các trạng thái nhận thức của môi trường đến các hành động cần thực hiện khi ở trong các trạng thái đó. Nó tương ứng với cái mà trong tâm lý học được gọi là 1 tập hợp các quy tắc hoặc mối liên hệ giữa kích thích & phản ứng. Trong 1 số trường hợp, chính sách có thể là 1 hàm đơn giản hoặc bảng tra cứu, trong khi ở những trường hợp khác, nó có thể liên quan đến các phép tính phức tạp, ví dụ như 1 quy trình tìm kiếm. Chính sách là cốt lõi của 1 tác nhân học tập theo nghĩa: chỉ riêng nó là đủ để xác định hành vi. Nhìn chung, các chính sách có thể là ngẫu nhiên, chỉ định xác suất cho mỗi hành động.*

- 1.4. Limitations & Scope.

PART I: TABULAR SOLUTION METHODS.

- 2. Multi-armed Bandits.
- 3. Finite Markov Decision Processes.
- 4. Dynamic Programming.

- 5. Monte Carlo Methods.
- 6. Temporal-Difference Learning.
- 7. n-step Boostrapping.
- 8. Planning & Learning with Tabular Methods.

  PART II: APPROXIMATE SOLUTION METHODS.
- 9. On-policy Prediction with Approximation.
- 10. On-policy Control with Approximation.
- 11. Off-policy Methods with Approximation.
- 12. Eligibility Traces.
- 13. Policy Gradient Methods.

  PART III: LOOKING DEEPER.
- 14. Psychology.
- 15. Neuroscience.
- 16. Applications & Case Studies.
- 17. Frontiers.

2 Reinforcement Learning for Optimal Job Scheduling

2.1 Xinquan Wu, Xuefeng Yan, Mingqiang Wei, Donghai Guan. An Efficient Deep Reinforcement Learning Environment for Flexible Job-Shop Scheduling. Sep 10, 2025

- Abstract. Flexible Job-shop Scheduling Problem (FJSP) is a classical combinatorial optimization problem that has a wide-range of applications in real world. In order to generate fast & accurate scheduling solutions for FJSP, various deep reinforcement learning (DRL) scheduling methods have been developed. However, these methods are mainly focused on design of DRL scheduling Agent, overlooking modeling of DRL environment. This paper presents a simple chronological DRL environment for FJSP based on discrete event simulation & an end-to-end DRL scheduling model is proposed based on proximal policy optimization (PPO). Furthermore, a short novel state representation of FJSP is proposed based on 2 state variables in scheduling environment & a novel comprehensive reward function is designed based on scheduling area of machines. Experimental results on public benchmark instances show: performance of simple priority dispatching rules (PDR) is improved in our scheduling environment & our DRL scheduling model obtains competing performance compared with OR-Tools, meta-heuristic, DRL & PDR scheduling methods.

  – Bài toán Lập lịch Xưởng Linh hoạt (FJSP) là 1 bài toán tối ưu hóa tổ hợp cổ điển có phạm vi ứng dụng rộng rãi trong thực tế. Để tạo ra các giải pháp lập lịch nhanh & chính xác cho FJSP, nhiều phương pháp lập lịch học tăng cường sâu (DRL) đã được phát triển. Tuy nhiên, các phương pháp này chủ yếu tập trung vào thiết kế Tác nhân lập lịch DRL, bỏ qua việc mô hình hóa môi trường DRL. Bài báo này trình bày 1 môi trường DRL theo trình tự thời gian đơn giản cho FJSP dựa trên mô phỏng sự kiện rời rạc & 1 mô hình lập lịch DRL đầu cuối được đề xuất dựa trên tối ưu hóa chính sách gần đúng (PPO). Hơn nữa, 1 biểu diễn trạng thái mới ngắn gọn của FJSP được đề xuất dựa trên 2 biến trạng thái trong môi trường lập lịch & 1 hàm thưởng toàn diện mới được thiết kế dựa trên vùng lập lịch của máy. Kết quả thử nghiệm trên các trường hợp chuẩn công khai cho thấy: hiệu suất của các quy tắc phân bổ ưu tiên đơn giản (PDR) được cải thiện trong môi trường lập lịch của chúng tôi & mô hình lập lịch DRL của chúng tôi đạt được hiệu suất cạnh tranh so với các phương pháp lập lịch OR-Tools, meta-heuristic, DRL & PDR.
- 1. Introduction. Job shop scheduling problem (JSSP) is a classical NP-hard combinatorial optimization problem. It has been studied for decades & has been applied in a wide-range of areas including semiconductor, machine manufacturing, metallurgy, automobile manufacturing, supply chain & other fields [25]. Flexible Job-shop Scheduling Problem (FJSP) is a generalization of JSSP, which allows each operation to be processed on multiple candidate machines [6]. This makes it a more challenging problem with more complex topology & larger solution space.

  – Bài toán lập lịch xưởng gia công (JSSP) là 1 bài toán tối ưu tổ hợp NP-khó kinh điển. Nó đã được nghiên cứu trong nhiều thập kỷ & đã được ứng dụng trong nhiều lĩnh vực bao gồm bán dẫn, chế tạo máy, luyện kim, sản xuất ô tô, chuỗi cung ứng & các lĩnh vực khác [25]. Bài toán lập lịch xưởng gia công linh hoạt (FJSP) là 1 dạng tổng quát của JSSP, cho phép mỗi thao tác được xử lý trên nhiều máy ứng viên [6]. Điều này làm cho nó trở thành 1 bài toán khó hơn với cấu trúc phức tạp hơn & không gian nghiệm lớn hơn.

  Extensive researches have been widely explored to solve FJSP in recent years, including exact, heuristic, meta-heuristic, hyper-heuristic & RL methods. Exact approaches e.g. mixed integer linear programming (MILP) [20] may suffer from curse of

dimension. So it is intractable to find exact scheduling solutions in a given time limit. Heuristic dispatching rules are widely used in real world manufacturing due to their simple & fast nature. Simple priority dispatching rules (PDR)[21], e.g. most work remaining (MWKR) possess advantages of high flexibility & easy implementation, but performance of PDR methods is not stable for different optimization objectives. Hyper-heuristic methods [29] e.g. genetic programming-based hyper-heuristic (GPHH), provide a way of automatically designing dispatching rules [28]. However, GP-evolved rules often has a large number of features in terminal set, making it difficult to identify promising search areas & determine optimal features.

– Các nghiên cứu sâu rộng đã được khai thác rộng rãi để giải quyết FJSP trong những năm gần đây, bao gồm các phương pháp chính xác, heuristic, meta-heuristic, hyper-heuristic & RL. Các phương pháp chính xác, ví dụ như lập trình tuyến tính số nguyên hỗn hợp (MILP) [20] có thể bị lời nguyền về chiều. Do đó, việc tìm ra các giải pháp lập lịch chính xác trong 1 giới hạn thời gian nhất định là khó khăn. Các quy tắc điều phối heuristic được sử dụng rộng rãi trong sản xuất thực tế do bản chất đơn giản & nhanh chóng của chúng. Các quy tắc điều phối ưu tiên đơn giản (PDR) [21], ví dụ như hầu hết công việc còn lại (MWKR) có ưu điểm là tính linh hoạt cao & dễ triển khai, nhưng hiệu suất của các phương pháp PDR không ổn định đối với các mục tiêu tối ưu khác nhau. Các phương pháp hyper-heuristic [29], ví dụ như hyper-heuristic dựa trên lập trình di truyền (GPHH), cung cấp 1 cách để tự động thiết kế các quy tắc điều phối [28]. Tuy nhiên, các quy tắc phát triển GP thường có 1 số lượng lớn các tính năng trong tập đầu cuối, khiến việc xác định các khu vực tìm kiếm đầy hứa hẹn & xác định các tính năng tối ưu trở nên khó khăn.

Different from PDR, meta-heuristic methods e.g. Tabu Search [12], Genetic Algorithm [19], Differential Evolution [7] & Particle Swarm Optimization [1], can produce higher solution quality than PDR due to introduction of iterative, randomized search strategies. Nevertheless, these methods also have shortcomings e.g. slow convergence speed, difficulty in obtaining global optimal solutions for large-scale problems & are difficult to apply to dynamic scenarios that need real-time decisions.

– Khác với PDR, các phương pháp siêu heuristic như Tìm kiếm Tabu [12], Thuật toán Di truyền [19], Tiến hóa Vi phân [7] & Tối ưu hóa Bầy đàn Hạt [1] có thể tạo ra chất lượng giải pháp cao hơn PDR nhờ áp dụng các chiến lược tìm kiếm lặp lại, ngẫu nhiên. Tuy nhiên, các phương pháp này cũng có những hạn chế, ví dụ như tốc độ hội tụ chậm, khó đạt được giải pháp tối ưu toàn cục cho các bài toán quy mô lớn & khó áp dụng cho các tình huống động cần quyết định theo thời gian thực.

RL is 1 of most important branches of ML & attracts interests of researchers from numerous fields especially for scheduling. Early RL scheduling methods represent scheduling policies using arrays or tables [2], which are only suitable for small-scale scheduling problems because of curse of dimension. However, scheduling tasks in real world usually have a higher dimensional state space & large action space, which limits applications of RL. Since deep neural networks have a strong fitting capability to process high-dimensional data, deep reinforcement learning (DRL) has been used to solve scheduling tasks with large or continuous state space & shows great potential to solve various scheduling problems. In DRL scheduling methods, scheduling policy is usually designed based on convolutional neural network (CNN) in [9], multi-layer perception (MLP) in [9, 10], RNN in [14], GNN in [15, 22, 26], attention networks in [23] & other deep neural networks in [16]. Scheduling agent is often trained by RL algorithms e.g. deep Q-learning (DQN), proximal policy optimization (PPO), Deep Deterministic Policy Gradient (DDPG). However, above DRL scheduling methods obtained low accurate solutions & some are even inferior to simple PDR. Even though recent GMAS model [13] whose policy is designed based on Graph Convolutional Network (GCN), obtained SOTA results on standard benchmark instances, it is a decentralized Multi-agent rRL (MARL) model. Besides, current DRL scheduling methods focus mainly on design of scheduling agents, overlooking modeling of scheduling environment which is of vital important for improving performance of DRL scheduling methods.

– RL là 1 trong những nhánh quan trọng nhất của ML & thu hút sự quan tâm của các nhà nghiên cứu từ nhiều lĩnh vực, đặc biệt là đối với việc lập lịch. Các phương pháp lập lịch RL ban đầu biểu diễn các chính sách lập lịch sử dụng mảng hoặc bảng [2], chỉ phù hợp với các vấn đề lập lịch quy mô nhỏ do giới hạn chiều. Tuy nhiên, các tác vụ lập lịch trong thế giới thực thường có không gian trạng thái nhiều chiều & không gian hành động lớn, điều này hạn chế các ứng dụng của RL. Vì mạng nơ-ron sâu có khả năng khớp mạnh để xử lý dữ liệu nhiều chiều, nên học tăng cường sâu (DRL) đã được sử dụng để giải quyết các tác vụ lập lịch với không gian trạng thái lớn hoặc liên tục & cho thấy tiềm năng to lớn trong việc giải quyết nhiều vấn đề lập lịch khác nhau. Trong các phương pháp lập lịch DRL, chính sách lập lịch thường được thiết kế dựa trên mạng nơ-ron tích chập (CNN) trong [9], nhận thức đa lớp (MLP) trong [9, 10], RNN trong [14], GNN trong [15, 22, 26], mạng chú ý trong [23] & các mạng nơ-ron sâu khác trong [16]. Tác nhân lập lịch thường được đào tạo bằng các thuật toán RL, ví dụ: Học sâu Q (DQN), tối ưu hóa chính sách gần đúng (PPO), Gradient Chính sách Xác định Sâu (DDPG). Tuy nhiên, các phương pháp lập lịch DRL trên cho kết quả độ chính xác thấp & 1 số thậm chí còn kém hơn PDR đơn giản. Mặc dù mô hình GMAS gần đây [13], với chính sách được thiết kế dựa trên Mạng Tích chập Đồ thị (GCN), đã đạt được kết quả SOTA trên các trường hợp chuẩn mực, nhưng nó là 1 mô hình rRL Đa tác tử phi tập trung (MARL). Bên cạnh đó, các phương pháp lập lịch DRL hiện tại chủ yếu tập trung vào thiết kế các tác nhân lập lịch, bỏ qua việc mô hình hóa môi trường lập lịch, vốn rất quan trọng để cải thiện hiệu suất của các phương pháp lập lịch DRL.

There are mainly 4 modeling methods for FJSP: mathematical [20], Petri Nets (PN)-based [17], Disjunctive Graph (DG)-based [3] & simulation-based modeling [24]. Mathematical modeling methods usually formulates FJSP as a MILP & formulated problem is then be solved by optimization methods e.g. Genetic Algorithm. PN is a versatile modeling tool that can be used to model scheduling problems as they can model parallel activities, resource sharing & synchronization. Both modeling methods are not suitable for DRL scheduling methods. DG is another alternative to model FJSP which integrates machine status & operation dependencies, & provides critical structural information for scheduling decisions. However, DG fails to incorporate dynamically changing state information of FJSP, requiring extra handcrafted node features. Besides, allocation order provided by DG model is not strictly chronological. Simulation-based modeling method: develop algorithms to simulate laws of activities

in real world. In current simulation environment of DRL scheduling methods, scheduling process is promoted by events in candidate queue, ignoring occurrence temporal order of these events.

– Có chủ yếu 4 phương pháp mô hình hóa cho FJSP: toán học [20], dựa trên Petri Nets (PN) [17], dựa trên Disjunctive Graph (DG) [3] & mô hình hóa dựa trên mô phỏng [24]. Các phương pháp mô hình hóa toán học thường xây dựng FJSP như 1 bài toán MILP & sau đó được giải quyết bằng các phương pháp tối ưu hóa, ví dụ như thuật toán di truyền. PN là 1 công cụ mô hình hóa đa năng có thể được sử dụng để mô hình hóa các vấn đề lập lịch vì chúng có thể mô hình hóa các hoạt động song song, chia sẻ tài nguyên & đồng bộ hóa. Cả hai phương pháp mô hình hóa đều không phù hợp với các phương pháp lập lịch DRL. DG là 1 giải pháp thay thế khác cho mô hình FJSP tích hợp trạng thái máy & phụ thuộc hoạt động, & cung cấp thông tin cấu trúc quan trọng cho các quyết định lập lịch. Tuy nhiên, DG không kết hợp thông tin trạng thái thay đổi động của FJSP, yêu cầu các tính năng nút thủ công bổ sung. Bên cạnh đó, thứ tự phân bổ do mô hình DG cung cấp không hoàn toàn theo thứ tự thời gian. Phương pháp mô hình hóa dựa trên mô phỏng: phát triển các thuật toán để mô phỏng các quy luật hoạt động trong thế giới thực. Trong môi trường mô phỏng hiện tại của các phương pháp lập lịch DRL, quá trình lập lịch được thúc đẩy bởi các sự kiện trong hàng đợi ứng viên, bỏ qua thứ tự thời gian xảy ra của các sự kiện này.

In this paper, a chronological DRL scheduling environment for FJSP is presented based on discrete event simulation algorithm where at each decision step, accurate state changes of scheduling process are recorded by state variables & a novel comprehensive reward function is designed based on scheduling area. Furthermore, an end to end DRL model for FJSP is proposed based on actor-critic PPO. Specially, a very short state representation is expressed by 2 state variables which are derived directly from necessary variables in environment simulation algorithm & action space is constructed using 6 PDRs from literature. Besides, in order to reduce computation time for RL agent, a simple scheduling policy is designed based on MLP with only 1 hidden layer & Softmax function.

– Bài báo này trình bày 1 môi trường lập lịch DRL theo thời gian cho FJSP dựa trên thuật toán mô phỏng sự kiện rời rạc, trong đó tại mỗi bước quyết định, các thay đổi trạng thái chính xác của quy trình lập lịch được ghi lại bởi các biến trạng thái & 1 hàm thưởng toàn diện mới được thiết kế dựa trên vùng lập lịch. Hơn nữa, 1 mô hình DRL đầu cuối cho FJSP được đề xuất dựa trên PPO tác nhân-phê bình. Cụ thể, 1 biểu diễn trạng thái rất ngắn được thể hiện bằng 2 biến trạng thái được suy ra trực tiếp từ các biến cần thiết trong thuật toán mô phỏng môi trường & không gian hành động được xây dựng bằng cách sử dụng 6 PDR từ tài liệu. Bên cạnh đó, để giảm thời gian tính toán cho tác nhân RL, 1 chính sách lập lịch đơn giản được thiết kế dựa trên MLP với chỉ 1 lớp ẩn & hàm Softmax.

Contributions of this work are listed as follows.

1. A basic simulation algorithm based on chronological discrete event is designed for FJSP, providing a general environment for DRL scheduling methods.
2. A simple DRL model for FJSP is proposed based on PPO where a very short state representation for FJSP is presented, avoiding handcrafted state features & massive experiments for feature selection & a novel comprehensive reward function is proposed based on scheduling area.
3. Experimental results show: performance of single PDR is improved in our environment, even better than some DRL scheduling methods & proposed DRL scheduling model obtains competing performance compared with OR-Tools, meta-heuristic, DRL & PDR scheduling methods.

– Những đóng góp của công trình này được liệt kê như sau.

1. 1 thuật toán mô phỏng cơ bản dựa trên sự kiện rời rạc theo trình tự thời gian được thiết kế cho FJSP, cung cấp 1 môi trường chung cho các phương pháp lập lịch DRL.
2. 1 mô hình DRL đơn giản cho FJSP được đề xuất dựa trên PPO, trong đó 1 biểu diễn trạng thái rất ngắn cho FJSP được trình bày, tránh các đặc trưng trạng thái thủ công & các thử nghiệm hàng loạt để lựa chọn đặc trưng & 1 hàm thưởng toàn diện mới được đề xuất dựa trên khu vực lập lịch.
3. Kết quả thử nghiệm cho thấy: hiệu suất của PDR đơn lẻ được cải thiện trong môi trường của chúng tôi, thậm chí còn tốt hơn 1 số phương pháp lập lịch DRL & mô hình lập lịch DRL được đề xuất đạt được hiệu suất cạnh tranh so với các phương pháp lập lịch OR-Tools, meta-heuristic, DRL & PDR.

Rest of paper is organized as follows. A chronological discrete event simulation-based scheduling environment for FJSP is introduced in Sect. II. In Sect. III, proposed DRL scheduling model for FJSP are constructed. Sect. IV demonstrates detailed experiments on standard benchmarks with different sizes & results are compared & discussed, while conclusions & future work reside in Sect. V.

– Phần còn lại của bài báo được tổ chức như sau. 1 môi trường lập lịch dựa trên mô phỏng sự kiện rời rạc theo thời gian cho FJSP được giới thiệu trong Phần II. Trong Phần III, mô hình lập lịch DRL đề xuất cho FJSP đã được xây dựng. Phần IV trình bày các thí nghiệm chi tiết trên các chuẩn mực chuẩn với các kích thước khác nhau & kết quả được so sánh & thảo luận, trong khi kết luận & các nghiên cứu trong tương lai nằm trong Phần V.

- 2. Simulation environment for FJSP. In this sect, 1st introduced basics of FJSPs. Then defined storage structure of FJSP & rules of state changing when scheduling. Finally, detailed simulation algorithm for FJSP is presented.

  – Môi trường mô phỏng cho FJSP. Trong phần này, trước tiên chúng tôi giới thiệu những kiến thức cơ bản về FJSP. Sau đó, chúng tôi định nghĩa cấu trúc lưu trữ của FJSP & các quy tắc thay đổi trạng thái khi lập lịch. Cuối cùng, thuật toán mô phỏng chi tiết cho FJSP được trình bày.

- 2.1. Flexible Job-shop Scheduling Problems. FJSP differs from JSSP in that it allows an operation to be processed by any machine from a given set & different machines may have different processing times. Problem: assign each operation to eligible machine s.t. maximal completion time (makespan) of all jobs is minimized. It can be presented as $FJ||Cmax$ by 3-field notations [6].

  – FJSP khác với JSSP ở chỗ nó cho phép 1 thao tác được xử lý bởi bất kỳ máy nào trong 1 tập hợp nhất định & các máy khác nhau có thể có thời gian xử lý khác nhau. Vấn đề: gán mỗi thao tác cho máy đủ điều kiện, thời gian hoàn thành tối đa (makespan) của tất cả các công việc được giảm thiểu. Nó có thể được biểu diễn dưới dạng $FJ||Cmax$ theo ký hiệu 3 trường [6].

  In public benchmark of FJSP, environment information is passed through an instance file. As is depicted in Fig. 1: An example of flexible job-shop scheduling problem (MK1 [4]), 1st line in instance file consists of 3 integers representing number of jobs, number of machines & average number of machines per operation (optional), resp. Each of following lines describes information of a job where 1st integer is number of operations & followed integers depicts operation information. Operation information includes 2 integers: 1st represents index of a machine & 2nd is processing time of operation on this machine. Processing operation order in a job is advanced from left to right.

  – Trong chuẩn mực công khai của FJSP, thông tin môi trường được truyền qua 1 tệp thể hiện. Như được mô tả trong Hình 1: Ví dụ về bài toán lập lịch xưởng công việc linh hoạt (MK1 [4]), dòng đầu tiên trong tệp thể hiện bao gồm 3 số nguyên biểu diễn số lượng công việc, số lượng máy & số lượng máy trung bình trên mỗi thao tác (tùy chọn), tương ứng. Mỗi dòng sau mô tả thông tin của 1 công việc, trong đó số nguyên đầu tiên là số lượng thao tác & các số nguyên tiếp theo mô tả thông tin thao tác. Thông tin thao tác bao gồm 2 số nguyên: số 1 biểu diễn chỉ số của 1 máy & số 2 là thời gian xử lý thao tác trên máy này. Thứ tự thao tác trong 1 công việc được sắp xếp theo thứ tự từ trái sang phải.

- 2.2. Data structure for FJSP. In this paper, based on benchmark instance file, propose a new & efficient storage structure for FJSP instances. Scheduling information is represented by a 2D table where job information is described in rows & column records processing stage information. Each element in this table includes 2 sets: machine set & remaining processing time set. Fig. 2: Storage structure of FJSP instance. demonstrates an example of a FJSP instance where storage structure is marked with a red box. It is efficient to retrieve any operation information using job index & processing stage index.

  – Cấu trúc dữ liệu cho FJSP. Trong bài báo này, dựa trên tệp phiên bản chuẩn, chúng tôi đề xuất 1 cấu trúc lưu trữ mới & hiệu quả cho các phiên bản FJSP. Thông tin lập lịch được biểu diễn bằng 1 bảng 2D, trong đó thông tin công việc được mô tả theo hàng & cột ghi lại thông tin giai đoạn xử lý. Mỗi phần tử trong bảng này bao gồm 2 tập hợp: tập máy & tập thời gian xử lý còn lại. Hình 2: Cấu trúc lưu trữ của phiên bản FJSP. minh họa 1 ví dụ về phiên bản FJSP, trong đó cấu trúc lưu trữ được đánh dấu bằng hộp đỏ. Việc truy xuất bất kỳ thông tin thao tác nào bằng chỉ mục công việc & chỉ mục giai đoạn xử lý đều hiệu quả.

- 2.3. Rules of state updating. In order to accurately record state changes of each scheduling step, 4 rules of state updating are defined based on type of operations. Job operations are divided into 4 categories: operations on processing, waiting operations without predecessors, completed operations & operations whose predecessors have not been completed. 4 rules are listed as follows & Fig. 3: Use of state update rules on a FJSP instance at time 4 provides an example to demonstrate use of these rules.

  – Quy tắc cập nhật trạng thái. Để ghi lại chính xác các thay đổi trạng thái của mỗi bước lập lịch, 4 quy tắc cập nhật trạng thái được xác định dựa trên loại thao tác. Các thao tác công việc được chia thành 4 loại: thao tác đang xử lý, thao tác chờ không có thao tác tiền nhiệm, thao tác đã hoàn thành & thao tác có thao tác tiền nhiệm chưa hoàn thành. 4 quy tắc được liệt kê như sau & Hình 3: Sử dụng quy tắc cập nhật trạng thái trên 1 thể hiện FJSP tại thời điểm 4 cung cấp 1 ví dụ để minh họa việc sử dụng các quy tắc này.

  * Rule 1: For an operation on processing, machine set has only 1 element which is negative index of selected machine while remaining time is recorded in remaining processing time set & its value decreases as time advances until completion of this operation.
    – Quy tắc 1: Đối với 1 thao tác xử lý, tập hợp máy chỉ có 1 phần tử là chỉ số âm của máy được chọn trong khi thời gian còn lại được ghi vào tập hợp thời gian xử lý còn lại & giá trị của nó giảm dần theo thời gian cho đến khi hoàn thành thao tác này.
  * Rule 2: For a waiting operation without predecessors whose needed machine is occupied, elements in machine set are all negative index of machines & elements in remaining processing time set stay unchanged. When needed machine is released, value of machine index is restored to be positive.
    – Quy tắc 2: Đối với 1 thao tác chờ không có tiền nhiệm mà máy cần thiết đang bị chiếm dụng, các phần tử trong tập máy đều có chỉ số máy âm & các phần tử trong tập thời gian xử lý còn lại không đổi. Khi máy cần thiết được giải phóng, giá trị chỉ số máy sẽ được khôi phục thành dương.
  * Rule 3: For a completed operation, values in machine set & remaining processing time set equal to negative value of machine index & 0, resp.
    – Quy tắc 3: Đối với 1 hoạt động đã hoàn thành, các giá trị trong bộ máy & thời gian xử lý còn lại được đặt bằng giá trị âm của chỉ số máy & 0, tương ứng.
  * Rule 4: For operations whose predecesors have not been proposed, values in machine set & remaining processing time set remain unchanged.
    – Quy tắc 4: Đối với các hoạt động mà hoạt động trước đó chưa được đề xuất, các giá trị trong tập hợp máy & thời gian xử lý còn lại vẫn không thay đổi.

This representation for FJSP instance can be recorded in a instance file & reloaded to new environment, i.e., it can be applied to nonzero or re-entrant scenarios.

– Biểu diễn này cho phiên bản FJSP có thể được ghi lại trong tệp phiên bản & tải lại vào môi trường mới, tức là có thể áp dụng cho các trường hợp khác không hoặc có thể nhập lại.

- 2.4. Simulation algorithm for FJSP. In this paper, proposed scheduling environment model is a simulation model based on chronological discrete events. There is a timer used to record current time & triggering time of events are recorded in a state variable. Once current time reaches triggering time of any events, this event will occur & corresponding event response program will be executed. Detailed process is illustrated in Algorithm 1: Simulation algorithm for FJSP.

  – Thuật toán mô phỏng cho FJSP. Trong bài báo này, mô hình môi trường lập lịch được đề xuất là 1 mô hình mô phỏng dựa trên các sự kiện rời rạc theo trình tự thời gian. Có 1 bộ đếm thời gian được sử dụng để ghi lại thời gian hiện tại & thời gian kích hoạt của các sự kiện được ghi lại trong 1 biến trạng thái. Khi thời gian hiện tại đạt đến thời gian kích hoạt của bất kỳ sự kiện nào, sự kiện này sẽ xảy ra & chương trình phản hồi sự kiện tương ứng sẽ được thực thi. Quy trình chi tiết được minh họa trong Thuật toán 1: Thuật toán mô phỏng cho FJSP.

  Proposed algorithm is mainly composed of 4 parts: selection of jobs & machines, state update of jobs & machines, time advance (3rd while statement) & machine release (4th while statement). In 1st part, decision action is divided into PDRs for job & machine selection & then specific job number & machine number are derived from PDRs. After allocating jobs & machines, their states are updated using state variables & table dictionaries. E.g., completion time of this job operation (or release time of machine) is recorded in `next_time_on_machine` & job-machine allocation information is recorded in `job_on_machine`. Once a job is allocated on a machine, state of this job is `assignable_job` changes to 0 & states of jobs which need that machine, are updated. If candidate machines of a job are all occupied by other jobs this job will become not assignable.

  – Thuật toán đề xuất chủ yếu bao gồm 4 phần: lựa chọn công việc & máy, cập nhật trạng thái của công việc & máy, tiến độ thời gian (câu lệnh while thứ 3) & giải phóng máy (câu lệnh while thứ 4). Trong phần 1, hành động quyết định được chia thành các PDR để lựa chọn công việc & máy & sau đó số công việc cụ thể & số máy được lấy từ PDR. Sau khi phân bổ công việc & máy, trạng thái của chúng được cập nhật bằng các biến trạng thái & từ điển bảng. Ví dụ: thời gian hoàn thành của hoạt động công việc này (hoặc thời gian giải phóng máy) được ghi lại trong `next_time_on_machine` & thông tin phân bổ công việc-máy được ghi lại trong `job_on_machine`. Sau khi 1 công việc được phân bổ trên 1 máy, trạng thái của công việc này là `assignable_job` thay đổi thành 0 & trạng thái của các công việc cần máy đó được cập nhật. Nếu tất cả các máy ứng viên của 1 công việc đều bị các công việc khác chiếm giữ thì công việc này sẽ không thể gán được.

  When there are no assignable jobs, time advance stage is coming. In this part, current time is 1st updated based on values in `next_time_on_machine` (if current time is smaller than any value in `next_time_on_machine`, take smallest value, otherwise take next smallest of them). Length of time step that needs to advance is then calculated based on `next_time_on_machine` & current time. Finally, next time of machines whose next time is slower than current time is advanced to current time. Machine release part releases machines whose release time reaches current time & updates status of jobs & machines. When current job operation is completed, occupied machine becomes idle & current operation of this job move to next operation. States of jobs & machines are updated in `assignable_job, job_on_machine`. If all operation of current job is completed, this job become not assignable & if machine needed for next operation is occupied, this job is still not assignable.

  – Khi không có công việc nào có thể gán được, giai đoạn tiến độ thời gian sẽ diễn ra. Trong phần này, thời gian hiện tại được cập nhật lần đầu tiên dựa trên các giá trị trong `next_time_on_machine` (nếu thời gian hiện tại nhỏ hơn bất kỳ giá trị nào trong `next_time_on_machine`, hãy lấy giá trị nhỏ nhất, nếu không thì lấy giá trị nhỏ tiếp theo trong số chúng). Sau đó, độ dài bước thời gian cần tiến lên được tính toán dựa trên `next_time_on_machine` & thời gian hiện tại. Cuối cùng, thời gian tiếp theo của các máy có thời gian tiếp theo chậm hơn thời gian hiện tại sẽ được tiến lên thời gian hiện tại. Bộ phận nhả máy sẽ nhả các máy có thời gian nhả đạt đến thời gian hiện tại & cập nhật trạng thái của công việc & máy. Khi thao tác công việc hiện tại hoàn thành, máy đang được chiếm dụng sẽ trở nên nhàn rỗi & thao tác hiện tại của công việc này chuyển sang thao tác tiếp theo. Trạng thái của công việc & máy được cập nhật trong `assignable_job, job_on_machine`. Nếu tất cả thao tác của công việc hiện tại đã hoàn thành, công việc này sẽ không thể gán được & nếu máy cần cho thao tác tiếp theo bị chiếm dụng, công việc này vẫn không thể gán được.

- 3. A DRL scheduling framework for FJSP. In this sect, introduce overall DRL framework for FJSP, illustrated in Fig. 4: DRl framework for flexible job-shop scheduling problems. It is composed of:

  1. state representation based on 2 state variables
  2. PDR action space
  3. reward function based on scheduling area
  4. scheduling policy based on deep neural networks & a Softmax function
  5. PPO agent with an actor-critic learning architecture.

  – Khung lập lịch DRL cho FJSP. Trong phần này, giới thiệu khung DRL tổng thể cho FJSP, được minh họa trong Hình 4: Khung DRL cho các bài toán lập lịch xưởng linh hoạt. Khung này bao gồm:

  1. biểu diễn trạng thái của mục dựa trên 2 biến trạng thái
  2. Không gian hành động PDR

3. Hàm thưởng dựa trên vùng lập lịch
4. Chính sách lập lịch dựa trên mạng nơ-ron sâu & 1 hàm Softmax
5. Tác tử PPO với kiến trúc học tập tác nhân-phê bình.

- 3.1. State representation based on state variables. State representation depicts features of scheduling environment & determines size of state space, which is very crucial in RL scheduling methods. Various state representations for FJSP are presented in literature & are mainly focused on disjunctive graph [15, 22], feature matrix [24] & handcrafted state variables [8, 10, 14]. However, whether it is node features of disjunctive graph or matrix, or variable features, state features of job shop are mainly manually designed, which requires a large amount of processional domain knowledge, computing resources & time.

  – Biểu diễn trạng thái dựa trên biến trạng thái. Biểu diễn trạng thái mô tả các đặc điểm của môi trường lập lịch & xác định kích thước của không gian trạng thái, điều này rất quan trọng trong các phương pháp lập lịch RL. Nhiều biểu diễn trạng thái cho FJSP được trình bày trong tài liệu & chủ yếu tập trung vào đồ thị rời rạc [15, 22], ma trận đặc trưng [24] & các biến trạng thái thủ công [8, 10, 14]. Tuy nhiên, cho dù là đặc trưng nút của đồ thị rời rạc hay ma trận, hay đặc trưng biến, đặc trưng trạng thái của job shop chủ yếu được thiết kế thủ công, đòi hỏi lượng lớn kiến thức về miền xử lý, tài nguyên tính toán & thời gian.

  In this paper, a novel short state representation is proposed to reduce feature redundancy & to avoid handcrafted feature design & feature selection, which requires less computation time of environment state variables & scheduling policy networks. 2 state variables: `assignable_job, completed_op_of_job` are selected from Algorithm 1 as state features of job shop scheduling environment. Variable `assignable_job` is a Boolean vector to represent whether a job can be allocated or not. `completed_op_of_job` represents number of completed operation of a job & this value is then scaled by maximum number of operations in jobs to be in range $[0, 1]$. Length of both variables equals number of jobs. Finally, in order to represent environment state, scaled variables are simply concatenated to a vector whose length equals 2 times of number of jobs.

  – Trong bài báo này, 1 biểu diễn trạng thái ngắn mới được đề xuất để giảm sự dư thừa tính năng & để tránh thiết kế tính năng thủ công & lựa chọn tính năng, đòi hỏi ít thời gian tính toán hơn của các biến trạng thái môi trường & mạng chính sách lập lịch. 2 biến trạng thái: `assignable_job, completed_op_of_job` được chọn từ Thuật toán 1 làm các tính năng trạng thái của môi trường lập lịch job shop. Biến `assignable_job` là 1 vectơ Boolean để biểu diễn liệu 1 công việc có thể được phân bổ hay không. `completed_op_of_job` biểu diễn số lượng hoạt động đã hoàn thành của 1 công việc & giá trị này sau đó được chia tỷ lệ theo số lượng hoạt động tối đa trong các công việc nằm trong phạm vi $[0, 1]$. Độ dài của cả hai biến bằng số lượng công việc. Cuối cùng, để biểu diễn trạng thái môi trường, các biến được chia tỷ lệ chỉ cần được nối thành 1 vectơ có độ dài bằng 2 lần số lượng công việc.

  There are many advantages of our state representation:

  1. length of state features is much less than that in previous research which means less computation time of environment state variables & scheduling policy networks
  2. state features are unique in a scheduling solution, which suggests that state features can be easily distinguished by scheduling agent
  3. state features are directly derived from 2 state variables in Algorithm 1, avoiding massive experiments on feature design & selection.

  – Biểu diễn trạng thái của chúng tôi có nhiều ưu điểm:

  1. Độ dài của các đặc trưng trạng thái ngắn hơn nhiều so với nghiên cứu trước đây, đồng nghĩa với việc giảm thời gian tính toán các biến trạng thái môi trường & mạng lưới chính sách lập lịch.
  2. Các đặc trưng trạng thái của mục là duy nhất trong 1 giải pháp lập lịch, điều này cho thấy các đặc trưng trạng thái có thể dễ dàng được phân biệt bởi tác nhân lập lịch.
  3. Các đặc trưng trạng thái của mục được suy ra trực tiếp từ 2 biến trạng thái trong Thuật toán 1, tránh được các thử nghiệm lớn về thiết kế & lựa chọn đặc trưng.

- 3.2. Action space based on PDR. In DRl scheduling methods based on single agent, action is output of scheduling policy networks, which is usually an integer. Since FJSP needs to select a job & a machine at each decision, decompose this integer into 2 parts by dividing it by number of PDRs for machine selection where quotient is index of PDR for jobs assignment & remainder represents index of PDR for machine selection.

  – Không gian hành động dựa trên PDR. Trong các phương pháp lập lịch DRl dựa trên tác nhân đơn lẻ, hành động là đầu ra của mạng chính sách lập lịch, thường là 1 số nguyên. Vì FJSP cần chọn 1 công việc & 1 máy tại mỗi quyết định, hãy phân tích số nguyên này thành 2 phần bằng cách chia nó cho số PDR để chọn máy, trong đó thương là chỉ số của PDR để gán công việc & phần dư biểu thị chỉ số của PDR để chọn máy.

  In this paper, 6 PDRs are selected to construct action space for simplicity of implementation & ease of generalization. For selecting a job, 4 PDRs are selected directly from literature [27] including Shortest Processing (SPT), Most Work Remaining (MWRK), Most Operations Remaining (MOR) & Minimum ratio of Flow Due Date to Most Work Remaining (FDD/MWRK). Longest Remaining Machine time not including current operation processing time (LRM) is selected from [11] due to its excellent performance. First In First Out (FIFO) is selected because it is widely used in various scheduling problems. In addition, action space for selecting machine given a job is composed of SPT & Longest Processing Time (LPT). So that total size of action space equals number of PDRs for selecting jobs multiplied by number of PDRs for selecting machines. Definitions of these PDR are listed as follows:

* SPT: $\min Z_{ij} = p_{ij}$
* LPT: $\max Z_{ij} = p_{ij}$
* FDD/MWKR: $\min Z_{ij} = \frac{\sum_1^j p_{ij}}{\sum_j^{n_i} p_{ij}}$
* MOR: $\max Z_{ij} = n_i - j + 1$
* LRM: $\max Z_{ij} = \sum_{j+1}^{n_i} p_{ij}$
* FIFO: $\max Z_{ij} = t - Rt_i$

where Z_{ij} is priority index of operation O_{ij}. p_{ij} is processing time of operation O_{ij}. n_i: number of operations for job J_i. j: number of completed operations. t: current time & Rt_i: release time of J_i.

– Trong bài báo này, 6 PDR được chọn để xây dựng không gian hành động nhằm đơn giản hóa việc triển khai & dễ dàng khái quát hóa. Để lựa chọn 1 công việc, 4 PDR được chọn trực tiếp từ tài liệu [27] bao gồm Xử lý ngắn nhất (SPT), Công việc còn lại nhiều nhất (MWRK), Hoạt động còn lại nhiều nhất (MOR) & Tỷ lệ tối thiểu giữa Ngày đến hạn luồng & Công việc còn lại nhiều nhất (FDD/MWRK). Thời gian máy còn lại dài nhất không bao gồm thời gian xử lý hoạt động hiện tại (LRM) được chọn từ [11] do hiệu suất tuyệt vời của nó. Vào trước ra trước (FIFO) được chọn vì nó được sử dụng rộng rãi trong nhiều vấn đề lập lịch khác nhau. Ngoài ra, không gian hành động để lựa chọn máy cho 1 công việc bao gồm SPT & Thời gian xử lý dài nhất (LPT). Do đó, tổng kích thước của không gian hành động bằng số PDR để lựa chọn công việc nhân với số PDR để lựa chọn máy. Định nghĩa của các PDR này được liệt kê như sau:

* SPT: $\min Z_{ij} = p_{ij}$
* LPT: $\max Z_{ij} = p_{ij}$
* FDD/MWKR: $\min Z_{ij} = \frac{\sum_1^j p_{ij}}{\sum_j^{n_i} p_{ij}}$
* MOR: $\max Z_{ij} = n_i - j + 1$
* LRM: $\max Z_{ij} = \sum_{j+1}^{n_i} p_{ij}$
* FIFO: $\max Z_{ij} = t - Rt_i$

trong đó Z_{ij} là chỉ số ưu tiên của phép toán O_{ij}. p_{ij} là thời gian xử lý của thao tác O_{ij}. n_i: số thao tác cho công việc J_i. j: số thao tác đã hoàn thành. t: thời gian hiện tại & Rt_i: thời gian giải phóng J_i.

- 3.3. Reward function based on scheduling area. In this paper, propose a comprehensible reward function based on scheduling area (1)

$$\text{reward}(s_t, a_t) = -p_{a,j} - \sum_M \text{vacancy}_m(s_t, s_{t+1})$$

where processing time of allocated operations & time vacancy of all machines is computed after each dispatching action, where s_t: current state & s_{t+1} is next state after applying action a_t; a_t is jth operation of a job with processing time $p_{a,j}$; vacancy(s_t, s_{t+1}) is a function returning total time vacancy on machine set M while transitioning from state s_t to s_{t+1}.

– Hàm thưởng dựa trên vùng lập lịch. Trong bài báo này, chúng tôi đề xuất 1 hàm thưởng dễ hiểu dựa trên vùng lập lịch (1)

$$\text{reward}(s_t, a_t) = -p_{a,j} - \sum_M \text{vacancy}_m(s_t, s_{t+1})$$

trong đó thời gian xử lý của các tác vụ được phân bổ & thời gian trống của tất cả các máy được tính sau mỗi hành động điều phối, trong đó s_t: trạng thái hiện tại & s_{t+1} là trạng thái tiếp theo sau khi áp dụng tác vụ a_t; a_t là tác vụ thứ j của 1 tác vụ có thời gian xử lý $p_{a,j}$; vacancy(s_t, s_{t+1}) là 1 hàm trả về tổng thời gian trống trên tập máy M khi chuyển từ trạng thái s_t sang s_{t+1}.

Proposed reward function is motivated by fact: total processing time of all jobs & time vacancy on all machines constructs scheduling area of all machines which equals maximum make-span multiplied by number of machines. As is demonstrated in Fig. 5: An example to show the scheduling area., scheduling area is composed of total processing time of all jobs (shaded area) & time vacancy on all machines (white color area). Relationship between accumulated reward & maximum scheduling makespan is derived from (2)

$$R = -a - b = -(a + b) = -S = -|M| * \text{makespan}$$

where R is accumulated reward, a: total processing time of all jobs, b: total time vacancy on all machines, S: scheduling time area & $|M|$: number of machines.

– Hàm phần thưởng được đề xuất dựa trên thực tế: tổng thời gian xử lý của tất cả các công việc & thời gian trống trên tất cả các máy tạo nên vùng lập lịch của tất cả các máy, bằng khoảng thời gian hoàn thành tối đa nhân với số máy. Như được minh họa trong Hình 5: Ví dụ minh họa vùng lập lịch., vùng lập lịch bao gồm tổng thời gian xử lý của tất cả các công việc (vùng tô bóng) & thời gian trống trên tất cả các máy (vùng màu trắng). Mối quan hệ giữa phần thưởng tích lũy & khoảng thời gian hoàn thành tối đa được suy ra từ (2)

$$R = -a - b = -(a + b) = -S = -|M| * \text{khongthigianhonthnh}$$

trong đó R là phần thưởng tích lũy, a: tổng thời gian xử lý của tất cả các công việc, b: tổng thời gian trống trên tất cả các máy, S: vùng thời gian lập lịch & $|M|$: số máy.

Obviously shown from (2): total reward & maximum makespan are negatively linearly dependent & coefficient is number of machines. I.e., minimizing maximum scheduling makespan is equivalent to maximizing total reward.

– Hiển nhiên từ (2): tổng phần thưởng & thời gian hoàn thành tối đa phụ thuộc tuyến tính âm & hệ số là số máy. Nghĩa là, việc giảm thiểu thời gian hoàn thành tối đa tương đương với việc tối đa hóa tổng phần thưởng.

- 3.4. Model training method based on PPO. In order to strengthen representation ability of RL scheduling method, scheduling policy is usually represented by DNNs e.g. CNN, RNN, & MLP. In our method, state feature vector is 1st fed to MLP to obtain a scalar score for each action & a Softmax function is then applied to output a distribution over computed score, which is shown in (3)

$$p(a_t|s_t) = \text{Softmax}(\text{MLP}_{\pi_\theta}(s_t)),$$

where $p(a_t|s_t)$ is selection probability of action a_t at time t in state s_t & θ is parameter of scheduling policy π. Structure of our scheduling policy is demonstrate in Fig. 4, which constructs actor network of PPO agent. Similar to actor network, critic network is implemented by MLP with 1 hidden layer. Parameters of our scheduling policy are learned by a clipped PPO whose loss function is expressed in (4)

$$L(\theta) = E_t[\min(r_t(\theta)A_t, \text{clip}(r_t(\theta), 1 \pm \epsilon)A_t)]$$

where $r_t(\theta) = \dfrac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$, A_t is an estimator of advantage function at time step t, clip is a clipping function & ϵ is a hyper parameter which is used to limit boundary of objective function.

– Phương pháp huấn luyện mô hình dựa trên PPO. Để tăng cường khả năng biểu diễn của phương pháp lập lịch RL, chính sách lập lịch thường được biểu diễn bằng các DNN, ví dụ: CNN, RNN, & MLP. Trong phương pháp của chúng tôi, vectơ đặc trưng trạng thái đầu tiên được đưa vào MLP để thu được điểm số vô hướng cho mỗi hành động & sau đó áp dụng hàm Softmax để đưa ra phân phối trên điểm số đã tính toán, được thể hiện trong (3)

$$p(a_t|s_t) = \text{Softmax}(\text{MLP}_{\pi_\theta}(s_t)),$$

trong đó $p(a_t|s_t)$ là xác suất lựa chọn hành động a_t tại thời điểm t trong trạng thái s_t & θ là tham số của chính sách lập lịch π. Cấu trúc của chính sách lập lịch của chúng tôi được minh họa trong Hình 4, trong đó xây dựng mạng lưới tác nhân của tác tử PPO. Tương tự như mạng diễn viên, mạng phê bình được triển khai bằng MLP với 1 lớp ẩn. Các tham số của chính sách lập lịch của chúng tôi được học bởi 1 PPO bị cắt xén có hàm mất mát được biểu thị trong (4)

$$L(\theta) = E_t[\min(r_t(\theta)A_t, \text{clip}(r_t(\theta), 1 \pm \epsilon)A_t)]$$

trong đó $r_t(\theta) = \dfrac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$, A_t là 1 ước lượng của hàm lợi thế tại bước thời gian t, clip là 1 hàm cắt xén & ϵ là 1 siêu tham số được sử dụng để giới hạn biên của hàm mục tiêu.

Detailed training process is provided by Algorithm 2: Model training method based on PPO. Training process includes 2 aspects: data collection & policy learning. When collecting training data, T independent complete trajectories of scheduling are generated & they are collected in memory buffer M. On policy learning stage, training data are used for K times. At each time, these data are randomly divided into batches whose length is related to scale of instances & scheduling agent learns from each batch data. After replaying these experience samples evenly, prioritized experience replay is performed for C times. Training process will stop until episode reaches maximum iterations or result is convergent or scheduling is time out. Define: result is thought as convergent if makespan values are same in 30 decision steps & training time is limited in an hour.

– Quá trình đào tạo chi tiết được cung cấp bởi Thuật toán 2: Phương pháp đào tạo mô hình dựa trên PPO. Quá trình đào tạo bao gồm 2 khía cạnh: thu thập dữ liệu & học chính sách. Khi thu thập dữ liệu đào tạo, T quỹ đạo hoàn chỉnh độc lập của lập lịch được tạo & chúng được thu thập trong bộ đệm M. Ở giai đoạn học chính sách, dữ liệu đào tạo được sử dụng trong K lần. Tại mỗi thời điểm, những dữ liệu này được chia ngẫu nhiên thành các lô có độ dài liên quan đến quy mô của các trường hợp & tác nhân lập lịch học từ mỗi dữ liệu lô. Sau khi phát lại các mẫu trải nghiệm này 1 cách đồng đều, việc phát lại trải nghiệm được ưu tiên sẽ được thực hiện trong C lần. Quá trình đào tạo sẽ dừng lại cho đến khi tập đạt số lần lặp tối đa hoặc kết quả hội tụ hoặc lập lịch hết thời gian. Định nghĩa: kết quả được coi là hội tụ nếu các giá trị makespan giống nhau trong 30 bước quyết định & thời gian đào tạo bị giới hạn trong 1 giờ.

- 4. Experiments. In this sect, in order to show effectiveness of our DRL scheduling environment for FJSP (Algorithm 1) & to evaluate performance of proposed DRL scheduling methods, a group of experiments are performed on public FJSP benchmarks with various sizes & results are compared with different types of scheduling methods from recent literature. Finally, training details e.g. training time are demonstrated.

  – Trong phần này, để chứng minh tính hiệu quả của môi trường lập lịch DRL cho FJSP (Thuật toán 1) & đánh giá hiệu suất của các phương pháp lập lịch DRL được đề xuất, 1 nhóm các thử nghiệm được thực hiện trên các chuẩn FJSP công khai với nhiều kích thước khác nhau & kết quả được so sánh với các loại phương pháp lập lịch khác nhau từ các tài liệu gần đây. Cuối cùng, các chi tiết đào tạo, ví dụ như thời gian đào tạo, sẽ được trình bày.

  - 4.1. Benchmark instances & baseline models. In this paper, 2 well-known benchmarks of FJSP are used to evaluate our proposed methods including MK instances (MK01-MK10) in [4] & 3 group of LA instances (edata, rdata, & vdata each with 40 instances) in [12].

– Trong bài báo này, 2 chuẩn mực nổi tiếng của FJSP được sử dụng để đánh giá các phương pháp đề xuất của chúng tôi bao gồm các thể hiện MK (MK01-MK10) trong [4] & 3 nhóm thể hiện LA (edata, rdata, & vdata mỗi nhóm có 40 thể hiện) trong [12].

This paper compared with PDR, exact solver, meta-heuristic & DRL models. 6 PDRs are used to compare scheduling results where 4 out of 6 PDRs are compared in old scheduling environment & our proposed environment. Also compare with well-known Google OR-Tools which is a powerful constraint programming solver showing strong performance in solving industrial scheduling problems. For meta-heuristic scheduling methods, 2 recent improved Genetic Algorithms are selected from [18] & [5]. For DRL scheduling methods, compared with competing models in recent 3 years, including 4 DRl methods proposed by Feng et al.[9], Zeng et al.[26], Song et al.[22] & Lei et al.[15] resp., GSMA model [13] which is a MARL scheduling method & obtains state-of-art results, DANILE model [23] which is based on attention mechanism.

– Bài báo này so sánh với PDR, bộ giải chính xác, mô hình meta-heuristic & DRL. 6 PDR được sử dụng để so sánh kết quả lập lịch trong đó 4 trong số 6 PDR được so sánh trong môi trường lập lịch cũ & môi trường đề xuất của chúng tôi. Cũng so sánh với Google OR-Tools nổi tiếng, 1 bộ giải lập trình ràng buộc mạnh mẽ cho thấy hiệu suất cao trong việc giải quyết các vấn đề lập lịch công nghiệp. Đối với các phương pháp lập lịch meta-heuristic, 2 Thuật toán di truyền được cải tiến gần đây được chọn từ [18] & [5]. Đối với các phương pháp lập lịch DRL, so sánh với các mô hình cạnh tranh trong 3 năm gần đây, bao gồm 4 phương pháp DRl do Feng et al.[9], Zeng et al.[26], Song et al.[22] & Lei et al.[15] đề xuất, tương ứng, mô hình GSMA [13] là 1 phương pháp lập lịch MARL & thu được kết quả tiên tiến, mô hình DANILE [23] dựa trên cơ chế chú ý.

- 4.2. Model configurations. PPO agent adopts actor-critic architecture where actor networks represent scheduling policy & critic networks calculate state-value of policy networks. Actor network is implemented by MLP & Softmax function while critic network is only represented by MLP. Both networks are optimized by Adam optimizer, use ReLU as activation function & have only 1 hidden layer with dynamic hidden dimension which equals length of state features. For each problem size, train policy network for $\leq$ 8000 iterations, each of which contains 9 independent trajectories (i.e., complete scheduling process of instances) & use a dynamic batch size which equals 2 times of scale (total number of operations of all jobs) of an instance. For PPO, set epochs of updating network to 10 & clipping parameter epsilon to 0.2. Set discount factor γ to 0.999 & learning rate are $1e-3, 3e-3$ for actor & critic network resp. For prioritized experience replay, parameter α is set to 0.6, value of β anneals from 0.4 to 1, number of prioritized experience replay C is set to 1, & number of convergence training steps is 2000.

  – Tác nhân PPO áp dụng kiến trúc actor-critic trong đó các mạng actor biểu diễn chính sách lập lịch & các mạng critic tính toán giá trị trạng thái của các mạng policy. Mạng actor được triển khai bởi hàm MLP & Softmax trong khi mạng critic chỉ được biểu diễn bởi MLP. Cả hai mạng đều được tối ưu hóa bởi trình tối ưu hóa Adam, sử dụng ReLU làm hàm kích hoạt & chỉ có 1 lớp ẩn với chiều ẩn động bằng với độ dài của các đặc trưng trạng thái. Đối với mỗi kích thước vấn đề, hãy huấn luyện mạng policy cho $\leq$ 8000 lần lặp, mỗi lần lặp chứa 9 quỹ đạo độc lập (tức là quy trình lập lịch hoàn chỉnh của các trường hợp) & sử dụng kích thước lô động bằng 2 lần quy mô (tổng số thao tác của tất cả các công việc) của 1 trường hợp. Đối với PPO, đặt các kỷ nguyên cập nhật mạng thành 10 & tham số cắt epsilon thành 0,2. Đặt hệ số chiết khấu γ thành $0,999$ & tốc độ học là $1e-3, 3e-3$ tương ứng với mạng actor & critic. Đối với phát lại trải nghiệm được ưu tiên, tham số α được đặt thành 0,6, giá trị của β được ủ từ 0,4 đến 1, số lần phát lại trải nghiệm được ưu tiên C được đặt thành 1, & số bước đào tạo hội tụ là 2000.

- 4.3. Results analysis. In order to evaluate our proposed scheduling environment, 1st test performance of 6 PDRs in our proposed scheduling environment on MK benchmark instances where 6 PDRs are used to select a job while SPT is used for selection of machines. Results are shown in Table 1: Scheduling results of PDRs on MK benchmark instances. Widely-used 4 PDRs (SPT, MWKR, FIFO, MOR) are performed in our environment & results are compared with that in old scheduling environment where their best results are selected from literature [22]. Symbol "-" means: specific results are not recorded in literature. Best scheduling result of all 6 PDRs on each instance are recorded in minPDR row. Besides, results of 2 DRL scheduling methods proposed resp. by Feng et al.[9] & Zeng et al.[26] are also compared in Table 1 & their results are directly from their literature instead of our implementation environment.

  – Để đánh giá môi trường lập lịch đề xuất của chúng tôi, hiệu suất thử nghiệm đầu tiên của 6 PDR trong môi trường lập lịch đề xuất của chúng tôi trên các phiên bản chuẩn MK trong đó 6 PDR được sử dụng để chọn 1 công việc trong khi SPT được sử dụng để chọn máy. Kết quả được hiển thị trong Bảng 1: Kết quả lập lịch của PDR trên các phiên bản chuẩn MK. 4 PDR được sử dụng rộng rãi (SPT, MWKR, FIFO, MOR) được thực hiện trong môi trường của chúng tôi & kết quả được so sánh với kết quả trong môi trường lập lịch cũ, trong đó kết quả tốt nhất của chúng được chọn từ tài liệu [22]. Ký hiệu "-" có nghĩa là: kết quả cụ thể không được ghi lại trong tài liệu. Kết quả lập lịch tốt nhất của tất cả 6 PDR trên mỗi phiên bản được ghi lại trong hàng minPDR. Bên cạnh đó, kết quả của 2 phương pháp lập lịch DRL do Feng et al.[9] & Zeng et al.[26] đề xuất cũng được so sánh trong Bảng 1 & kết quả của chúng được lấy trực tiếp từ tài liệu của họ thay vì môi trường triển khai của chúng tôi.

  As shown in table 1, results indicate: performance of SPT, FIFO, MOR in our environment is improved while performance of MWKR is suddenly worse than before. LRM obtained best average results among all 6 PDRs & even better than some DRL scheduling methods e.g. models proposed by Feng et al. & Zeng et al., which is surprisingly interesting. Best result on different instances is distributed in different PDRs, e.g. MWKR obtained best results on MK1 & FIFO performs best on MK3. So that, minPDR can get better results than LRM.

  – Như thể hiện trong bảng 1, kết quả cho thấy: hiệu suất của SPT, FIFO, MOR trong môi trường của chúng tôi được cải thiện trong khi hiệu suất của MWKR đột nhiên kém hơn trước. LRM đạt được kết quả trung bình tốt nhất trong số tất cả 6 PDR & thậm chí còn tốt hơn 1 số phương pháp lập lịch DRL, ví dụ như các mô hình do Feng & cộng sự đề xuất & Zeng

& cộng sự, điều này thật đáng ngạc nhiên. Kết quả tốt nhất trên các trường hợp khác nhau được phân phối trong các PDR khác nhau, ví dụ: MWKR đạt được kết quả tốt nhất trên MK1 & FIFO hoạt động tốt nhất trên MK3. Do đó, minPDR có thể đạt được kết quả tốt hơn LRM.

2nd group of experiments are also performed on MK benchmark instances to evaluate generality of our proposed DRL scheduling agent in our environment. Train these instances independently for 5 times & average results are recorded in "Ours" column. As shown in Table 2, compare performance of our proposed DRL scheduling model with exact solver (OR-Tools), meta-heuristic scheduling methods (2SGA[18] & SLGA [5]), DRL scheduling methods (GMAS[13], DANILE[23], & models proposed by Song et al.[22]) & minPDR method. LB & UB columns resp. record lower bound & upper bound scheduling results in literature. Results in these compared methods are directly from literature except results of PDRs which are performed in our environment.

– Nhóm thí nghiệm thứ 2 cũng được thực hiện trên các trường hợp chuẩn MK để đánh giá tính tổng quát của tác nhân lập lịch DRL được đề xuất của chúng tôi trong môi trường của chúng tôi. Đào tạo các trường hợp này độc lập trong 5 lần & kết quả trung bình được ghi lại trong cột "Của chúng tôi". Như thể hiện trong Bảng 2, hãy so sánh hiệu suất của mô hình lập lịch DRL được đề xuất của chúng tôi với bộ giải chính xác (OR-Tools), các phương pháp lập lịch meta-heuristic (2SGA[18] & SLGA [5]), các phương pháp lập lịch DRL (GMAS[13], DANILE[23], & các mô hình do Song et al.[22] đề xuất) & phương pháp minPDR. Các cột LB & UB tương ứng ghi lại kết quả lập lịch giới hạn dưới & giới hạn trên trong tài liệu. Kết quả trong các phương pháp được so sánh này được lấy trực tiếp từ tài liệu, ngoại trừ kết quả của PDR được thực hiện trong môi trường của chúng tôi.

Average makespan is usually used to evaluate accuracy of scheduling solutions. As demonstrated in Table 2: Makespan performance of our DRL model on MK benchmark instances, GMAS which is a DRL scheduling method based on MARL, obtained best average result. Nevertheless, they used optimal results of each instance rather than average results. Average makespan of our proposed DRL model is smaller than SLGA, DANILE, Song, & minPDR methods & is larger than OR-Tools, 2SGA & GMAS methods, which shows: performance of DRL scheduling methods are getting close to meta-heuristic & exact solver. However, different from complex scheduling networks of GMAS, our model is simple & easy to train to converge, which is stable.

– Makespan trung bình thường được sử dụng để đánh giá độ chính xác của các giải pháp lập lịch. Như được minh họa trong Bảng 2: Hiệu suất Makespan của mô hình DRL của chúng tôi trên các trường hợp chuẩn MK, GMAS, 1 phương pháp lập lịch DRL dựa trên MARL, đã thu được kết quả trung bình tốt nhất. Tuy nhiên, họ đã sử dụng kết quả tối ưu của từng trường hợp thay vì kết quả trung bình. Makespan trung bình của mô hình DRL đề xuất của chúng tôi nhỏ hơn các phương pháp SLGA, DANILE, Song, & minPDR & lớn hơn các phương pháp OR-Tools, 2SGA & GMAS, điều này cho thấy: hiệu suất của các phương pháp lập lịch DRL đang tiến gần đến trình giải meta-heuristic & chính xác. Tuy nhiên, khác với các mạng lập lịch phức tạp của GMAS, mô hình của chúng tôi đơn giản & dễ huấn luyện để hội tụ, điều này rất ổn định.

Beside, evaluate convergence performance of our DRL scheduling method. As is demonstrated in Fig. 6: Taining trajectories & training time on MK benchmarks., our DRL model on all MK instances is convergent in half an hour on average & number of needed trajectories is < 900. Convergence time of 8 out of 10 instances is < 600 seconds, which is close to industrial time limitation.

– Ngoài ra, hãy đánh giá hiệu suất hội tụ của phương pháp lập lịch DRL của chúng tôi. Như được minh họa trong Hình 6: Định vị quỹ đạo & thời gian huấn luyện trên chuẩn MK., mô hình DRL của chúng tôi trên tất cả các trường hợp MK đều hội tụ trung bình trong nửa giờ & số quỹ đạo cần thiết là < 900. Thời gian hội tụ của 8 trên 10 trường hợp là < 600 giây, gần với giới hạn thời gian công nghiệp.

In order to show stability & generality of our proposed DRL scheduling agent in our environment, more experiments are performed on LA benchmark instances, including edta, rdata, & vdata, whose flexibility is getting increased. As shown in Table 3: Average makespan comparison LA instances, compare with various scheduling methods in literature e.g. exact solver OR-Tools, meta-heuristic 2SGA, DRL scheduling methods: DANILE, GMAS, Lei [15] & Song as well as PDR methods. Results of these methods are directly from literature. Results of PDR methods are from running in our proposed environment & only minimum result of 6 PDRs on each instance is recorded in minPDR column.

– Để chứng minh tính ổn định & tính tổng quát của tác nhân lập lịch DRL được đề xuất trong môi trường của chúng tôi, nhiều thử nghiệm hơn đã được thực hiện trên các phiên bản chuẩn LA, bao gồm edta, rdata, & vdata, với tính linh hoạt ngày càng tăng. Như được thể hiện trong Bảng 3: So sánh makespan trung bình của các phiên bản LA, hãy so sánh với các phương pháp lập lịch khác nhau trong tài liệu, ví dụ: bộ giải chính xác OR-Tools, meta-heuristic 2SGA, các phương pháp lập lịch DRL: DANILE, GMAS, Lei [15] & Song cũng như các phương pháp PDR. Kết quả của các phương pháp này được trích dẫn trực tiếp từ tài liệu. Kết quả của các phương pháp PDR được thực hiện trong môi trường được đề xuất của chúng tôi & chỉ có kết quả tối thiểu là 6 PDR trên mỗi phiên bản được ghi lại trong cột minPDR.

As is demonstrated in Table 3, our proposed DRL scheduling agent got smaller average makespan than DANILE, Lei, Song, & PDR models & obtained larger average makespan than OR-Tools, GMAS, & 2SGA, which shows competing performance of our proposed DRL scheduling agent in our environment. More interestingly, minPDR obtained smaller makespan than Lei model which is used to solve large-scale dynamic FJSP. That shows efficiency of our proposed environment again.

– Như được minh họa trong Bảng 3, tác nhân lập lịch DRL đề xuất của chúng tôi có makespan trung bình nhỏ hơn các mô hình DANILE, Lei, Song, & PDR & đạt makespan trung bình lớn hơn OR-Tools, GMAS, & 2SGA, điều này cho thấy hiệu suất cạnh tranh của tác nhân lập lịch DRL đề xuất trong môi trường của chúng tôi. Thú vị hơn, minPDR đạt makespan nhỏ hơn mô hình Lei, được sử dụng để giải các bài toán FJSP động quy mô lớn. Điều này 1 lần nữa cho thấy hiệu quả của môi trường chúng tôi đề xuất.

Finally, training time of our proposed DRL scheduling agent on edata, rdata, & vdata are depicted in Fig. 7: Training time on edata, rdata, & vdata instances. Our proposed DRL model can be trained to converge in an hour on all benchmark instances & total training time on edta, rdata, & vdata are 22064, 31650 & 40137 seconds, resp., which shows training time rises as increase of complicity of scheduling instances.

– Cuối cùng, thời gian đào tạo của tác nhân lập lịch DRL được đề xuất của chúng tôi trên edata, rdata, & vdata được mô tả trong Hình 7: Thời gian đào tạo trên các phiên bản edata, rdata, & vdata. Mô hình DRL được đề xuất của chúng tôi có thể được đào tạo để hội tụ trong 1 giờ trên tất cả các phiên bản chuẩn & tổng thời gian đào tạo trên edta, rdata, & vdata tương ứng là 22064, 31650 & 40137 giây, cho thấy thời gian đào tạo tăng lên khi mức độ phức tạp của các phiên bản lập lịch tăng lên.

- 5. Conclusion. In this paper, proposed a chronological discrete event simulation based DRL environment for FJSP. Scheduling process is described by a simulation algorithm where state changes are captured by state variables & reward function is calculated based on scheduling area of machines at each decision step. In this simulaiton environment, an end-to-end DRL framework is proposed based on actor-critic PPO, providing a flexible infrastructure for design of state representation, action space & scheduling policy networks. Besides, a simple DRL model for FJSP is presented by defining state representation of very short state features based on 2 state variables in simulation environment, action space composed of widely-used priority dispatching rules in literature & scheduling policy baesd on MLP with only 1 hidden layer. Various experiments are performed on classic benchmark instances with different sizes & results show: performance of PDR is improved in our environment, even better than some DRL methods. Besides, our DRL scheduling method provides competing scheduling performance compared with DRL, meta-heuristic & PDR methods.

  – Bài báo này đề xuất 1 môi trường DRL dựa trên mô phỏng sự kiện rời rạc theo trình tự thời gian cho FJSP. Quá trình lập lịch được mô tả bằng 1 thuật toán mô phỏng, trong đó các thay đổi trạng thái được ghi lại bởi các biến trạng thái & hàm thưởng được tính toán dựa trên diện tích lập lịch của máy tại mỗi bước quyết định. Trong môi trường mô phỏng này, 1 khuôn khổ DRL đầu cuối được đề xuất dựa trên PPO tác nhân-phê bình, cung cấp 1 cơ sở hạ tầng linh hoạt cho việc thiết kế biểu diễn trạng thái, không gian hành động & mạng lưới chính sách lập lịch. Bên cạnh đó, 1 mô hình DRL đơn giản cho FJSP được trình bày bằng cách định nghĩa biểu diễn trạng thái của các đặc trưng trạng thái rất ngắn dựa trên 2 biến trạng thái trong môi trường mô phỏng, không gian hành động bao gồm các quy tắc phân bổ ưu tiên được sử dụng rộng rãi trong tài liệu & chính sách lập lịch dựa trên MLP với chỉ 1 lớp ẩn. Nhiều thử nghiệm khác nhau được thực hiện trên các phiên bản chuẩn cổ điển với các kích thước khác nhau & kết quả cho thấy: hiệu suất của PDR được cải thiện trong môi trường của chúng tôi, thậm chí còn tốt hơn 1 số phương pháp DRL. Hơn nữa, phương pháp lập lịch DRL của chúng tôi cung cấp hiệu suất lập lịch cạnh tranh so với các phương pháp DRL, meta-heuristic & PDR.

  Future research will mainly focus on design of scheduling policy networks as well as state representation. State features can be thought as texts or images so that various networks in NLP & CV fields can be used in scheduling policy networks, e.g. SPP networks, TextCNN, & Transformer.

  – Nghiên cứu trong tương lai sẽ chủ yếu tập trung vào thiết kế mạng lưới chính sách lập lịch cũng như biểu diễn trạng thái. Các đặc điểm trạng thái có thể được hiểu dưới dạng văn bản hoặc hình ảnh, do đó, các mạng lưới khác nhau trong trường NLP & CV có thể được sử dụng trong mạng lưới chính sách lập lịch, ví dụ: mạng SPP, TextCNN, & Transformer.

3 Miscellaneous