The Eastasouth Journal of Information System and Computer Science
Vol. 1, No. 03, April 2024, pp. 277-290
ISSN: 3025-566X, DOI: 10.58812/esiscs.v1i03

Deep Reinforcement Learning-Driven Engineering Change Sequencing for Complex Product Development Programs

Yogesh Gawade
Oakland University, Rochester, Michigan, USA

Article Info

Article history:

Received April, 2024
Revised April, 2024
Accepted April, 2024

Keywords:

Change Propagation;
Change Sequencing;
Combinatorial Optimization;
Deep Reinforcement Learning;
Design Structure Matrix;
Engineering Change Management;
Markov Decision Process;
Product Development;
Systematic Review

ABSTRACT

The order of the Engineering changes (ECs) that are acted upon in complex product development programs is tightly coupled to the ordering of the component, the parameters and the activities, and this ordering significantly affects rework, cost, and lead time. This systematic review combines 24 peer-reviewed papers from four research streams: design structure matrix (DSM) and process architecture, engineering change management and propagation, deep reinforcement learning (DRL) foundations and DRL-based production scheduling. The evidence is used to build EC sequencing as a Markov decision process with the propagation indices generated from DSM as the state, change or sequencing rules as the action, and penalties for cost growth, schedule growth and triggered rework as the reward. The synthesis revealed that changes require between 1/3 and 1/2 of engineering time, 48.2% of the 41,500 change requests that were raised in an 8-year period for a sensor program were implemented, and changes were propagated no further than three engineering interfaces. In learning, a deep Q-network solved 43 of 49 benchmark games while previous methods were unable to solve the same number of games. A search-augmented network architecture beat other algorithms in 99.8% of games in the benchmark. The results show that rule-selecting deep Q-network agents achieve the best trade-off between scalability and verifiability, while search-based planners have a much greater look-ahead capability but at a significantly higher computational cost. The simulator fidelity, limited change histories, explainability and certification traceability are key challenges. A reference architecture and a research agenda are suggested for the development of the transfer of DRL from production scheduling to engineering change control.

This is an open access article under the CC BY-SA license.

Corresponding Author:

Name: Yogesh Gawade
Institution: Oakland University, Rochester, Michigan, USA
Email: yogeshgawade@oakland.edu

1. INTRODUCTION

Complex product development programs like aircraft, engines and large sensor systems contain thousands of coupled components, parameters and activities, and changes to one of these parts often trickle down to other parts. Since the order of processing pending engineering changes (ECs) can affect the amount of rework, cost, and lead time, it is not surprising that processing the order is determined mostly by

expert judgement and static priority rules. Deep reinforcement learning (DRL) refers to the fusion of reinforcement learning (RL) with deep neural networks that can be used to function as value function approximators, policy function approximators, or actor-critics for learning sequential decision policies from interaction with complex environments.Deep reinforcement learning (DRL) refers to the combination of RL with deep neural network function approximators that can learn sequential decision policies directly from interaction with complex environments through value-based, policy-based and actor-critic algorithm families.

It can be stated as a combinatorial optimization (CO) problem, since the number of possible processing orders increases in a factorial manner with the number of open changes, a backlog of 10 open changes alone allows 3,628,800 different orders. To date, machine learning has been integrated in CO in three complementary paradigms: end-to-end learning of solutions, learning the configuration of optimization algorithms, and learning within exact or heuristic solvers [1]. These paradigms suggest that a learned sequencing policy can either serve as the replacement for the traditional heuristics, or lead to them if data is available or if the number of decisions that cannot be verified is acceptable.

Typically, the structural knowledge which is necessary for such a policy is represented by the design structure matrix (DSM), which is a square matrix in which the off-diagonal elements are used to record the dependencies between the components of the system. The preliminary review identified four types: component based, team based, activity based, and parameter based matrices and the first two were analysed primarily through the application of clustering and the latter two through the application of sequencing [2]. The aim of the present review is to combine the research efforts from the two fields of change propagation with DR scheduling, to formulate an EC sequencing as a Markov decision process (MDP), and to benchmark a number of candidate algorithm families against the published benchmarks and practical obstacles.

2. RESEARCH METHOD

A search of previously published peer-reviewed studies was conducted using a structured literature review protocol on a set corpus of 24 studies published from 2001 to 2023, as shown in Figure 1 and Table 1. The studies were categorized into four streams: (4) design structure and process architecture, (10) engineering change management and propagation, (5) DRL and learning based CO foundations, and (5) DRL in production scheduling. The coding logic followed the approach of the survey organization of the DSM extensions, that is, extensions are organized by domain, analysis type, and multi-domain matrices that connect domains [3]. Items of data collected from each study included the formalism of the model, the context of the cases, quantitative results of the study, and limitations.

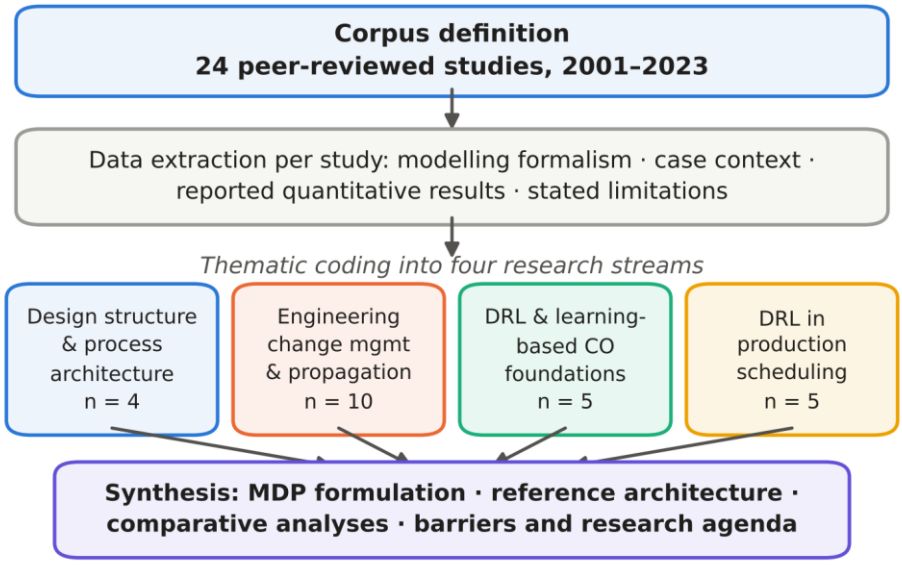


Figure 1. Systematic review protocol and coding of the 24-study corpus into four research streams
Source: Compiled by the authors from the reviewed literature

Table 1. Literature Map of the Reviewed Corpus

Ref.	Year	Stream	Core contribution	Role in EC sequencing
[4]	2017	DRL foundations	Survey of value-based, policy-based, and actor-critic DRL	Algorithm families for the agent
[1]	2021	DRL foundations	Three paradigms of machine learning for CO	Positioning of learned sequencing policies
[2]	2001	DSM / process	Four DSM types; clustering and sequencing	Structural representation of changes
[3]	2016	DSM / process	Survey of DSM extensions and multi-domain matrices	Heterogeneous state design
[5]	2002	DSM / process	Simulation of process architecture, cost and schedule risk	Order as a risk-controlling variable
[6]	2004	ECM	Change prediction method with likelihood and impact DSMs	Combined-risk state features
[7]	2004	ECM	Initiated vs. emergent changes; multipliers and absorbers	Prioritization of high-leverage elements
[8]	2009	ECM	41,500 change requests; normalized CPI	Data-driven propagation indices
[9]	2012	ECM	Multidomain function-behaviour-structure model	Multi-domain state graph
[10]	2013	ECM	Holistic categorization of ECM literature	Locating sequencing in the in-change stage
[11]	2011	ECM	Overview of EC definitions, costs, and processes	Economic justification
[12]	2023	DRL scheduling	Review of RL in machine scheduling	Transferable problem classes and objectives
[13]	2012	ECM	Assessment of change propagation effects on requirements	Reward definition
[14]	2020	DRL scheduling	DQN selecting composite rules in dynamic job shop	Rule-selecting action design
[15]	2021	DRL foundations	Survey of RL for CO	Constructive and graph-based policies

Ref.	Year	Stream	Core contribution	Role in EC sequencing
[16]	2015	DRL foundations	Deep Q-network with replay and target network	Value-based learning backbone
[17]	2012	ECM	Higher-order DSMs for requirement change	Indirect reachability features
[18]	2022	DRL scheduling	Systematic review of DRL in production systems	Evidence on industrial maturity
[19]	2012	ECM	Multilayer product-change-social network	Planning horizon and organizational layer
[20]	2003	ECM	Parameter-based change management in concurrent engineering	Action granularity
[21]	2016	DRL foundations	Policy and value networks with tree search	Look-ahead for critical changes
[22]	2021	DRL scheduling	Review of RL-based scheduling optimization	Hybrid RL and meta-heuristics
[23]	2018	DRL scheduling	Cooperative DQN agents in semiconductor production	Deployment and acceptance evidence
[24]	2003	DSM / process	DSM for iteration and overlapping in concurrent engineering	Static baseline for adaptive ordering

Source: Compiled by the authors from the reviewed literature

3. LITERATURE SYNTHESIS

3.1 Process Architecture and Rework Dynamics

Process architecture, which involves how activities are arranged and the information they pass between each other, was demonstrated to influence cost and schedule risk in the product development process. The simulation model developed for the preliminary design of an uninhabited combat aerial vehicle assigned uncertain durations and costs to each activity, an improvement curve for each activity, and a probability of rework due to changes in inputs for each activity; alternative architectures were then repeated simulated to produce cost and schedule distributions [5]. The basic idea is that the same set of activities performed in different sequences has different materially different risk profiles, thus making the sequence a decision variable, and the DSM types underlying such models are summarized in Table 2.

Table 2. DSM Types and Their Use in Engineering Change Sequencing

DSM type	Elements represented	Typical analysis	Use in EC sequencing
Component-based	Product components and interfaces	Clustering into modules	Identifying modules that confine propagation
Team-based	Teams and communication links	Clustering	Assigning change work to coordinated teams
Activity-based	Activities and information flows	Sequencing, partitioning, tearing	Ordering implementation tasks to minimise feedback
Parameter-based	Design parameters and dependencies	Sequencing	Fine-grained ordering of parameter updates
Multi-domain matrix	Several domains linked by mapping matrices	Combined cross-domain analysis	Heterogeneous state representation for DRL

Source: Synthesized by the authors from the reviewed literature

3.2 *Change Prediction and Propagation*

The change prediction method (CPM) was used for quantifying propagation, and it was the combination of direct likelihood and impact DSMs obtained from engineers, combined with the aggregation of indirect paths into a combined risk between two subsystems, that were demonstrated in helicopter design at an industrial partner [6]. Each path p is associated with a risk RP, where the path likelihood Lp is the product of the direct likelihoods luv on path p and parallel paths in the set Pab are combined by the complement-product rule of Equations (1) and (2).

$$R_{ab} = 1 - \prod_{p \in P_{ab}} (1 - L_p I_p) \quad (1)$$

$$L_p = \prod_{(u,v) \in p} l_{uv} \quad (2)$$

Empirical research in complex engineering organisations, such as a helicopter manufacturer, identified initiated changes as design changes that came out of new requirements, customer requests and certification, and emergent changes as changes that came out of problems found within the design itself [7]. Components were further classified according to their behaviour as constants which receive no change; absorbers that absorb more change; carriers that pass on the change in similar measure; and multipliers that produce more change than received. High-leverage elements, changes that reach them early can cause cascades, and informed order can contain these changes, are called multipliers for sequencing.

A study of 41,500 change requests raised over 8 years when developing a complex sensor system, mapped to 46 subsystem areas [8] provided the largest empirical evidence in the corpus. Only 48.2 % of the changes were implemented, the activity related to cyclical changes was highest in the latter part of the program, in the phase of integration and functional testing, and a normalized change propagation index (CPI) was defined to position each area within a range between −1 and +1 (see Equation (3), in which eij represents a recorded propagation link from area i to area j). These indices can be calculated using past data and be used as state features as shown in Figure 2 and Table 3.

$$\mathrm{CPI}_i = \frac{\sum_j e_{ij} - \sum_j e_{ji}}{\sum_j e_{ij} + \sum_j e_{ji}} \quad (3)$$

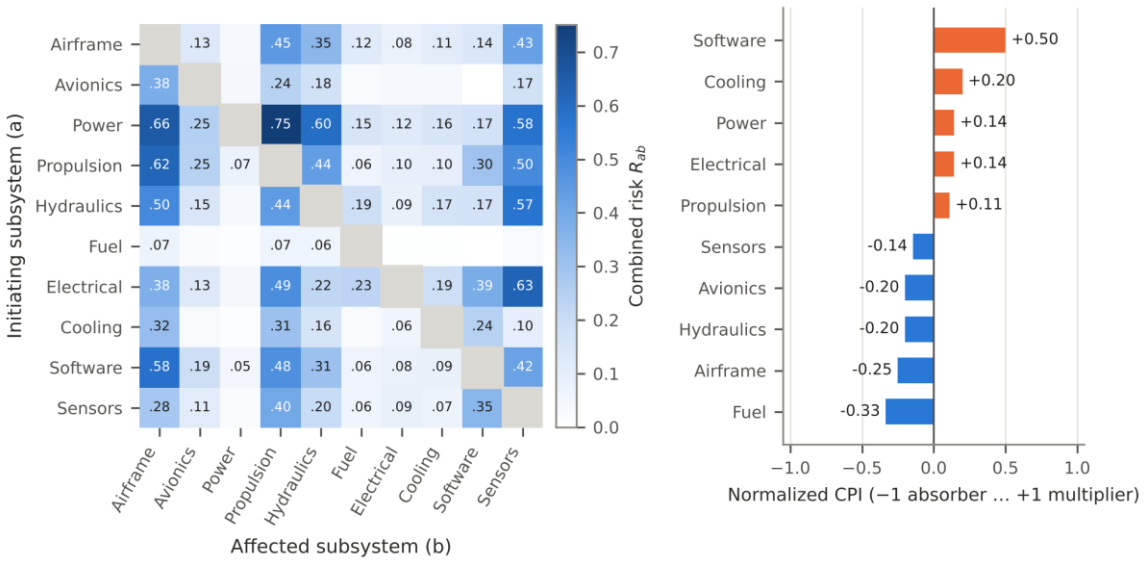


Figure 2. Illustrative combined-risk DSM (left) and normalized CPI (right) computed with Equations (1)-(3) for a hypothetical ten-subsystem product

Source: Illustrative computation by the authors using randomly generated likelihood and impact values

Table 3. Quantitative Descriptors of Engineering Change Propagation Reported in the Corpus

Descriptor	Reported value	Context	Ref.
Engineering capacity consumed by ECs	One third to one half	Reported industrial estimates	[11]
Share of tooling cost attributable to ECs	20–50%	Reported industrial estimates	[11]
Change requests analysed	41,500	Sensor system, 8-year program	[8]
Subsystem areas mapped	46	Same program	[8]
Proposed changes implemented	48.2%	Same program	[8]
Normalized CPI range	−1 (absorber) to +1 (multiplier)	Area-level index	[8]
Interfaces spanned by parent-child propagation	At most 3	Multilayer analysis, same program	[19]
Generations of descendant changes	At most 5	Multilayer analysis, same program	[19]
Component behaviours under change	4 (constant, absorber, carrier, multiplier)	Complex engineering products	[7]
Domains in multidomain model	3 (function, behaviour, structure)	Multidomain propagation model	[9]
Layers in multilayer model	3 (product, change, social)	Multilayer network model	[19]

Source: Values as reported in the cited studies

Single domain models were extended by the multidomain change propagation model that relates function, behaviour and structure, and propagation through the physical components are followed back to the functions and behaviours that cause it [9]. The model was designed to assist with uncertainty reduction and risk management, and illustrated that propagation paths mediated through common behaviours can be underestimated using component level matrices. This result is applicable for the heterogeneous state representation, in which nodes of multiple domains co-exist, for DRL based sequencing instead of a single component adjacency matrix.

3.3 Change Management Frameworks and Economic Significance

A comprehensive classification of ECM literature grouped the contributions by the various phases of the change process: prevention and front loading of change; effective and efficient processing of the change; learning from previous changes, [10]. In that context, sequencing decisions happen in the in-change stage and it is where the process support and the decision tools lie. As can be seen from Figure 3, propagation studies focus on the period between 2001 and 2013, while DRL scheduling studies are only found from 2018 onward, which is why the two lines of work have only infrequently been linked.

The economic argument for improved sequencing is based on the size of the effort for change. The engineering change literature reviewed reports that in some firms, the use of EC can account for one third to one half of engineering capacity, and that the total cost of EC can be 20-50% of tooling costs [11]. The same overview also established that a change to parts, drawings, or software once released would result in a significant cost increase, and that this increased cost would increase with the later stages of a product's life cycle, making the penalty for an unfavorable processing order later in a program much greater.

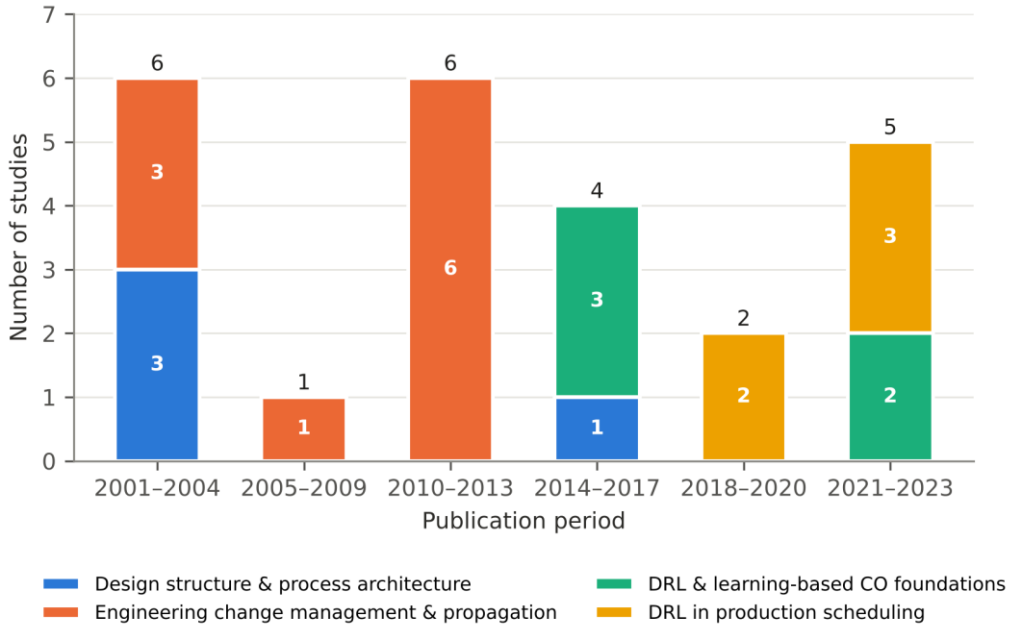


Figure 3. Distribution of the reviewed studies by publication period and research stream
Source: Compiled by the authors from the reviewed literature

3.4 Reinforcement Learning in Scheduling

Machine scheduling is the most immediate operational counterpart of EC sequencing, since both consist of optimizing tasks with precedence constraints in the presence of unpredictable disruptions. In a comprehensive review of RL applications to machine scheduling, the most common variants of Q-learning and DQN were found to be widely used, along with single machine and parallel machine settings, flow shop and job shop settings, and a number of objectives, including makespan and tardiness [12]. The review also reported that there was a heavy dependence on simulated benchmark instances, and that while there is some degree of generalization to industrial settings, the same applies to engineering change environments.

4. DRL FRAMEWORK FOR ENGINEERING CHANGE SEQUENCING

4.1 Formulation as a Markov Decision Process

The reward of a sequencing agent should be related to the consequences of propagation, and these consequences were operationalized to assess the impact of propagation by mapping the predicted changes of the components to the product requirements, in order to be able to compare different propagation options based on the impact on fulfilling the product requirements [13]. Based on this rationale, EC sequencing is formalized in this section as an MDP (S, A, P, R, γ) with the agent optimizing the discounted return R in Equation (4), where $\gamma \in [0, 1)$, and the cost growth ΔC, schedule growth ΔT and the number of triggered rework instances N are penalized by α, β, and κ, respectively.

$$G_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \quad (4)$$

$$r_t = -(\alpha \Delta C_t + \beta \Delta T_t + \kappa N_t^{\mathrm{rw}}) \quad (5)$$

State and action design can be transferred from a dynamic flexible job shop scheduling, where the deep Q-network is learned with double Q-learning and soft target update, and the target is based on 7 generic state features observed by the network, such as average machine utilisation and estimated tardy machine rates etc in previous steps, and 6 composite dispatching rules are to be selected for

each rescheduling step after new jobs are inserted [14]. The agent performed better than every single rule, as well as other benchmark approaches (with respect to total tardiness), for various production scenarios. The analogous design for EC sequencing, summarized in Table 4, is of the normalized propagation features being the state and a small library of sequencing rules being the action set, with the action space remaining independent of the size of the backlog.

Table 4. MDP Formulation of Engineering Change Sequencing

MDP element	Definition in EC sequencing	Example features or values	Derived from
State s_t	Open changes, propagation structure, resource status	Normalized CPI, combined risk R_{ab}, reachability $H^{(K)}$, due-date slack, queue length	[6], [8], [14], [17]
Action a_t	Next change or sequencing rule	Library of 6 rules, e.g., highest-risk-first, multiplier-first, earliest-due-date	[14], [15]
Transition P	Propagation to neighbours and arrival of new changes	Stochastic, driven by likelihood DSM	[5], [6]
Reward r_t	Penalty on cost, schedule, and rework growth	$-(\alpha\Delta C + \beta\Delta T + \kappa N)$, Equation (5)	[5], [13]
Discount γ	Weight of future consequences	0.99 in the DQN benchmark	[16]
Episode	Backlog cleared or program milestone reached	Bounded by ≤ 5 propagation generations	[19]

Source: Synthesized by the authors from the reviewed literature

When instead individual changes must be expressed, architectures provided by the survey of RL for CO are appropriate such as pointer networks, attention-based encoders, or graph neural networks that build the solution one element at a time and are trained by policy gradient or value-based approaches [15]. A graph representation of open changes and their DSM dependencies enables that one training policy to be applied to program(s) that may have different numbers of components, thus taking care of the variable size of inputs typical of product development.

4.2 *Learning Algorithms and State Representation*

The experience replay memory used 1 million transitions, periodically updated target network [16] and the deep Q-network (DQN) was trained using 49 Atari 2600 games with 84 × 84 pixel inputs that were stacked over 4 frames. The network won more games than the best previous RL approaches and surpassed 75% of the score of a professional human tester on 29 of the games. It has a learning target and its loss are defined in Equations (6) and (7), where θ^- represents the target network parameters and D represents the replay memory.

$$y = r + \gamma \max_{a'} Q(s', a'; \theta^-) \quad (6)$$

$$L(\theta) = \mathrm{E}_D\left[\left(y - Q(s, a; \theta)\right)^2\right] \quad (7)$$

Indirect propagation is built in the state by higher order DSMs; these have been used in an industry case study to calculate the propagation of a requirement change by putting the requirement dependency matrix to successive powers, with the n-th power of an entry on the matrix representing connections of length n between two requirements [17]. The resulting reachability feature is given in Equation (8), where D is the binary DSM and K is the maximum propagation depth that is considered. The whole architecture, where the input of the algorithm is derived from the DSM, the output of the encoder is fed into the agent, is depicted in Figure 4, and the candidate algorithm families are contrasted in Table 5.

$$H^{(K)} = \sum_{n=1}^{K} D^n \quad (8)$$

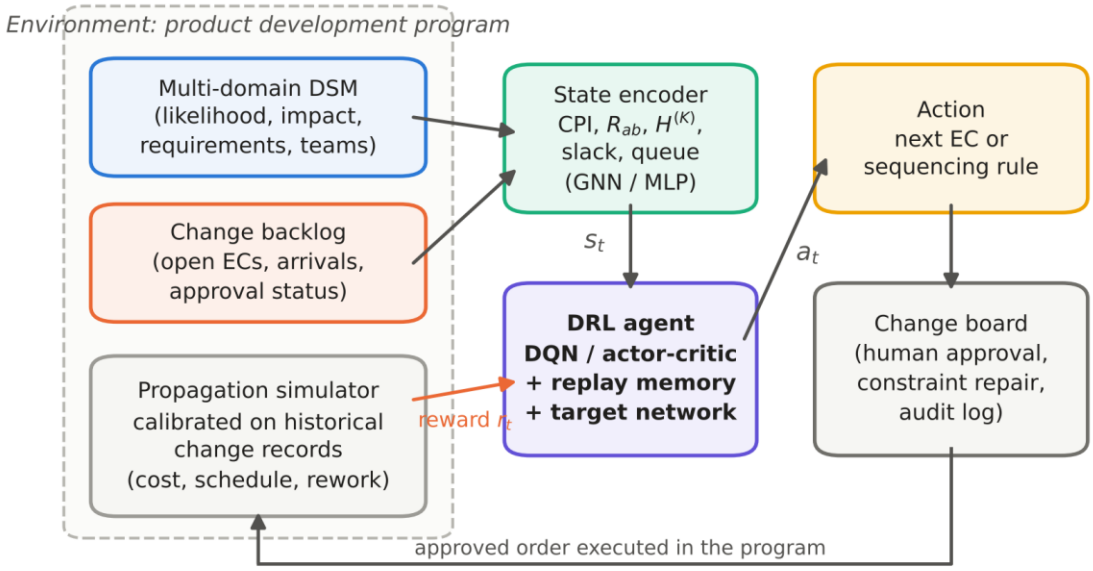


Figure 4. Reference architecture for DRL-driven engineering change sequencing
Source: Compiled by the authors from the reviewed literature

Table 5. Comparison of Candidate DRL Algorithm Families for Engineering Change Sequencing

Family	Representative design	Action space	Training stability	Scalability	Best use
Value-based	DQN, double DQN	Small, discrete	Improved by replay and target network	High	Rule selection
Policy gradient / actor-critic	Asynchronous and trust-region methods	Discrete or continuous	Moderate	High with parallel actors	Constructive policies
Constructive neural CO	Pointer, attention, graph networks	Permutations	Moderate	Variable-size inputs	Direct change ordering
Search-augmented	Policy and value networks with MCTS	Large, discrete	High	Low (1,202 CPUs, 176 GPUs)	High-stakes critical changes
Hybrid RL and meta-heuristic	RL-guided operators or parameters	Operators or parameters	High	Moderate	Constraint-heavy change boards

Source: Synthesized by the authors from the reviewed literature

5. COMPARATIVE ANALYSIS AND DISCUSSION

5.1 *Performance Evidence and Benchmarks*

Based on a systematic literature review of the use of DRL in production systems, the authors identified the most popular applications, namely production scheduling, process control, robotics, and maintenance, where algorithms from the DQN family were most widely used and best-evaluated in simulation and not in real production systems [18]. The headline results of the foundational algorithms in Figure 5 reflect the best that can be achieved, which is what motivates transfer to engineering decisions, but the lack of industrial validation in the production research signals a gap that

we are likely to face in the EC sequencing applications.

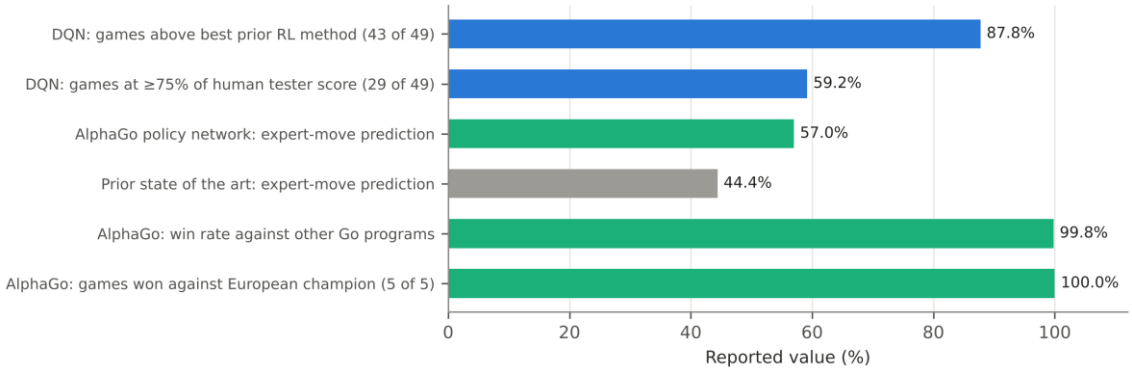


Figure 5. Headline performance metrics reported for foundational DRL systems
Source: Compiled by the authors from the reviewed literature

An added comparison axis is organizational coupling. The multi-layer model of change propagation included a product layer, a change layer and a social layer with engineers, and in the case of the large sensor program, linked the organizational role of engineers to the propagation effects their jobs had. [19] It was reported that propagation from a parent-child occurred at most three system interfaces and that it terminated after within five generations of descendant changes, this gives an upper bound to the useful depth K in Equation (8) for propagation and thus short planning horizons for a sequencing agent.

A second dimension is granularity of the sequencing decision. One parameter based approach to change management in concurrent engineering indicated that changes on parameter level, instead of changes in documents and entire components, enabled selective notification of changes to dependent teams and partner companies, and coordinating the approvals [20]. A parameter-level action space is more fine-grained, and may require less rework, but increases the size of the state space; depending on the granularity, this has a cost/benefit relationship, as shown by the qualitative scores in Figure 6.

Look-ahead planning shows to be the most promising approach for combining learned evaluation and search. The combined policy and value networks with Monte Carlo tree search (MCTS) Go program was in a game with a breadth of approximately 250 moves and a depth of approximately 150 moves, it predicted expert moves with 57.0% accuracy compared with 44.4% for the earlier state of the art Go program, it played 5 games to 0 against the European champion, and it defeated the European champion by 5 games to 0 [21]. In distributed version, 1,202 CPU and 176 GPU were used which was a scalability cost compared to light dispatching agent in Table 6.

Table 6. Comparison of Learning-Based Decision Studies Relevant to Engineering Change Sequencing

Study	Problem	Method	State / action	Reported outcome	Transfer to EC
[14]	Dynamic flexible job shop with new job insertions	DQN, double Q-learning, soft target updates	7 features / 6 composite rules	Better total tardiness than each rule	Rule-selecting change agent

Study	Problem	Method	State / action	Reported outcome	Transfer to EC
[16]	49 Atari 2600 games	DQN with replay and target network	84 × 84 × 4 pixels / joystick actions	43 of 49 above earlier methods	Value-based backbone
[21]	Game of Go	Policy and value networks with MCTS	Board position / legal moves	99.8% win rate; 5–0 match	Look-ahead for critical changes
[23]	Semiconductor front-end production	Cooperative multi-agent DQN	Work-center status / dispatch choices	Comparable to expert heuristics	Multi-team change boards

Source: Values as reported in the cited studies

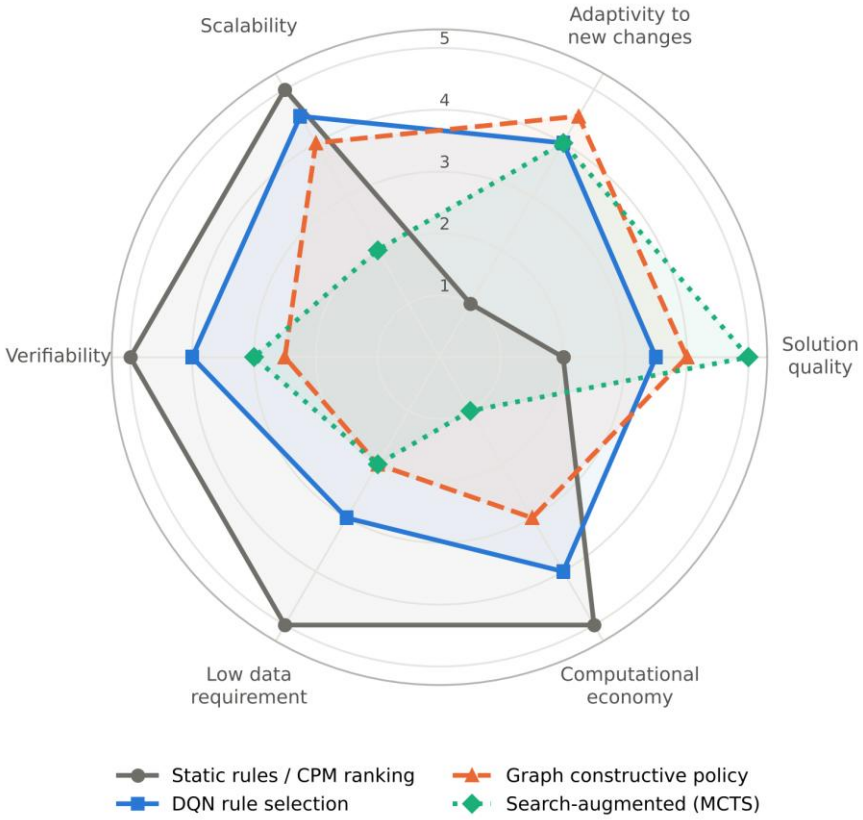


Figure 6. Qualitative comparison of sequencing approach families (scores 1–5 assigned in this review)

Source: Compiled by the authors from the reviewed literature

5.2 *Hybridization, Deployment, and Governance*

As one can see, several approaches have been proposed in the literature [22] based on RL for intelligent optimization of manufacturing scheduling, including approaches in which RL selects dispatching rules, approaches in which RL builds schedules directly, and approaches in which RL tunes the parameters or operators of meta-heuristic algorithms. The engineering change board concept is particularly relevant for the hybrid approach, as a hard precedence and approval constraint can be enforced by a meta-heuristic or exact repair routine, while priorities are provided by the learned component which maintains the verifiability of the final order.

Cooperative DQN agents, capable of controlling a single work center, were trained to perform a simulation of a semiconductor

production environment and successfully achieved performance levels equal to the performance of the dispatching heuristics designed by experts without the need for any rules [23]. The same work revealed factors such as reliance on a faithful simulator, length of training and the requirement for the operator to accept "black box" decisions (as outlined in Table 7). These barriers gain ethical and regulatory significance when considering EC sequencing, as it involves making changes that can be tracked back to accountable engineers and the certification requirements.

Table 7. Barriers to DRL-Driven Engineering Change Sequencing and Mitigation Strategies

Barrier	Manifestation	Mitigation	Evidence
Simulator fidelity	Policy learned on an inaccurate propagation model	Calibrate likelihood DSMs on historical change records	[8], [23]
Data scarcity	Few programs with complete change histories	Graph policies reusable across programs	[15], [18]
Reward misspecification	Cost minimized at the expense of requirements	Requirement-linked effect assessment in the reward	[13]
Explainability and acceptance	Engineers reject opaque orders	Rule-selecting actions and ranked recommendations	[14], [23]
Constraint violation	Orders that conflict with approvals or precedence	Hybrid RL with constraint repair	[22]
Computational cost	Search-based planning is expensive	Bound look-ahead to observed propagation depth	[19], [21]
Ethical and regulatory traceability	Certification requires accountable decisions	Human approval, audit logs, documented rationale	[10], [11]

Source: Synthesized by the authors from the reviewed literature

Lastly, the evolution of the field is traced in Figure 7 and leads back to concurrent engineering, as the DSM was applied to analysing information flow and overlapping of activities in a complex project, and coupled activities are either iterated or carefully overlapped [24]. These static partitioning techniques can be viewed as the static predecessors of a DRL sequencer, in which an order is computed once and then applied to every subsequent change, and learned estimates of propagation and rework are used to determine an order.

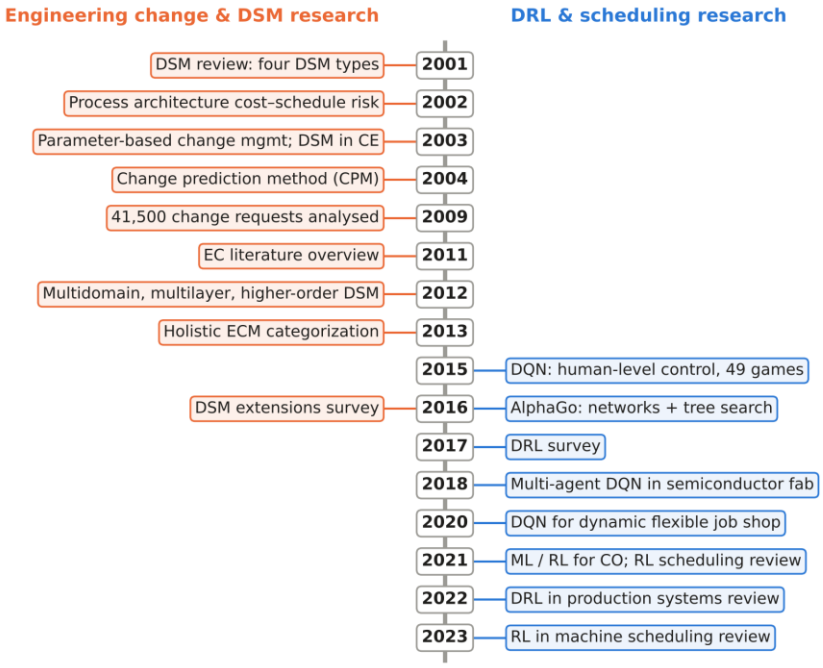


Figure 7. Evolution of engineering change, DSM, and DRL scheduling research represented in the corpus

Source: Compiled by the authors from the reviewed literature

6. CONCLUSION

This review brings together evidence from engineering change management and deep reinforcement learning into a single perspective for EC sequencing in complex product development programs. Structural representation research, which provides likelihood and impact DSMs, multidomain linkages, normalized propagation indices and higher order reachability, is provided by change propagation research; and scheduling research provides a suite of tested state, action, reward designs, as well as evidence that learned policies can outperform or match expert dispatching rules. Based on the resulting MDP formulation and reference architecture, we conclude that rule-selecting DQN agents are the most readily deployable approach, graph-based constructive policies are needed when individual changes have to be ordered across programs of varying sizes, and search-augmented planners are warranted only if the benefits of look-ahead outweigh the computational cost. The actual application of the system is still dependent on simulators that are calibrated using historical change records, clear reward definitions, and humans' approval of all the learned recommendations. Future works should focus on open benchmark data sets of change history, hybrid learning and optimization approaches which ensure constraint satisfaction, and field test to measure the reduction in rework and lead time in operating change boards.

REFERENCES

[1] Y. Bengio, A. Lodi, and A. Prouvost, "Machine learning for combinatorial optimization: A methodological tour d'horizon," *Eur. J. Oper. Res.*, vol. 290, no. 2, pp. 405–421, 2021, doi: https://doi.org/10.1016/j.ejor.2020.07.063.

[2] T. R. Browning, "Applying the design structure matrix to system decomposition and integration problems: a review and new directions," *IEEE Trans. Eng. Manag.*, vol. 48, no. 3, pp. 292–306, 2001, doi: 10.1109/17.946528.

[3] T. R. Browning, "Design Structure Matrix Extensions and Innovations: A Survey and New Opportunities," *IEEE Trans. Eng. Manag.*, vol. 63, no. 1, pp. 27–52, 2016, doi: 10.1109/TEM.2015.2491283.

[4] K. Arulkumaran, M. Deisenroth, M. Brundage, and A. Bharath, "Deep Reinforcement Learning: A Brief Survey,"

IEEE Signal Process. Mag., vol. 34, no. 6, pp. 26–38, Aug. 2017, doi: 10.1109/MSP.2017.2743240.

[5] T. R. Browning and S. D. Eppinger, "Modeling impacts of process architecture on cost and schedule risk in product development," *IEEE Trans. Eng. Manag.*, vol. 49, no. 4, pp. 428–442, 2002, doi: 10.1109/TEM.2002.806709.

[6] P. J. Clarkson, C. Simons, and C. Eckert, "Predicting Change Propagation in Complex Design ," *J. Mech. Des.*, vol. 126, no. 5, pp. 788–797, Oct. 2004, doi: 10.1115/1.1765117.

[7] C. Eckert, P. J. Clarkson, and W. Zanker, "Change and customisation in complex engineering domains," *Res. Eng. Des.*, vol. 15, no. 1, pp. 1–21, 2004, doi: 10.1007/s00163-003-0031-7.

[8] M. Giffin, O. de Weck, G. Bounova, R. Keller, C. Eckert, and P. J. Clarkson, "Change Propagation Analysis in Complex Technical Systems," *J. Mech. Des.*, vol. 131, no. 8, Jul. 2009, doi: 10.1115/1.3149847.

[9] B. Hamraz, N. H. M. Caldwell, and P. John Clarkson, "A Multidomain Engineering Change Propagation Model to Support Uncertainty Reduction and Risk Management in Design," *J. Mech. Des.*, vol. 134, no. 10, Sep. 2012, doi: 10.1115/1.4007397.

[10] B. Hamraz, N. H. M. Caldwell, and P. J. Clarkson, "A Holistic Categorization Framework for Literature on Engineering Change Management," *Syst. Eng.*, vol. 16, no. 4, pp. 473–505, Dec. 2013, doi: https://doi.org/10.1002/sys.21244.

[11] T. A. W. Jarratt, C. M. Eckert, N. H. M. Caldwell, and P. J. Clarkson, "Engineering change: an overview and perspective on the literature," *Res. Eng. Des.*, vol. 22, no. 2, pp. 103–124, 2011, doi: 10.1007/s00163-010-0097-y.

[12] B. M. Kayhan and G. Yildiz, "Reinforcement learning applications to machine scheduling problems: a comprehensive literature review," *J. Intell. Manuf.*, vol. 34, no. 3, pp. 905–929, 2023, doi: 10.1007/s10845-021-01847-3.

[13] E. C. Y. Koh, N. H. M. Caldwell, and P. J. Clarkson, "A method to assess the effects of engineering change propagation," *Res. Eng. Des.*, vol. 23, no. 4, pp. 329–351, 2012, doi: 10.1007/s00163-012-0131-3.

[14] S. Luo, "Dynamic scheduling for flexible job shop with new job insertions by deep reinforcement learning," *Appl. Soft Comput.*, vol. 91, p. 106208, 2020, doi: https://doi.org/10.1016/j.asoc.2020.106208.

[15] N. Mazyavkina, S. Sviridov, S. Ivanov, and E. Burnaev, "Reinforcement learning for combinatorial optimization: A survey," *Comput. Oper. Res.*, vol. 134, p. 105400, 2021, doi: https://doi.org/10.1016/j.cor.2021.105400.

[16] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015, doi: 10.1038/nature14236.

[17] B. Morkos, P. Shankar, and J. D. Summers, "Predicting requirement change propagation, using higher order design structure matrices: an industry case study," *J. Eng. Des.*, vol. 23, no. 12, pp. 905–926, Dec. 2012, doi: 10.1080/09544828.2012.662273.

[18] M. Panzer and B. Bender, "Deep reinforcement learning in production systems: a systematic literature review," *Int. J. Prod. Res.*, vol. 60, no. 13, pp. 4316–4341, Jul. 2022, doi: 10.1080/00207543.2021.1973138.

[19] M. C. Pasqual and O. L. de Weck, "Multilayer network model for analysis and management of change propagation," *Res. Eng. Des.*, vol. 23, no. 4, pp. 305–328, 2012, doi: 10.1007/s00163-011-0125-6.

[20] K. Rouibah and K. R. Caskey, "Change management in concurrent engineering from a parameter perspective," *Comput. Ind.*, vol. 50, no. 1, pp. 15–34, 2003, doi: https://doi.org/10.1016/S0166-3615(02)00138-0.

[21] D. Silver *et al.*, "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016, doi: 10.1038/nature16961.

[22] L. Wang, Z. Pan, and J. Wang, "A Review of Reinforcement Learning Based Intelligent Optimization for Manufacturing Scheduling," *Complex Syst. Model. Simul.*, vol. 1, no. 4, pp. 257–270, 2021, doi: 10.23919/CSMS.2021.0027.

[23] B. Waschneck *et al.*, "Optimization of global production scheduling with deep reinforcement learning," *Procedia CIRP*, vol. 72, pp. 1264–1269, 2018, doi: https://doi.org/10.1016/j.procir.2018.03.212.

[24] A. Yassine and D. Braha, "Complex Concurrent Engineering and the Design Structure Matrix Method," *Concurr. Eng.*, vol. 11, no. 3, pp. 165–176, Sep. 2003, doi: 10.1177/106329303034503.