ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL

MULTI-AGENT PLANNING WITH AUTOMATED CURRICULUM LEARNING

Ph.D. THESIS

Onur AKGÜN

Department of Mechatronics Engineering

Mechatronics Engineering Programme

JUNE 2025

ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL

MULTI-AGENT PLANNING WITH AUTOMATED CURRICULUM LEARNING

Ph.D. THESIS

Onur AKGÜN
(518182018)

Department of Mechatronics Engineering

Mechatronics Engineering Programme

Thesis Advisor: Assoc. Prof. Dr. Nazım Kemal ÜRE

JUNE 2025

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

OTOMATİK MÜFREDAT ÖĞRENMESİ İLE ÇOKLU AJAN PLANLAMASI

DOKTORA TEZİ

Onur AKGÜN
(518182018)

Mekatronik Mühendisliği Anabilim Dalı

Mekatronik Mühendisliği Programı

Tez Danışmanı: Doç. Dr. Nazım Kemal ÜRE

HAZİRAN 2025

Onur AKGÜN, a Ph.D. student of ITU Graduate School student ID 518182018 successfully defended the thesis entitled “MULTI-AGENT PLANNING WITH AUTOMATED CURRICULUM LEARNING”, which he/she prepared after fulfilling the requirements specified in the associated legislations, before the jury whose signatures are below.

Thesis Advisor : **Assoc. Prof. Dr. Nazım Kemal ÜRE** ....
Istanbul Technical University

Jury Members : **Prof. Dr. Metin GÖKAŞAN** ....
Istanbul Technical University

Assoc. Prof. Dr. Tuba ÇONKA YILDIZ ....
Fraunhofer- Institut für Integrierte Schaltungen IIS

Prof. Dr. Behçet Uğur TÖREYİN ....
Istanbul Technical University

Prof. Dr. Ahmet YILDIZ ....
Macromedia University of Applied Sciences

Date of Submission : **28 April 2025**
Date of Defense : **11 June 2025**

To my family,

FOREWORD

Pursuing a PhD has been one of the most challenging and rewarding experiences of my life. This dissertation reflects not only years of academic effort but also significant personal growth and perseverance.

I would first like to express my sincere appreciation to my supervisor, Assoc. Prof. Dr. Nazım Kemal ÜRE, for his guidance, support, and insightful feedback, which have been invaluable in shaping this work.

I am also grateful to my colleagues and friends who have shared in both the struggles and joys of this journey. Their support has meant a great deal to me.

I am deeply grateful to my parents, whose encouragement and belief in me have guided me through every stage of my academic path.

Finally, I would like to thank my wife, Gülen BÜYÜKOLCA AKGÜN, for her endless support, patience, and love throughout this journey. Her presence has been a constant source of strength.

June 2025

Onur AKGÜN
(M.Sc.)

TABLE OF CONTENTS

Page

FOREWORD .... **ix**
TABLE OF CONTENTS .... **xi**
ABBREVIATIONS .... **xv**
SYMBOLS .... **xvii**
LIST OF TABLES .... **xix**
LIST OF FIGURES .... **xxiii**
SUMMARY .... **xxv**
ÖZET .... **xxix**
1. INTRODUCTION **1**
  1.1 Background 1
  1.2 Purpose of Thesis 3
  1.3 Research Objectives 3
  1.4 Research Questions 4
  1.5 Significance and Potential Applications 4
2. LITERATURE REVIEW **7**
  2.1 Reinforcement Learning Fundamentals 7
  2.2 Bayesian Networks 8
  2.3 Curriculum Learning 9
    2.3.1 Teacher-student interaction methods 10
    2.3.2 Intermediate goal creation 10
    2.3.3 Self-play approaches 11
    2.3.4 Exploration-enhanced approaches 11
    2.3.5 Recent advances in structured progression 12
  2.4 Comparative Analysis and Research Positioning 12
3. BACKGROUND **15**
  3.1 Bayesian Networks 15
    3.1.1 Formal definition 15
    3.1.2 Key concepts and properties 17
      3.1.2.1 Conditional independence and the markov property 17
      3.1.2.2 Factorization of the joint probability distribution 17
    3.1.3 Inference and learning 18
      3.1.3.1 Probabilistic inference 18
      3.1.3.2 Learning bayesian networks 19
    3.1.4 Advantages and use 20
  3.2 Reinforcement Learning 21
    3.2.1 Core concepts 21
    3.2.2 Markov decision processes (MDPs) 21
    3.2.3 Policies and value functions 22
    3.2.4 Bellman equations 23
    3.2.5 Exploration vs. exploitation 23
    3.2.6 Reinforcement learning algorithm categories 23
    3.2.7 Proximal Policy Optimization (PPO) 23
    3.2.8 Challenges 25
  3.3 Curriculum Learning 25
    3.3.1 Motivation and application in reinforcement learning 26
    3.3.2 Formalizing curriculum components 27
    3.3.3 Curriculum learning and transfer learning 28
    3.3.4 Automation and approaches 29

3.3.5 Benefits and challenges 29
3.4 Autoencoders 30
3.4.1 Architecture 30
3.4.2 Training objective 32
3.4.3 Use cases and variants 32
4. METHODOLOGY 35
4.1 Problem Formulation 35
4.2 Bayesian Curriculum Learning Framework 36
4.2.1 Probabilistic task modeling with bayesian networks 36
4.2.2 Task representation and difficulty evaluation 37
4.2.3 Clustering and adaptive task sequencing 37
4.3 Bayesian Curriculum Generation (BCG) Algorithm 38
4.3.1 Task representation processing 38
4.3.2 Difficulty assessment and normalization 38
4.3.3 Difficulty clustering and task selection 40
4.3.4 Probabilistic task generation and bayesian updating 41
4.3.5 Dynamic difficulty adjustment and thresholding 41
4.3.6 Computational complexity analysis 42
4.4 Hyperparameter Analysis and Configuration 43
4.5 Implementation Details 49
4.5.1 Software and hardware environment 49
4.5.2 Visual task representation: autoencoder implementation 50
4.5.3 Dimensionality reduction for visualization (t-SNE) 51
4.5.4 Visual difficulty assessment pipeline 54
4.5.5 Scalar difficulty assessment pipeline 55
5. RESULTS 57
5.1 Performance in the MiniGrid Environment 57
5.1.1 MiniGrid environment characteristics and setup 57
5.1.1.1 Evaluation metrics 58
5.1.2 Analysis of visual task representation 58
5.1.3 Comparative performance evaluation 62
5.1.4 BCG process analysis (Minigrid) 64
5.1.5 Minigrid summary 66
5.2 Performance in the AeroRival Pursuit Environment 67
5.2.1 Evaluation metrics 67
5.2.2 AeroRival pursuit environment characteristics 67
5.2.3 Comparative performance evaluation 68
5.2.4 AeroRival summary 70
6. DISCUSSIONS 71
6.1 Synthesis of Key Findings 71
6.2 Relation to Research Objectives and Questions 72
6.3 Comparison with Existing Literature 76
6.4 Implications and Significance 77
6.5 Limitations 77
6.6 Concluding Remarks on Discussion 78
7. CONCLUSION AND FUTURE WORK 79
7.1 Conclusion 79
7.2 Future Research Directions 80
REFERENCES 83
CURRICULUM VITAE 91

ABBREVIATIONS

AE	: Autoencoder
AIC	: Akaike Information Criterion
ANN	: Artificial Neural Network
BCE	: Binary Cross-Entropy
BCG	: Bayesian Curriculum Generation
BIC	: Bayesian Information Criterion
BN	: Bayesian Network
CL	: Curriculum Learning
CPD	: Conditional Probability Distribution
CPT	: Conditional Probability Table
DAG	: Directed Acyclic Graph
DDPG	: Deep Deterministic Policy Gradient
EM	: Expectation-Maximization
GAE	: Generalized Advantage Estimation
MAP	: Maximum A Posteriori
MCMC	: Markov Chain Monte Carlo
MDP	: Markov Decision Process
MLE	: Maximum Likelihood Estimation
MPE	: Most Probable Explanation
MSE	: Mean Squared Error
PGM	: Probabilistic Graphical Model
PPO	: Proximal Policy Optimization
ReLU	: Rectified Linear Unit
RL	: Reinforcement Learning
SAC	: Soft Actor-Critic
SGD	: Stochastic Gradient Descent
TD3	: Twin Delayed Deep Deterministic Policy Gradient
TRPO	: Trust Region Policy Optimization
t-SNE	: t-Distributed Stochastic Neighbor Embedding

SYMBOLS

$\boldsymbol{\hat{A}_t}$: Estimated advantage function at time t (PPO)
$\mathscr{A}$: Autoencoder network/function
$\boldsymbol{d(T_i)}$: Base difficulty measure (e.g., distance) for task T_i
$\mathscr{D}$: Dimensionality reduction function (e.g., t-SNE)
$\boldsymbol{e(\cdot)}$: Encoder function of Autoencoder
$\boldsymbol{d(\cdot)}$: Decoder function of Autoencoder
$\boldsymbol{G}$: Graph representing the Bayesian Network (V,E)
$\boldsymbol{G_t}$: Expected return (cumulative discounted reward) from time t
$\boldsymbol{L(\cdot,\cdot)}$: General loss function
$\boldsymbol{l(T_j)}$: Difficulty level assignment (index) for task T_j
$\boldsymbol{N_{\textbf{bins}}}$: Hyperparameter: number of difficulty clusters/bins (BCG)
$\boldsymbol{P}$: State transition function $P(s'|s,a)$; Probability distribution
$\mathbf{Pa}\boldsymbol{(V_i)}$: Parent nodes of V_i in the BN
$\boldsymbol{Q^{\pi}(s,a)}$: Action-value function (Q-function)
$\boldsymbol{r_t(\theta)}$: Probability ratio $\pi_\theta(a_t|s_t)/\pi_{\theta_{old}}(a_t|s_t)$ in PPO
$\boldsymbol{V}$: Set of nodes (variables) in the BN
$\boldsymbol{V^{\pi}(s)}$: State-value function
$\boldsymbol{\beta}$: Hyperparameter (BCG: threshold factor; PPO: KL coeff.)
$\boldsymbol{\gamma}$: Discount factor in RL
$\boldsymbol{\delta_i}$: Normalized difficulty score for task T_i
$\boldsymbol{\eta}$: Success threshold hyperparameter (BCG task completion)
$\boldsymbol{\lambda_1,\lambda_2}$: Weighting hyperparameters for similarity scores (BCG)
$\boldsymbol{\phi}$: Parameters of value function V_ϕ; Task embedding function
$\boldsymbol{\psi_i}$: Parameter vector representing task T_i
$\boldsymbol{\pi}$: Policy in RL (agent's strategy)
$\boldsymbol{\tau_t}$: Adaptive similarity threshold hyperparameter (BCG)
$\boldsymbol{\theta}$: General parameters (e.g., policy π_θ, BN CPDs θ_{ijk})

LIST OF TABLES

Page

Table 4.1: BCG Framework Hyperparameters: Settings and Effects (adapted from (Akgün & Üre, 2025)). **44**
Table 4.2: Detailed Architecture of the Deep Convolutional Autoencoder Network. The table specifies the layer type, output shape, spatial dimensions (HxW), feature depth, and filter size for each step in the encoder (compression) and decoder (reconstruction) paths (adapted from (Akgün & Üre, 2025)). **52**
Table 5.1: Quantitative Comparison of Algorithm Performance on MiniGrid Tasks. Mean reward ± std dev over three trials. Higher is better. Bold indicates best per task (adapted from (Akgün & Üre, 2025)). ... **63**
Table 5.2: Quantitative Comparison of Algorithm Performance on AeroRival Pursuit Tasks. Mean reward ± std dev over three trials. Positive values indicate success. Bold indicates best per task (adapted from (Akgün & Üre, 2025)). **68**

LIST OF FIGURES

Page

Figure 3.1: The standard agent-environment interaction loop in Reinforcement Learning. The agent observes state s_t, selects action a_t, receives reward r_{t+1} and transitions to state s_{t+1}. **22**

Figure 3.2: Conceptual illustration of Curriculum Learning. The model is initially trained on a small, easy subset of data/tasks (Curriculum stage Q_1), then progresses to larger, harder subsets (Q_t), eventually encompassing the entire target distribution ($Q_T = P$). Image adapted from (Wang, Chen, & Zhu, 2021). **26**

Figure 3.3: Typical architecture of an Autoencoder. The encoder maps the input x to a lower-dimensional latent code z. The decoder attempts to reconstruct the original input x' from the code z. The bottleneck structure forces the network to learn a compressed representation... **31**

Figure 4.1: Overview of the Bayesian Curriculum Generation (BCG) Algorithm (adapted from (Akgün & Üre, 2025)): This flowchart illustrates the structured approach. It starts with building a Bayesian Network for curriculum modeling. Unsupervised clustering differentiates task complexity levels, and a cost function guides task sequencing. The curriculum evolves dynamically through iterative cycles of task generation, difficulty evaluation, and agent training, repeating until the target task is mastered. This enhances sample efficiency and reduces training burden. **39**

Figure 4.2: Performance variation with β and N_{bins}. Warmer colors indicate higher combined performance (adapted from (Akgün & Üre, 2025)). **45**

Figure 4.3: Performance variation with η and β. Warmer colors indicate higher combined performance (adapted from (Akgün & Üre, 2025)). **46**

Figure 4.4: Performance variation with λ_1 and λ_2. Warmer colors indicate higher combined performance (adapted from (Akgün & Üre, 2025)). **47**

Figure 4.5: Conceptual illustration of the Convolutional Autoencoder architecture used for processing visual task inputs (e.g., from MiniGrid). Input images are compressed by the encoder into a latent feature space and subsequently reconstructed by the decoder (adapted from (Akgün & Üre, 2025)). **51**

Figure 4.6: Example 2D visualization produced by applying t-SNE to the latent features generated by the autoencoder's encoder network for various tasks. Each point corresponds to a task, and proximity suggests similarity in the learned latent space (adapted from (Akgün & Üre, 2025)). **53**
Figure 5.1: General illustration of a MiniGrid environment variant. The DoorKey tasks used involve navigating grids, finding a key (object), unlocking a door (obstacle), and reaching a goal (target square) **58**
Figure 5.2: Example 2D t-SNE visualization of latent features extracted from MiniGrid tasks by the autoencoder's encoder. Proximity of points suggests similarity in the learned feature space, potentially correlating with task difficulty or structure (adapted from (Akgün & Üre, 2025)). **59**
Figure 5.3: t-SNE projection highlighting tasks grouped into Cluster A (adapted from (Akgün & Üre, 2025)). **60**
Figure 5.4: Example task configurations from Cluster A, illustrating shared structural features (adapted from (Akgün & Üre, 2025)). **60**
Figure 5.5: t-SNE projection highlighting tasks grouped into Cluster B, distinct from Cluster A (adapted from (Akgün & Üre, 2025)). **61**
Figure 5.6: Example task configurations from Cluster B (adapted from (Akgün & Üre, 2025)). **61**
Figure 5.7: Illustration of Increasing Task Difficulty in MiniGrid Test Scenarios. (a) Task 0: Baseline map. (b) Task 1: Simple obstacles. (c) Task 2: Complex barriers, longer paths. (d) Task 3: Highest complexity map. This progression tests algorithm adaptability (adapted from (Akgün & Üre, 2025)). **62**
Figure 5.8: Learning Curves of RL Algorithms on MiniGrid Tasks. Shows mean reward ± std dev. BCG demonstrates consistent high performance and low variance. AMIGO/CBET excel early but decline. AGAC/DEIR show moderate, variable performance. RAPID/PPO lag. MASK fails to scale. PREFVEC struggles. Highlights BCG's adaptability (adapted from (Akgün & Üre, 2025)). **63**
Figure 5.9: Example illustration of task progression generated by BCG in MiniGrid, showing increasing complexity over the curriculum stages (adapted from (Akgün & Üre, 2025)). **65**
Figure 5.10: Conceptual depiction of adaptive task selection within BCG. If progress stalls, the framework can sample alternative tasks within or near the current difficulty level based on BN probabilities and performance feedback (adapted from (Akgün & Üre, 2025)). **65**

Figure 5.11: Training progress showing mean (and standard deviation) timesteps to task completion across curriculum stages in MiniGrid. Illustrates initial speed on easy tasks and successful learning despite increasing time on harder tasks (adapted from (Akgün & Üre, 2025)). **66**
Figure 5.12: AeroRival Pursuit Baseline Setup (Task 0). Single threat zone (adapted from (Akgün & Üre, 2025)). **68**
Figure 5.13: AeroRival Pursuit Complex Setup (Task 1). Three threat zones (adapted from (Akgün & Üre, 2025)). **68**
Figure 5.14: Learning Curves of RL Algorithms on AeroRival Pursuit Tasks. Mean reward ± std dev. BCG shows rapid convergence to high positive rewards in Task 0 and maintains strong performance in Task 1, benefiting from curriculum structure. Baselines struggle significantly, mostly failing to achieve positive rewards, especially in Task 1 (adapted from (Akgün & Üre, 2025)). **69**

MULTI-AGENT PLANNING WITH AUTOMATED CURRICULUM LEARNING

SUMMARY

Reinforcement learning (RL) represents a formidable paradigm for training autonomous agents to master sequential decision-making tasks. Its core principle, learning through trial and error guided by a reward signal, has proven successful in a variety of domains. However, the efficacy of standard RL algorithms diminishes drastically in environments characterized by sparse rewards or complex, high-dimensional state spaces. In these challenging settings, an agent receives meaningful feedback only after executing a long and specific sequence of correct actions. This "credit assignment problem" makes exploration, the process of discovering rewarding behaviors, profoundly inefficient. An agent may wander aimlessly without ever stumbling upon the feedback necessary to learn, preventing standard algorithms from developing effective policies.

To overcome this fundamental limitation, this thesis turns to curriculum learning (CL), a strategy inspired by the principles of human pedagogy. Just as we teach students arithmetic before calculus, CL structures the learning process by initially presenting the agent with simpler tasks and gradually increasing the difficulty as its competence grows. This guided approach helps the agent build foundational skills that can be leveraged to solve more complex problems. The primary bottleneck of traditional CL, however, is its reliance on manual design; creating an effective curriculum requires significant human expertise, intuition, and domain-specific knowledge, making it a process that is both laborious and difficult to generalize.

This thesis addresses this critical gap by proposing a novel framework for the automated and adaptive generation of learning curricula. The central objective was to develop, implement, and rigorously evaluate an algorithmic framework, termed Bayesian Curriculum Generation (BCG), designed to dynamically construct and adapt a curriculum based on an understanding of the task's underlying structure and the agent's real-time progress. The aim is to significantly enhance the performance, stability, and sample efficiency of RL agents, particularly in complex, sparse-reward scenarios where traditional methods falter.

The proposed BCG algorithm is built upon a synergistic integration of several key concepts. At its heart, the framework utilizes Bayesian Networks (BNs), a type of probabilistic graphical model, to represent the structural dependencies among the key parameters that define the tasks within an environment. For instance, in a navigation task, these parameters might include map size, the number of obstacles, or the presence of adversaries. The BN captures the probabilistic relationships between these parameters, serving as a powerful generative model. This allows the framework

to sample a diverse yet coherent set of task configurations, moving beyond simple parameter randomization to generate tasks with a principled structure.

A critical component of the framework is its ability to handle diverse input modalities through flexible task representation techniques. For visual environments like MiniGrid, where the state is an image, a convolutional autoencoder (CAE) is trained to compress high-dimensional observations into a low-dimensional latent feature vector. This vector captures the essential semantic content of the state, providing a compact and meaningful representation for analysis. For environments defined by a set of scalar parameters, such as the physics-based AeroRival simulator, normalized parameter vectors are used directly.

Once tasks are represented in a common feature space, their difficulty is quantified. This is typically achieved by measuring the distance (e.g., Euclidean distance) between a given task's representation and that of the final target task. The intuition is that tasks with representations closer to the target are more similar in the skills they require. These raw distance values are then normalized and processed using unsupervised clustering algorithms, such as K-Means, to automatically group tasks into a discrete number of difficulty levels or "bins." This process effectively creates the structured stages of the curriculum.

A defining feature of BCG is its adaptability. The curriculum is not a static, predefined sequence. Instead, the selection of tasks for the agent to train on is performed probabilistically, guided by the agent's real-time performance metrics, such as its average reward or task success rate. If an agent consistently succeeds at a certain difficulty level, the probability of sampling tasks from the next, more challenging level increases. Conversely, if the agent struggles, the framework can present it with easier tasks to help it consolidate its skills. This closed-loop system ensures the agent is always training at the edge of its capabilities, preventing both stagnation and frustration.

Crucially, the BCG framework implicitly and effectively leverages transfer learning to accelerate skill acquisition. The policy and value function parameters, learned by the base RL agent (in our evaluations, Proximal Policy Optimization - PPO) on tasks from one curriculum stage, are used to initialize the learning process for the subsequent, more challenging stage. This prevents the agent from having to learn from scratch at each step, allowing it to build upon previously acquired knowledge and dramatically speeding up convergence to an optimal policy for the final task.

The practical efficacy and robustness of the BCG framework were empirically validated through comprehensive experiments in two distinct and demanding RL environments. The first, MiniGrid (specifically, the DoorKey variant), provided a discrete, grid-based navigation challenge characterized by partial observability (the agent can only see a small portion of its surroundings) and a hierarchically sparse reward (the agent must first find a key, then navigate to a door, and only then receive a reward). The second, AeroRival Pursuit, offered a continuous control task involving high-speed adversarial interaction, dynamic hazard avoidance, and sparse rewards, simulating an aerial combat scenario. In both testbeds, BCG's performance was rigorously benchmarked against a baseline PPO agent (with no curriculum) and a diverse set of relevant contemporary algorithms designed to address similar challenges.

The experimental results consistently and unequivocally demonstrated the superiority of the BCG approach. Across both the discrete MiniGrid and continuous AeroRival environments, agents trained with BCG achieved significantly higher final performance levels and converged on successful policies more reliably than all tested baselines. Furthermore, BCG exhibited greater learning stability, as evidenced by a lower variance in performance across multiple independent training runs, indicating that its success is not due to random chance. Notably, in MiniGrid, BCG enabled the agent to master tasks of progressively increasing complexity where many baselines failed to scale. In the highly complex AeroRival environment, BCG was the only method that consistently enabled the agent to learn a successful policy, whereas most baselines failed to obtain any positive rewards at all. This success across environments with fundamentally different dynamics underscores the versatility and generality of the framework.

In conclusion, this research makes a significant contribution to the field of reinforcement learning by developing, implementing, and validating the Bayesian Curriculum Generation algorithm. BCG presents a robust, principled, and effective solution for automated and adaptive curriculum learning, particularly in challenging domains hampered by sparse rewards and complex state spaces. By synergistically combining probabilistic modeling of the task space, adaptive task selection driven by agent performance, and efficient knowledge transfer between stages, BCG successfully guides exploration and accelerates the acquisition of complex skills. While acknowledging certain limitations—such as the initial need for domain knowledge to identify parameters for the BN and the added computational overhead—the presented results are highly promising. Future work will focus on automating parameter selection, extending the framework to non-stationary environments, and further improving computational efficiency. Ultimately, BCG offers a powerful approach that advances the potential for training more capable, efficient, and autonomous AI agents in the complex scenarios of tomorrow.

OTOMATİK MÜFREDAT ÖĞRENMESİ İLE ÇOKLU AJAN PLANLAMASI

ÖZET

Pekiştirmeli öğrenme (RL), özerk ajanların ardışık karar verme görevlerinde ustalaşmaları için güçlü bir paradigma sunmaktadır. Temel prensibi olan, ödül sinyali rehberliğinde deneme-yanılma yoluyla öğrenme yaklaşımı, çeşitli alanlarda başarıyla uygulanmıştır. Ancak, standart RL algoritmalarının etkinliği, ödüllerin seyrek olduğu ya da karmaşık ve yüksek boyutlu durum uzaylarıyla karakterize edilen ortamlarda ciddi şekilde azalmaktadır. Bu tür zorlu ortamlarda bir ajan, ancak doğru eylemlerden oluşan uzun ve belirli bir dizi gerçekleştirdikten sonra anlamlı bir geri bildirim alabilir. Bu durum, keşif sürecini—yani ödüllendirici davranışların bulunmasını—son derece verimsiz kılan "kredi atfetme problemi" olarak adlandırılır. Ajan, öğrenmesi için gerekli geri bildirimi asla elde edemeden amaçsızca dolaşabilir ve bu da standart algoritmaların etkili politikalar geliştirmesini engeller.

Bu temel sınırlamanın üstesinden gelmek amacıyla, bu tezde insan pedagojisinden esinlenen bir strateji olan müfredat öğrenmesi (Curriculum Learning, CL) ele alınmaktadır. Nasıl ki öğrencilere kalkülüs öğretmeden önce aritmetik öğretiyorsak, CL de öğrenme sürecini başlangıçta daha basit görevlerle başlatıp, ajan yeterlilik kazandıkça zorluk derecesini kademeli olarak artırarak yapılandırır. Bu rehberli yaklaşım, ajanın daha karmaşık problemleri çözmek için kullanabileceği temel beceriler geliştirmesine olanak tanır. Ancak geleneksel CL'in temel darboğazı, büyük ölçüde manuel tasarıma bağımlı olmasıdır; etkili bir müfredat oluşturmak ciddi derecede insan uzmanlığı, sezgi ve alan bilgisi gerektirir; bu da süreci hem zahmetli hem de genelleştirilmesi güç kılar.

Bu tez, bu kritik boşluğu doldurmayı amaçlayarak öğrenme müfredatlarının otomatik ve uyarlanabilir bir şekilde oluşturulmasına yönelik özgün bir çerçeve önermektedir. Temel hedef, Bayesyen Müfredat Oluşturma (Bayesian Curriculum Generation, BCG) adı verilen ve görevin altında yatan yapının ve ajanın gerçek zamanlı gelişiminin anlaşılmasına dayalı olarak dinamik biçimde müfredat inşa eden ve uyarlayan algoritmik bir çerçevenin geliştirilmesi, uygulanması ve titizlikle değerlendirilmesidir. Amaç, özellikle geleneksel yöntemlerin başarısız olduğu, karmaşık ve ödülü seyrek ortamlarda, RL ajanlarının performansını, kararlılığını ve örnek verimliliğini önemli ölçüde artırmaktır.

Önerilen BCG algoritması, birkaç temel kavramın bütünleşik şekilde kullanılmasına dayanmaktadır. Çerçevenin merkezinde, Bayesyen Ağlar (Bayesian Networks, BN) adı verilen olasılıksal grafiksel modeller yer almakta olup, bir ortam içindeki görevleri tanımlayan temel parametreler arasındaki yapısal bağımlılıkları temsil etmektedir. Örneğin bir gezinme (navigasyon) görevinde bu parametreler harita boyutu, engel

sayısı veya rakip varlığı gibi unsurları içerebilir. BN, bu parametreler arasındaki olasılıksal ilişkileri yakalar ve güçlü bir üretici model işlevi görür. Bu sayede çerçeve, basit parametre rastgeleleştirmesini aşarak ilkesel bir yapıya sahip, çeşitli ve tutarlı görev konfigürasyonları üretebilir.

Çerçevenin kritik bir bileşeni de, esnek görev gösterim teknikleriyle farklı girdi türlerini işleyebilmesidir. MiniGrid gibi görsel ortamlarda, durumun bir görüntü olduğu durumlarda, yüksek boyutlu gözlemleri düşük boyutlu gizli özellik vektörüne sıkıştırmak üzere evrişimli bir otomatik kodlayıcı (CAE) eğitilmektedir. Bu vektör, durumun temel anlamsal içeriğini yakalayarak analiz için kompakt ve anlamlı bir gösterim sağlar. Fizik tabanlı AeroRival simülatörü gibi ortamlar ise doğrudan normalleştirilmiş parametre vektörleriyle temsil edilir.

Görevler ortak bir özellik uzayında temsil edildikten sonra, zorluk dereceleri nicel olarak belirlenir. Bu genellikle, belirli bir görevin gösterimi ile nihai hedef görevinki arasındaki (örneğin Öklidyen) mesafe ölçülerek yapılır. Bu yaklaşım, gösterimi hedefe yakın olan görevlerin gerektirdiği becerilerin de benzer olduğu varsayımına dayanır. Elde edilen ham mesafe değerleri daha sonra normalleştirilir ve K-Ortalamalar (K-Means) gibi gözetimsiz kümeleme algoritmalarıyla otomatik olarak ayrık zorluk seviyelerine veya “kutulara” gruplandırılır. Bu süreç, müfredatın yapılandırılmış aşamalarını etkin biçimde oluşturur.

BCG’nin ayırt edici özelliği, uyarlanabilirliğidir. Müfredat, statik ve önceden tanımlanmış bir dizi değildir. Ajanın eğitim alacağı görevlerin seçimi, ajanın gerçek zamanlı performans ölçütleri (ortalama ödül veya görev başarısı gibi) doğrultusunda olasılıksal olarak gerçekleştirilir. Ajan bir zorluk seviyesinde tutarlı şekilde başarılı olursa, bir sonraki daha zor seviyeden görev seçme olasılığı artar; tersi durumda ise çerçeve, ajanın becerilerini pekiştirmesi için daha kolay görevler sunabilir. Bu kapalı döngü sistem, ajanın her zaman yeteneklerinin sınırında eğitim almasını sağlar ve hem durağanlaşmayı hem de hayal kırıklığını önler.

Önemli olarak, BCG çerçevesi bilgi aktarımını (transfer learning) dolaylı ve etkili biçimde kullanarak yetenek edinimini hızlandırır. Temel RL ajanı (deneylerde Proximal Policy Optimization – PPO) tarafından bir müfredat aşamasındaki görevlerde öğrenilen politika ve değer fonksiyonu parametreleri, bir sonraki daha zorlu aşamadaki öğrenme süreci için başlangıç noktası olarak kullanılır. Bu sayede ajan her adımda tekrar baştan öğrenmek zorunda kalmaz, önceki bilgilerini üzerine inşa edebilir ve nihai göreve ulaşmak için optimal politikaya çok daha hızlı yakınsar.

BCG çerçevesinin pratik etkinliği ve sağlamlığı, iki farklı ve zorlu RL ortamında gerçekleştirilen kapsamlı deneylerle doğrulanmıştır. İlki, MiniGrid’in DoorKey varyantı olup, kısmi gözlemlenebilirliğe (ajan yalnızca çevresinin küçük bir kısmını görebilir) ve hiyerarşik olarak seyrek ödüllere sahip, ayrık ve ızgara tabanlı bir gezinme problemidir (ajan önce anahtarı bulmalı, ardından kapıya giderek ödül almalıdır). İkincisi ise AeroRival Pursuit olup, yüksek hızlı rakip etkileşimleri, dinamik tehlikelerden kaçınma ve seyrek ödüller içeren, sürekli kontrol gerektiren bir hava muharebe senaryosudur. Her iki test ortamında da, BCG’nin performansı, müfredatsız temel PPO ajanı ve benzer zorlukları çözmeye yönelik çağdaş algoritmalarla karşılaştırmalı olarak değerlendirilmiştir.

Deneysel sonuçlar, BCG yaklaşımının üstünlüğünü tutarlı ve açık biçimde ortaya koymuştur. Hem ayrık MiniGrid hem de sürekli AeroRival ortamlarında, BCG ile eğitilen ajanlar, test edilen tüm taban çizgilerine kıyasla anlamlı derecede daha yüksek nihai performans düzeylerine ulaşmış ve başarılı politikalara daha güvenilir biçimde yakınsamıştır. Ayrıca BCG, birden fazla bağımsız eğitim koşusunda daha düşük performans varyansı ile daha yüksek öğrenme kararlılığı sergilemiş, başarının rastlantısal olmadığını göstermiştir. Özellikle MiniGrid'de, BCG ajanların artan karmaşıklıktaki görevleri başarıyla öğrenmesini sağlarken, birçok taban çizgisi ölçeklenememiştir. Son derece karmaşık AeroRival ortamında ise, BCG ajanların tutarlı şekilde başarılı politika öğrenebildiği tek yöntem olmuş, çoğu taban çizgisi herhangi bir olumlu ödül elde edememiştir. Temel dinamikleri tamamen farklı iki ortamda dahi elde edilen bu başarı, çerçevenin çok yönlülüğünü ve genellenebilirliğini vurgulamaktadır.

Sonuç olarak, bu çalışma, Bayesyen Müfredat Oluşturma algoritmasını geliştirerek, uygulayarak ve doğrulayarak pekiştirmeli öğrenme alanına önemli bir katkı sunmaktadır. BCG, özellikle ödülün seyrek ve durum uzayının karmaşık olduğu zorlu alanlarda, otomatik ve uyarlanabilir müfredat öğrenimi için sağlam, ilkesel ve etkili bir çözüm sunmaktadır. Görev uzayının olasılıksal modellenmesi, ajanın performansına dayalı uyarlanabilir görev seçimi ve aşamalar arasında verimli bilgi aktarımını bütünleştirerek, BCG keşif süreçlerini etkin şekilde yönlendirmekte ve karmaşık becerilerin edinimini hızlandırmaktadır. Bazı sınırlamalar—örneğin BN için parametrelerin belirlenmesinde başlangıçta alan bilgisine ihtiyaç duyulması ve ek hesaplama maliyeti—olmakla birlikte, sunulan sonuçlar son derece umut vericidir. Gelecekteki çalışmalar, parametre seçiminin otomatikleştirilmesi, çerçevenin durağan olmayan ortamlara genişletilmesi ve hesaplama verimliliğinin artırılmasına odaklanacaktır. Sonuç olarak, BCG, yarının karmaşık senaryolarında daha yetenekli, verimli ve özerk yapay zekâ ajanlarının eğitilmesinde potansiyeli ileriye taşıyan güçlü bir yaklaşım sunmaktadır.

1. INTRODUCTION

1.1 Background

Reinforcement Learning (RL) has emerged as a powerful paradigm for training autonomous agents to perform complex tasks through trial-and-error interactions with their environment (Sutton & Barto, 2018). Despite impressive achievements in various domains such as game playing (Mnih, Kavukcuoglu, Silver, Rusu, Veness, Bellemare, Graves, Riedmiller, Fidjeland, Ostrovski, et al., 2015; Silver, Schrittwieser, Simonyan, Antonoglou, Huang, Guez, Hubert, Baker, Lai, Bolton, et al., 2017) and robotics (Levine, Finn, Darrell, & Abbeel, 2016), RL algorithms continue to face significant challenges in environments with sparse rewards, where meaningful feedback is only received after completing a sequence of correct actions (Andrychowicz, Wolski, Ray, Schneider, Fong, Welinder, McGrew, Tobin, Abbeel, & Zaremba, 2017).

In sparse reward settings, agents must explore vast state-action spaces with minimal guidance, often resulting in inefficient learning processes and poor sample complexity (Pathak, Agrawal, Efros, & Darrell, 2017). This fundamental challenge has motivated research into methods that can structure the learning process to make it more efficient and tractable (Bengio, Louradour, Collobert, & Weston, 2009).

Curriculum learning, inspired by human educational practices, offers a promising approach by organizing the learning experience through a progression of increasingly difficult tasks (Bengio, Louradour, Collobert, & Weston, 2009; Narvekar, Peng, Leonetti, Sinapov, Taylor, & Stone, 2020). Rather than immediately confronting an agent with the full complexity of a target task, curriculum learning introduces simpler, related tasks that allow the agent to build foundational knowledge and skills gradually. This approach has been shown to accelerate learning and improve final performance across various domains (Graves, Bellemare, Menick, Munos, & Kavukcuoglu, 2017; Matiisen, Oliver, Cohen, & Schulman, 2019).

However, designing effective curricula presents its own challenges. Traditional approaches often rely on manually crafted task sequences designed by domain experts (Narvekar, Peng, Leonetti, Sinapov, Taylor, & Stone, 2020), which can be labor-intensive and may not generalize well across different environments or agents. Automating curriculum generation has thus become an active area of research, with approaches ranging from task sequencing based on learning progress (Graves, Bellemare, Menick, Munos, & Kavukcuoglu, 2017) to procedural content generation (Justesen, Torrado, Bontrager, Khalifa, Togelius, & Risi, 2018) and goal generation (Florensa, Held, Geng, & Abbeel, 2018).

Despite these advances, existing automated curriculum methods often face limitations: many are tightly coupled with specific RL algorithms, limiting their applicability; others lack mechanisms to efficiently evaluate task difficulty without extensive training; and most struggle to adapt dynamically to the agent's evolving capabilities during the learning process (Narvekar, Peng, Leonetti, Sinapov, Taylor, & Stone, 2020; Portelas, Colas, Hofmann, & Oudeyer, 2020a).

This thesis introduces the Bayesian Curriculum Generator (BCG) (Akgün & Üre, 2025), a novel approach that addresses these limitations. Our method leverages Bayesian networks to dynamically create and adapt curricula for RL agents in sparse reward environments. The algorithm operates by systematically altering environment parameters to generate tasks of varying difficulty, creating a structured learning pathway that evolves based on the agent's performance.

A key innovation of our approach is its independence from the underlying RL algorithm, allowing it to be integrated with various learning techniques as a modular component. Additionally, our algorithm incorporates an unsupervised task classification system that efficiently categorizes tasks based on difficulty without requiring exhaustive training for each potential task. This classification can operate on both image outputs and scalar parameter values, providing flexibility across different environment types.

BCG represents a significant advancement in curriculum learning for reinforcement learning, offering a dynamic and adaptive solution that enhances learning efficiency in complex environments with sparse rewards.

1.2 Purpose of Thesis

The primary purpose of this thesis is to address the fundamental challenges of reinforcement learning in sparse reward environments through the development, analysis, and validation of the Bayesian Curriculum Generator (BCG). This research aims to create a novel framework that dynamically adapts learning progression to match agent capabilities, making complex reinforcement learning tasks more tractable.

1.3 Research Objectives

This thesis pursues six core objectives that collectively advance the field of curriculum learning for reinforcement learning:

1. Develop a robust and generalizable methodology for automatically generating curricula that can adapt to the learning progress of RL agents in sparse reward settings.

2. Design and implement an unsupervised task classification system that can efficiently categorize tasks based on difficulty without requiring extensive training for each potential task.

3. Create a modular curriculum generation framework that operates independently from the underlying RL algorithm, enabling compatibility with a wide range of learning techniques.

4. Establish a theoretical foundation for the relationship between Bayesian parameter modification and task difficulty in the context of curriculum learning.

5. Empirically evaluate the effectiveness of the proposed approach across diverse environments with varying characteristics, including maze-like structures, discrete and continuous action spaces, and adversarial elements.

6. Contribute to the broader understanding of curriculum learning by analyzing the conditions under which dynamically generated curricula enhance learning efficiency.

1.4 Research Questions

To guide our investigation, this thesis addresses five fundamental questions about curriculum learning in reinforcement learning environments, with particular emphasis on probabilistic modeling approaches:

RQ1: How can Bayesian networks be effectively utilized to create adaptive curriculum learning frameworks that systematically structure task progressions based on agent capabilities?

RQ2: What advantages might probabilistic curriculum generation offer compared to existing approaches such as teacher-student frameworks, intermediate goal creation, and self-play methods?

RQ3: How can a curriculum learning system effectively balance between task difficulty and agent capability in environments with sparse rewards or complex skill hierarchies?

RQ4: To what extent can a generalized curriculum learning framework maintain effectiveness across diverse reinforcement learning algorithms and environmental contexts?

RQ5: What design principles enable curriculum learning systems to adapt dynamically to changing agent capabilities throughout the training process?

1.5 Significance and Potential Applications

By developing and validating the Bayesian Curriculum Generation (BCG) framework (Akgün & Üre, 2025), this thesis aims to make a significant contribution to the field of curriculum learning for reinforcement learning. The research provides a novel, principled approach for automating the generation of adaptive curricula, offering researchers and practitioners a potent tool particularly suited for enhancing learning efficiency and final performance in complex environments where sparse rewards pose a major obstacle. The demonstrated success of BCG in navigating such challenges suggests wide-ranging potential applications across various domains.

In robotics, where acquiring complex manipulation skills (Andrychowicz, Baker, Chociej, Jozefowicz, McGrew, Pachocki, Petron, Plappert, Powell, Ray, et al., 2020; Gu, Holly, Lillicrap, & Levine, 2017) or robust navigation strategies in unstructured environments (Kahn, Villaflor, Ding, Abbeel, & Levine, 2018) often involves extensive trial-and-error with sparse feedback, BCG's ability to structure learning through an adaptive sequence of tasks could drastically reduce training time and improve generalization. The framework's capacity to handle both continuous parameters and visual inputs further enhances its applicability to diverse robotic systems.

The adaptive nature of BCG also aligns well with personalized educational technologies. By modeling task difficulty (analogous to problem difficulty) and adapting the sequence based on learner performance (analogous to student progress), similar principles could inform systems that dynamically tailor educational content to individual needs, potentially improving learning outcomes and engagement (Clement, Roy, Oudeyer, & Lopes, 2015; Tabibian, Upadhyay, De, Zarezade, Schölkopf, & Gomez-Rodriguez, 2019).

Furthermore, in high-stakes domains like healthcare, where RL is explored for treatment optimization (Yu, Liu, & Nemati, 2021) or resource management (Shortreed, Laber, Lizotte, Stroup, Pineau, & Murphy, 2011), minimizing sample requirements during learning is critical. BCG's curriculum-based approach, by guiding the agent efficiently, holds the potential to reduce the amount of interaction needed to learn effective policies, making RL applications more feasible in these data-sensitive or safety-critical contexts.

Additionally, the framework could be highly beneficial in complex game AI development. Training agents to achieve human-level or superhuman performance in sophisticated games often involves mastering intricate skill hierarchies with delayed rewards (OpenAI et al., 2019; Vinyals, Babuschkin, Czarnecki, Mathieu, Dudzik, Chung, Choi, Powell, Ewalds, Georgiev, et al., 2019). An automated curriculum generator like BCG could structure the acquisition of these skills, potentially accelerating the development of highly capable game-playing agents.

Beyond specific applications, the theoretical contributions of this work—demonstrating the effective use of Bayesian Networks for modeling task dependencies within a CL framework and developing specific adaptive mechanisms—inform broader research on automated machine learning, skill acquisition, and transfer learning. The principles underlying BCG relate to how complex knowledge can be built incrementally, drawing connections to lifelong learning concepts and efficient knowledge transfer strategies explored in works like Progressive Networks (Rusu, Rabinowitz, Desjardins, Soyer, Kirkpatrick, Kavukcuoglu, Pascanu, & Hadsell, 2016) and Meta-Learning (e.g., MAML (Finn, Abbeel, & Levine, 2017)).

By directly addressing our research questions concerning adaptive, structured curriculum generation and its advantages, this thesis contributes to a deeper understanding of how to design effective learning progressions for artificial agents. The practical BCG framework and the empirical results provide valuable insights and tools for systematically overcoming the bottlenecks of sparse rewards and complex skill acquisition that currently limit the reach of reinforcement learning in many real-world applications.

2. LITERATURE REVIEW

This chapter examines the foundational concepts and recent advances relevant to our research. We explore three main areas: reinforcement learning fundamentals, Bayesian networks, and curriculum learning approaches.

2.1 Reinforcement Learning Fundamentals

Reinforcement learning (RL) represents a computational approach to learning through interaction with an environment. In RL, an agent learns to make decisions by taking actions and receiving feedback in the form of rewards or penalties. This learning paradigm has demonstrated remarkable achievements across various domains including game playing and robotics applications (Duan, Chen, Houthooft, Schulman, & Abbeel, 2016; Silver, Hubert, Schrittwieser, Antonoglou, Lai, Guez, Lanctot, Sifre, Kumaran, Graepel, et al., 2018).

Despite these successes, reinforcement learning faces significant challenges when rewards are sparse–situations where meaningful feedback is only received after completing a sequence of correct actions (Florensa, Held, Geng, & Abbeel, 2018; Florensa, Held, Wulfmeier, Zhang, & Abbeel, 2017). This sparsity creates exploration difficulties, as the agent may struggle to discover reward-yielding action sequences through random exploration alone (Parisi, Dean, Pathak, & Gupta, 2021; Wan, Tang, Tian, & Kaneko, 2023).

The core principles of reinforcement learning encompass several interconnected concepts. Agents learn primarily through trial-and-error interactions with their environment, gradually improving decision-making through experience (Duan, Chen, Houthooft, Schulman, & Abbeel, 2016; Silver, Hubert, Schrittwieser, Antonoglou, Lai, Guez, Lanctot, Sifre, Kumaran, Graepel, et al., 2018). This process involves delayed feedback mechanisms, where the consequences of actions may only become apparent after several subsequent steps (Sukhbaatar, Lin, Kostrikov, Synnaeve, Szlam,

& Fergus, 2017; Sukhbaatar, Szlam, Synnaeve, Chintala, & Fergus, 2015). A central challenge in RL involves balancing exploration of unknown possibilities with exploitation of known rewarding strategies (Campero, Raileanu, Küttler, Tenenbaum, Rocktäschel, & Grefenstette, 2020; Zha, Ma, Yuan, Hu, & Liu, 2021). Throughout this process, agents optimize their policies based on cumulative rewards rather than immediate gains, enabling long-term strategic planning (Szoke, Shperberg, Holtz, & Allievi, 2024; Uchendu, Xiao, Lu, Zhu, Yan, Simon, Bennice, Fu, Ma, Jiao, et al., 2023). These principles form the foundation for the more specialized approaches discussed in subsequent sections, particularly as they relate to curriculum development and learning optimization.

2.2 Bayesian Networks

Bayesian networks (BNs) provide a powerful framework for modeling complex systems as networks of interactive variables, portraying relationships from fundamental causes to ultimate outcomes while making all cause-effect assumptions explicit (Voinov & Bousquet, 2010). Their effectiveness in analyzing tradeoffs and integrating multiple factors makes them particularly suitable for environmental systems modeling, though their applications extend much further. A key advantage of Bayesian networks is their relatively simple causal graphical structure, which allows for construction without highly specialized technical modeling expertise. This accessibility enables both technical and non-technical stakeholders to understand and interpret the models–a particularly valuable attribute in collaborative environments requiring transdisciplinary approaches (Castelletti & Soncini-Sessa, 2007; Voinov & Bousquet, 2010).

The operational foundation of Bayesian networks rests on Conditional Probability Tables (CPTs) associated with each node in the network. These tables quantify relationship strengths by specifying probabilistic beliefs about node states based on parent node conditions. When observations or scenario values are introduced as evidence, the network updates a priori probabilities across all nodes using Bayes' Theorem and belief propagation mechanisms (Chen & Pollino, 2012). This propagation capability supports both diagnostic (bottom-up) and explanatory (top-down) reasoning processes (Castelletti & Soncini-Sessa, 2007). Unlike "black box" approaches such as neural networks (Chen, Jakeman, & Norton, 2008), Bayesian networks maintain transparency by clearly presenting variable interactions, allowing users to interrogate the reasoning behind model outputs and facilitating system understanding.

Bayesian networks offer notable advantages for classification and prediction tasks even when working with incomplete or ambiguous data (Newton, 2010). This capability represents a significant benefit compared to conventional statistical models that typically require substantial empirical datasets for construction (Marcot, Steventon, Sutherland, & McCann, 2006). The flexible nature of Bayesian networks allows for integration of expert knowledge, empirical data, and theoretical constructs within a unified modeling framework. This integration capability makes Bayesian networks particularly valuable in domains where comprehensive data collection may be impractical or prohibitively expensive, allowing for model development and refinement through iterative incorporation of diverse information sources (Chen & Pollino, 2012; Newton, 2010).

2.3 Curriculum Learning

Curriculum learning represents an approach to training in which learning experiences are organized in progressively increasing complexity, similar to educational curricula for human learners. This methodology has proven particularly valuable in reinforcement learning contexts, where it enables the creation of personalized and efficient learning paths for autonomous agents. According to the comprehensive survey by K. Gupta, Mukherjee, and Najjaran (2022), curriculum methods can be categorized into several distinct approaches, each contributing uniquely to the learning process.

The following subsections explore these primary curriculum learning methodologies, examining their implementations, advantages, and limitations compared to our proposed BCG approach.

2.3.1 Teacher-student interaction methods

In teacher-student curriculum learning, a teacher agent guides a student agent based on performance metrics, facilitating the development of an automatic curriculum (Graves, Bellemare, Menick, Munos, & Kavukcuoglu, 2017; Matiisen, Oliver, Cohen, & Schulman, 2019). This framework positions the teacher as a semi-explicit supervisor that adjusts tasks based on student performance. Portelas and colleagues expanded this approach to accommodate both discrete and continuous task environments using Gaussian mixture models for environment sampling (Portelas, Colas, Hofmann, & Oudeyer, 2020b), demonstrating improved efficiency in complex learning scenarios (Brockman, Cheung, Pettersson, Schneider, Schulman, Tang, & Zaremba, 2016).

Recent innovations include Diaz's data-centric perspective analyzing teacher-student dynamics through cooperative game theory (Diaz, Paull, & Tacchetti, 2024), Bajaj's "Task Phasing" approach that progressively increases task complexity using demonstrations (Bajaj, Sharon, & Stone, 2023), and Li's framework transforming single-task RL problems into multi-task problems through curriculum development (Li, Zhai, Ma, & Levine, 2023). While effective, these approaches generally lack the nuanced adaptability of probabilistic modeling that characterizes Bayesian methods.

2.3.2 Intermediate goal creation

Intermediate goal methods establish achievable milestones that guide agent progression through complex environments with sparse rewards (K. Gupta & Najjaran, 2021). Florensa's Goal Generative Adversarial Network (Goal GAN) generates intermediate goals calibrated to an agent's current capabilities—challenging enough to promote learning while remaining achievable (Florensa, Held, Geng, & Abbeel, 2018). This enhances learning processes particularly in robotic domains, though it requires expert input for goal specification.

An alternative approach, reverse curriculum learning (Florensa, Held, Wulfmeier, Zhang, & Abbeel, 2017), establishes progressively distant starting points toward a fixed goal state, simplifying complex trajectory learning by initially positioning the agent closer to success. Both approaches offer valuable frameworks but typically require substantial domain knowledge or predetermined constraints compared to more adaptive Bayesian approaches.

2.3.3 Self-play approaches

Self-play has transformed game-based reinforcement learning by enabling "tabula rasa" agents to begin without prior knowledge and rapidly progress to expert performance levels (Silver, Hubert, Schrittwieser, Antonoglou, Lai, Guez, Lanctot, Sifre, Kumaran, Graepel, et al., 2018). Sukhbaatar and colleagues (Sukhbaatar, Lin, Kostrikov, Synnaeve, Szlam, & Fergus, 2017) enhanced traditional self-play by integrating intrinsic motivation principles, creating a framework supporting navigation through expansive environments. Their implementation utilizes two agent modules in a coordinated learning process, demonstrating effectiveness across varied environments including Mazebase and StarCraft (Duan, Chen, Houthooft, Schulman, & Abbeel, 2016; Sukhbaatar, Szlam, Synnaeve, Chintala, & Fergus, 2015).

Despite these advantages, self-play approaches typically require substantial data resources and may experience stability issues due to their reliance on continuously generated tasks. Probabilistic frameworks like BCG can offer more structured adaptation while reducing dependence on extensive data collection.

2.3.4 Exploration-enhanced approaches

Several strategies improve exploration in reinforcement learning by implicitly implementing curricula through guided exploration. Campero's AMIGO (Campero, Raileanu, Küttler, Tenenbaum, Rocktäschel, & Grefenstette, 2020) utilizes a goal-generating teacher to suggest progressively complex intrinsic objectives, naturally inducing task progression. The Adversarially Guided Actor-Critic (AGAC) algorithm (Flet-Berliac, Ferret, Pietquin, Preux, & Geist, 2021) employs adversarial critics to encourage novel strategy discovery without systematic sequencing.

Other approaches include Parisi's Change-Based Exploration Transfer (C-BET) (Parisi, Dean, Pathak, & Gupta, 2021), which facilitates skill reuse across environments, and novelty-based exploration methods like DEIR (Wan, Tang, Tian,

& Kaneko, 2023) and RAPID (Zha, Ma, Yuan, Hu, & Liu, 2021), which derive exploration rewards from novelty and imitation respectively. These methods provide effective exploration strategies but depend more on environmental characteristics than explicitly designed task sequences.

2.3.5 Recent advances in structured progression

Contemporary curriculum learning research has increasingly focused on explicit task progression structures. Sayar's COHER (Sayar, Iacca, & Knoll, 2024) dynamically adjusts training environment complexity without requiring explicit obstacle position knowledge, demonstrating effectiveness in real-world robotic applications. Niu's GOATS (Niu, Jin, Zhang, Zhu, Zhao, & Zhang, 2023) employs goal-factorized reward formulations for robotic skill development through progressive interpolation of goal distributions.

Other advances include Margolis's adaptive curriculum approaches with online system identification for legged robots (Margolis, Yang, Paigwar, Chen, & Agrawal, 2024), Szoke's PREFVEC (Szoke, Shperberg, Holtz, & Allievi, 2024) for addressing environments with imbalanced reward components, Lee's CQM (Lee, Cho, Park, & Kim, 2024) leveraging vector quantized-variational autoencoders for semantic goal representations, and Uchendu's JSRL (Uchendu, Xiao, Lu, Zhu, Yan, Simon, Bennice, Fu, Ma, Jiao, et al., 2023) improving sample efficiency through guide-policy initialization. These approaches collectively demonstrate how structured progression can enhance learning across diverse domains and challenges.

2.4 Comparative Analysis and Research Positioning

Our proposed Bayesian Curriculum Generator (BCG) (Akgün & Üre, 2025) builds upon these foundations while addressing several limitations in existing approaches. While sharing fundamental principles with teacher-student methods such as TSCL (Graves, Bellemare, Menick, Munos, & Kavukcuoglu, 2017; Matiisen, Oliver, Cohen, & Schulman, 2019) and environment sampling using Gaussian mixture models (Linden, Lopes, & Bidarra, 2013; Portelas, Colas, Hofmann, & Oudeyer, 2020b), BCG demonstrates enhanced adaptability, seamless integration capabilities with diverse reinforcement learning algorithms, and optimized task sequencing through unsupervised learning and adaptive Bayesian networks.

Though BCG shares conceptual similarities with intermediate goal creation methods (Florensa, Held, Geng, & Abbeel, 2018; Florensa, Held, Wulfmeier, Zhang, & Abbeel, 2017) in guiding agents through progressively complex tasks, it offers superior adaptability and precision through its probabilistic modeling framework. Similarly, compared to self-play methods (Duan, Chen, Houthooft, Schulman, & Abbeel, 2016; Sukhbaatar, Lin, Kostrikov, Synnaeve, Szlam, & Fergus, 2017; Sukhbaatar, Szlam, Synnaeve, Chintala, & Fergus, 2015), BCG provides more structured progression while reducing data collection requirements.

Given the demonstrated success of implicit curriculum methods in environments like MiniGrid and the explicit progression enabled by recent research advances, these works serve as valuable comparative baselines for evaluating our approach's effectiveness. The BCG methodology, with its nuanced probabilistic curriculum structure, offers a comprehensive solution addressing limitations in both traditional and contemporary approaches while maintaining their core advantages in progressive skill development.

3. BACKGROUND

This chapter provides the foundational knowledge required to understand the core concepts underpinning this thesis. We delve into three key areas: Bayesian Networks, which form the basis of our probabilistic task modeling; Reinforcement Learning, the paradigm within which our learning problem is situated; and Curriculum Learning, the training strategy that our proposed method automates and enhances. Each section aims to provide sufficient detail for readers who may not be deeply familiar with these topics.

3.1 Bayesian Networks

Reasoning and making decisions in complex, real-world scenarios often involve dealing with uncertainty and intricate dependencies between various factors. Probabilistic Graphical Models (PGMs) provide a powerful framework for representing and reasoning about such uncertain knowledge, combining principles from graph theory and probability theory. Bayesian Networks (BNs), also known as Belief Networks or Directed Probabilistic Graphical Models, are a prominent type of PGM that represent conditional dependencies among a set of random variables using a Directed Acyclic Graph (DAG) (Jensen, 1996; Koller & Friedman, 2009; Pearl, 1988).

3.1.1 Formal definition

Formally, a Bayesian Network for a set of random variables $V = \{V_1, V_2, \ldots, V_n\}$ consists of two components:

1. *Directed acyclic graph (DAG): $G = (V, E)$*, where each node $V_i \in V$ corresponds to a random variable V_i, and the directed edges $E \subseteq V \times V$ represent direct conditional dependencies between these variables. An edge from V_i to V_j indicates that V_i has a direct influence on V_j. If such an edge exists, V_i is called a *parent* of V_j, denoted

$V_i \in \mathrm{Pa}_G(V_j)$ or simply $\mathrm{Pa}(V_j)$ when the graph G is clear from context. The acyclic nature means there are no directed paths $V_{k_1} \rightarrow V_{k_2} \rightarrow \cdots \rightarrow V_{k_m}$ such that $V_{k_1} = V_{k_m}$.

2. *Set of conditional probability distributions (CPDs):* For each variable V_i, there is a CPD, $P(V_i|\mathrm{Pa}(V_i))$, that quantifies the probabilistic relationship between V_i and its parents. This distribution specifies the probability (or probability density for continuous variables) of V_i taking on a particular value v_i given any combination of values pa_j assigned to its parent variables $\mathrm{Pa}(V_i)$.

The random variables V_i can be discrete or continuous.

- For *discrete variables*, the CPD $P(V_i|\mathrm{Pa}(V_i))$ is typically represented as a Conditional Probability Table (CPT). A CPT lists the probability $P(V_i = k|\mathrm{Pa}(V_i) = j)$ for each possible state (value) k of V_i and each possible configuration j of its parents $\mathrm{Pa}(V_i)$. Let V_i have r_i possible values and its parents $\mathrm{Pa}(V_i)$ have q_i possible configurations. Then the CPT for V_i contains $r_i \times q_i$ probability values. For each parent configuration j, the probabilities must sum to one: $\sum_{k=1}^{r_i} P(V_i = k|\mathrm{Pa}(V_i) = j) = 1$. The specific parameters defining these probabilities are often denoted $\theta_{ijk} = P(V_i = k|\mathrm{Pa}(V_i) = j)$.

- For *continuous variables*, common CPDs include linear Gaussian models (Geiger & Heckerman, 1994), where $P(V_i|\mathrm{Pa}(V_i) = pa_j)$ is a normal distribution $\mathcal{N}(\mu_j, \sigma_j^2)$ whose mean μ_j is a linear function of the parent values pa_j, i.e., $\mu_j = w_0 + \sum_{V_p \in \mathrm{Pa}(V_i)} w_p v_p$, where v_p is the value of parent V_p in configuration pa_j. Other forms like logistic functions (for modeling discrete children of continuous parents) are also used.

The set of all parameters defining the CPDs for all variables in the network is often denoted collectively by θ.

3.1.2 Key concepts and properties

3.1.2.1 Conditional independence and the markov property

The structure of the DAG G encodes assumptions about conditional independence. The *Local Markov Property* states that each variable V_i is conditionally independent of its non-descendants, given its parents $\text{Pa}(V_i)$. This is a fundamental assumption linking the graph structure to probabilistic independence.

More generally, the graph structure G defines a set of conditional independence assertions $(X \perp Y|Z)_G$, meaning the set of variables X is independent of the set Y given the set Z, according to the graph structure. These independencies can be determined using the graphical criterion called *d-separation* (directed separation) (Pearl, 1988). Two sets of nodes X and Y are d-separated by a set of evidence nodes Z if every undirected path between any node in X and any node in Y is "blocked" by Z. A path is blocked if there is a node W on the path such that either:

1. $W \in Z$ and the path connections meeting at W are serial ($\rightarrow W \rightarrow$) or diverging ($\leftarrow W \rightarrow$).
2. $W \notin Z$, and neither are any of its descendants, and the path connections meeting at W are converging ($\rightarrow W \leftarrow$). This node W is often called a collider or v-structure.

If all paths are blocked, then $(X \perp Y|Z)_G$ holds. This criterion allows us to read off all conditional independencies implied by the BN structure directly from the graph.

3.1.2.2 Factorization of the joint probability distribution

A key consequence of the graph structure and the associated conditional independence assumptions (specifically, the ordered Markov property derived from the DAG) is the ability to factorize the full joint probability distribution $P(V_1, \ldots, V_n)$ over all variables into a product of local CPDs (Pearl, 1988):

$$P(V_1, V_2, \ldots, V_n) = \prod_{i=1}^{n} P(V_i|\text{Pa}(V_i)) \tag{3.1}$$

This factorization is immensely powerful. Consider n binary variables. The full joint distribution requires specifying $2^n - 1$ parameters. However, if each variable in the BN has at most k parents, the number of parameters needed is $\sum_{i=1}^{n} |\text{States}(V_i)| \times |\text{Configs}(\text{Pa}(V_i))| \leq \sum_{i=1}^{n} 2 \times 2^k = n \cdot 2^{k+1}$. If k is small compared to n, the BN provides an exponential reduction in the number of parameters required to represent the distribution, making both representation and computation more tractable (Koller & Friedman, 2009).

3.1.3 Inference and learning

3.1.3.1 Probabilistic inference

Inference in BNs involves computing probabilities of interest given the network structure G, parameters θ, and potentially some observed evidence $E \subset V$. Let $Q \subset V$ be the query variables. Common inference tasks include:

- *Marginal probability:* Computing $P(Q)$ by summing (or integrating for continuous variables) over all other variables $W = V \setminus Q$:

$$P(Q) = \sum_{W} P(Q, W) = \sum_{W} \prod_{i=1}^{n} P(V_i|\text{Pa}(V_i)) \tag{3.2}$$

- *Conditional probability (Posterior probability):* Computing $P(Q|E = e)$, where e represents the observed values for variables in E. Using the definition of conditional probability:

$$P(Q|E = e) = \frac{P(Q, E = e)}{P(E = e)} = \frac{\sum_{W'} P(Q, E = e, W')}{\sum_{Q'} \sum_{W'} P(Q', E = e, W')} \tag{3.3}$$

  where $W' = V \setminus (Q \cup E)$. Both numerator and denominator are computed via marginalization.

- *Most probable explanation (MPE):* Finding the most likely assignment of values w^* to a set of unobserved variables $W = V \setminus E$ given evidence $E = e$:

$$w^* = argmax_w P(W = w|E = e) = argmax_w P(W = w, E = e) \tag{3.4}$$

Exact inference is generally NP-hard (Cooper, 1990). Efficient exact algorithms like *Variable Elimination* (Dechter, 1996) exist but scale poorly with graph complexity (treewidth). Approximate methods like *Markov Chain Monte Carlo (MCMC)* sampling (e.g., Gibbs sampling (Geman & Geman, 1984)) or *Variational Inference* (Jordan, Ghahramani, Jaakkola, & Saul, 1999) are widely used for larger networks.

3.1.3.2 Learning bayesian networks

Learning BNs from a dataset $D = \{d_1, \ldots, d_m\}$ of m independent observations involves two primary tasks (Heckerman, 1995):

- *Parameter learning (Given structure G):* Estimate parameters θ.
  - *Maximum likelihood estimation (MLE):* Find parameters θ that maximize the likelihood of the data: $\hat{\theta}_{MLE} = argmax_{\theta} P(D|G,\theta)$. Assuming complete data, the likelihood function factorizes according to the BN structure:

    $$L(\theta|D,G) = P(D|G,\theta) = \prod_{j=1}^{m} P(d_j|G,\theta) = \prod_{j=1}^{m}\prod_{i=1}^{n} P(v_{ij}|pa_{ij},\theta_i) \qquad (3.5)$$

    where v_{ij} and pa_{ij} are the values of V_i and its parents in data point d_j, and θ_i are the parameters for V_i's CPD. This often decomposes into independent maximizations for each node V_i. For discrete variables with CPTs, the MLE is given by frequency counts: $\hat{\theta}_{ijk} = \frac{N_{ijk}}{N_{ij}}$, where N_{ijk} counts observations where $V_i = k$ and $\mathrm{Pa}(V_i) = j$, and $N_{ij} = \sum_k N_{ijk}$.
  - *Maximum a posteriori (MAP) estimation:* Incorporate prior beliefs $P(\theta|G)$ about the parameters using Bayes' theorem: $P(\theta|D,G) \propto P(D|G,\theta)P(\theta|G)$. Find the parameters maximizing the posterior: $\hat{\theta}_{MAP} = argmax_{\theta} P(\theta|D,G)$. If conjugate priors are used (e.g., Dirichlet priors for multinomial CPDs), the posterior distribution has the same form as the prior, simplifying calculation (Heckerman, 1995). With a Dirichlet prior with hyperparameters $\boldsymbol{\alpha} = \{\alpha_{ijk}\}$, the MAP estimate for discrete variables is $\hat{\theta}_{ijk} = \frac{N_{ijk}+\alpha_{ijk}-1}{N_{ij}+\alpha_{ij}-|V_i|}$, where $\alpha_{ij} = \sum_k \alpha_{ijk}$ and $|V_i|$ is the number of states of V_i. This acts like adding pseudo-counts based on the prior.

  Handling missing data or latent variables typically requires iterative methods like the Expectation-Maximization (EM) algorithm (Dempster, Laird, & Rubin, 1977; Lauritzen, 1995).

- *Structure learning (Structure G unknown):* Learn the DAG G. This is computationally harder due to the large space of possible DAGs. Common approaches include:
    - *Score-based methods:* Define a scoring function $\mathrm{Score}(G|D)$ that measures how well a structure G fits the data D, often penalizing complexity. Search for the structure G^* maximizing the score. Common scores include Bayesian scores (like BDeu (Heckerman, 1995)) derived from $P(G|D) \propto P(D|G)P(G)$, or information-theoretic scores like BIC (Schwarz, 1978) and AIC. For example:

      $$\mathrm{BIC}(G|D) = \log P(D|G, \hat{\theta}_{MLE}) - \frac{\log m}{2}|\theta_G| \tag{3.6}$$

      where $|\theta_G|$ is the number of independent parameters in the model for structure G. The search often involves heuristic methods like greedy hill-climbing.
    - *Constraint-based methods:* Perform statistical tests of conditional independence on the data (e.g., χ^2 tests) to identify constraints, then find a DAG consistent with these independence facts (e.g., using the PC algorithm (Spirtes, Glymour, & Scheines, 2000)).

In practice, domain knowledge is often used to constrain the structure search or validate learned structures.

3.1.4 Advantages and use

Bayesian Networks provide a mathematically rigorous and intuitive framework for modeling complex systems involving uncertainty and interdependence. Key advantages include compact representation, explicit dependency modeling, integration of prior knowledge with data, and support for probabilistic reasoning (Koller & Friedman, 2009). Their utility spans numerous fields, including diagnostics, bioinformatics, risk assessment, and machine learning. In this thesis, their ability to model task parameter dependencies forms the core of the proposed curriculum generation mechanism.

3.2 Reinforcement Learning

Reinforcement Learning (RL) is a paradigm of machine learning concerned with how intelligent agents ought to take actions in an environment to maximize some notion of cumulative reward. Inspired by behavioral psychology, RL focuses on learning through interaction and feedback, rather than learning from labeled examples (as in supervised learning) or finding hidden structure (as in unsupervised learning).

3.2.1 Core concepts

The standard RL setting involves an *agent* interacting with an *environment* over a sequence of discrete time steps $t = 0, 1, 2, \ldots$. At each time step t:

1. The agent observes the current *state* s_t of the environment.
2. Based on the state, the agent selects an *action* a_t according to its current policy π.
3. The environment transitions to a new state s_{t+1} based on the previous state s_t and the agent's action a_t.
4. The environment provides a scalar *reward* signal r_{t+1} to the agent, indicating the immediate consequence of the action taken in the previous state.

This interaction forms a closed loop, often referred to as the agent-environment interaction loop, illustrated in Figure 3.1. The agent continually perceives the state, acts upon the environment, and receives feedback in the form of rewards and new states. The fundamental goal of the agent is to learn a policy π that maximizes the expected sum of rewards collected over time.

The agent's goal is typically to learn a behavior, known as a *policy*, that maximizes the expected cumulative reward over time.

3.2.2 Markov decision processes (MDPs)

The interaction between the agent and the environment is often formalized using a Markov Decision Process (MDP). An MDP is defined by a tuple (S, A, P, R, γ), where:

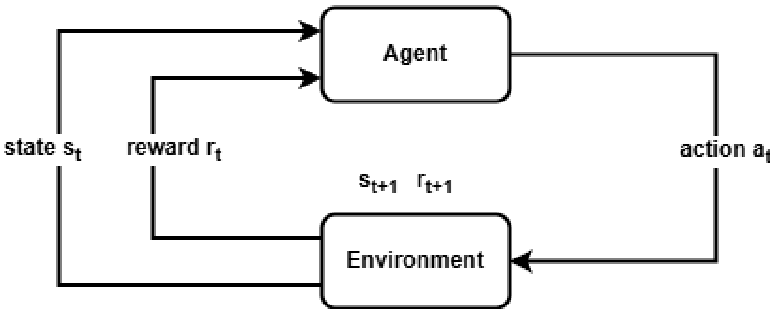


Figure 3.1: The standard agent-environment interaction loop in Reinforcement Learning. The agent observes state s_t, selects action a_t, receives reward r_{t+1} and transitions to state s_{t+1}.

- S: A set of possible states.
- A: A set of possible actions.
- P: The state transition probability function, $P(s'|s,a) = \Pr(s_{t+1} = s'|s_t = s, a_t = a)$.
- R: The reward function, specifying the immediate reward r_{t+1}.
- γ: The discount factor, $\gamma \in [0,1]$.

The key assumption is the *Markov property*: the next state and reward depend only on the current state and action.

3.2.3 Policies and value functions

The agent's behavior is defined by its *policy* π. A stochastic policy $\pi(a|s)$ gives the probability of taking action a in state s. The goal is to find an *optimal policy* π^* maximizing the *expected return* (cumulative discounted reward):

$$G_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \tag{3.7}$$

Value functions estimate the expected return:

- *State-value function* $V^\pi(s) = \mathbb{E}_\pi[G_t|s_t = s]$.
- *Action-value function* $Q^\pi(s,a) = \mathbb{E}_\pi[G_t|s_t = s, a_t = a]$.

Optimal value functions $V^*(s)$ and $Q^*(s,a)$ represent the maximum possible expected return.

3.2.4 Bellman equations

Value functions satisfy recursive Bellman equations. The Bellman optimality equation for V^* is:

$$V^*(s) = \max_a \sum_{s'} P(s'|s,a)[R(s,a,s') + \gamma V^*(s')] \tag{3.8}$$

These equations are fundamental to many RL algorithms.

3.2.5 Exploration vs. exploitation

A key challenge is balancing *exploration* (trying new actions) and *exploitation* (using known good actions) to find the true optimal policy.

3.2.6 Reinforcement learning algorithm categories

RL algorithms can be broadly categorized:

- *Value-Based Methods:* Learn value functions (e.g., Q-learning, DQN).
- *Policy-Based Methods:* Directly learn policies (e.g., REINFORCE).
- *Actor-Critic Methods:* Learn both a policy (actor) and a value function (critic) (e.g., A2C, A3C, PPO, SAC, DDPG, TD3).

3.2.7 Proximal Policy Optimization (PPO)

Proximal Policy Optimization (PPO) (Schulman, Wolski, Dhariwal, Radford, & Klimov, 2017) is a highly effective and widely used policy gradient algorithm within the Actor-Critic family. It has become a default choice for many Deep RL applications due to its favorable balance between sample efficiency, implementation simplicity, and robust performance across a variety of continuous and discrete control tasks. PPO aims to achieve the stability and reliable performance of Trust Region Policy Optimization (TRPO) (Schulman, Levine, Abbeel, Jordan, & Moritz, 2015) while using only first-order optimization, making it simpler to implement and computationally less demanding.

The core idea behind policy gradient methods is to directly optimize the parameters θ of the agent's policy π_θ to maximize the expected return $J(\theta) = \mathbb{E}_{\tau\sim\pi_\theta}[\sum_t \gamma^t r_t]$. The gradient is typically estimated as $\nabla_\theta J(\theta) \approx \hat{\mathbb{E}}_t[\nabla_\theta \log \pi_\theta(a_t|s_t)\hat{A}_t]$, where $\hat{\mathbb{E}}_t$ denotes the empirical average over a batch of state-action pairs collected under the policy, and $\hat{A}_t$ is an estimate of the advantage function at time t. A major challenge with standard policy gradients is that a single large gradient update can drastically change the policy, potentially collapsing performance.

TRPO addresses this by maximizing a surrogate objective function subject to a constraint on the KL divergence between the old and new policies, $D_{KL}(\pi_{\theta_{old}}||\pi_\theta) \leq \delta$, effectively limiting the size of the policy update. However, TRPO involves complex second-order optimization methods.

PPO simplifies this by incorporating the constraint directly into the objective function using either a clipped surrogate objective or a KL penalty. The most common variant uses the clipped objective. Let the probability ratio be $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$, where θ_{old} are the policy parameters before the update. The PPO clipped surrogate objective is:

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t\left[\min\left(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\varepsilon, 1+\varepsilon)\hat{A}_t\right)\right] \tag{3.9}$$

Here, $\hat{A}_t$ is the advantage estimate (often computed using Generalized Advantage Estimation - GAE (Schulman, Moritz, Levine, Jordan, & Abbeel, 2018)), and ε is a small hyperparameter (e.g., 0.2) defining the clipping range. The ‘clip‘ function restricts $r_t(\theta)$ to the interval $[1-\varepsilon, 1+\varepsilon]$. The ‘min‘ operator ensures that the final objective is a lower bound (a pessimistic estimate) of the unclipped objective when the ratio $r_t(\theta)$ would cause a large policy change away from $\pi_{\theta_{old}}$. Specifically:

- If $\hat{A}_t > 0$ (action was better than average), the update is clipped if the new policy becomes too likely ($r_t > 1+\varepsilon$), preventing overly optimistic updates.
- If $\hat{A}_t < 0$ (action was worse than average), the update is clipped if the new policy becomes too unlikely ($r_t < 1-\varepsilon$), preventing overly pessimistic updates that could destabilize learning.

This clipping mechanism effectively discourages large policy updates, promoting stable learning without the computational overhead of TRPO’s explicit trust region constraint.

PPO is typically implemented using an Actor-Critic architecture. The policy π_θ (the Actor) is updated by maximizing $L^{CLIP}(\theta)$ via gradient ascent. A separate value function network $V_\phi(s)$ (the Critic) is trained concurrently, usually by minimizing a squared-error loss between its predictions and empirical targets (e.g., Monte Carlo returns or λ-returns used in GAE): $L^{VF}(\phi) = \hat{\mathbb{E}}_t[(V_t^{\text{target}} - V_\phi(s_t))^2]$. The learned value function V_ϕ is used to compute the advantage estimates $\hat{A}_t$ needed for the policy objective. Often, an entropy bonus term is added to the PPO objective to encourage exploration.

In practice, PPO algorithms collect a batch of trajectories using the policy $\pi_{\theta_{old}}$, compute the advantages $\hat{A}_t$ using these trajectories and the current value function estimate V_ϕ, and then perform multiple epochs of stochastic gradient updates on the combined objective (policy loss + value function loss + entropy bonus) using mini-batches sampled from the collected data before updating θ_{old} to the newly optimized θ.

Given its strong empirical performance, stability, and ease of use, PPO serves as the foundational RL algorithm upon which our BCG framework is built in this thesis.

3.2.8 Challenges

Despite significant progress, RL faces challenges, including:

- *Sample Efficiency:* Learning often requires many interactions.
- *Sparse Rewards:* Difficulty learning when feedback is infrequent.
- *High-Dimensional Spaces:* Handling large state/action spaces (Deep RL helps). * *Partial Observability:* When the true state is not fully perceived.

Curriculum learning, discussed next, is one strategy specifically aimed at addressing challenges like sparse rewards and complex tasks.

3.3 Curriculum Learning

Curriculum Learning (CL) is a training strategy in machine learning inspired by human and animal pedagogy, where learning progresses through stages of increasing difficulty

(Bengio, Louradour, Collobert, & Weston, 2009). Instead of exposing the learner to the full complexity of the target task or dataset from the outset, CL introduces simpler concepts or easier examples first, gradually building up to the more challenging ones. This structured approach aims to improve learning speed, final performance, and generalization. Figure 3.2 provides a general illustration of this concept, showing how a model is trained on progressively larger and harder subsets of data drawn according to a curriculum schedule.

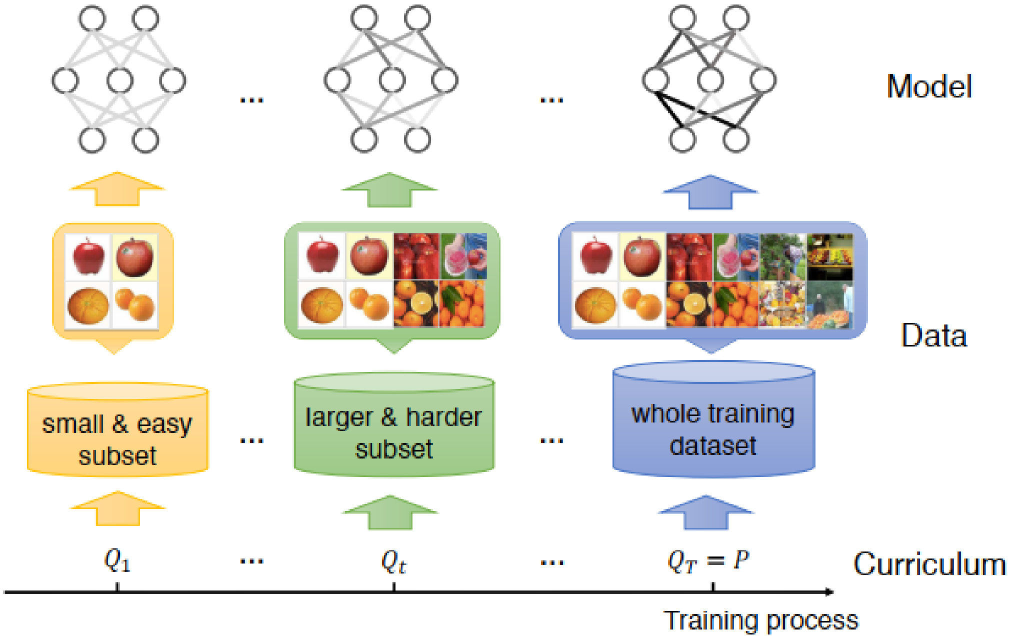


Figure 3.2: Conceptual illustration of Curriculum Learning. The model is initially trained on a small, easy subset of data/tasks (Curriculum stage Q_1), then progresses to larger, harder subsets (Q_t), eventually encompassing the entire target distribution ($Q_T = P$). Image adapted from (Wang, Chen, & Zhu, 2021).

3.3.1 Motivation and application in reinforcement learning

In Reinforcement Learning (RL), CL is particularly valuable for tackling complex problems where standard training might fail or be prohibitively inefficient. These often include scenarios with:

- *Sparse Rewards:* A curriculum can introduce intermediate tasks with denser or shaped rewards, guiding the agent towards the sparsely rewarded goal of the target task.

- *Complex Tasks:* Tasks requiring long sequences of actions or intricate skills can be decomposed into simpler prerequisite tasks.
- *Large State/Action Spaces:* A curriculum can effectively constrain the exploration problem in early stages, focusing learning on essential skills before exposing the agent to the full complexity.

The core idea in RL is to design or automatically generate a sequence of tasks (MDPs), $T_1, T_2, \ldots, T_N$, where T_N is the final target task, such that learning progresses efficiently through the sequence.

3.3.2 Formalizing curriculum components

A curriculum can be formalized by defining its key components:

1. *Task space and distribution:* Let $\mathscr{T}$ be the space of all possible tasks, often defined by a set of parameters $\psi \in \Psi$. The target is often a specific task T_N (with parameters ψ_N) or a target distribution $P(T)$ over $\mathscr{T}$. A curriculum defines a sequence of N intermediate task distributions $Q_1, Q_2, \ldots, Q_N$, where typically $Q_N = P(T)$ or samples heavily around T_N. At stage i, the agent trains on tasks $T \sim Q_i$.
2. *Difficulty measure:* A function $D : \mathscr{T} \to \mathbb{R}$ that assigns a difficulty score to each task T. This score should ideally correlate with the learning challenge posed to the agent. Measures can be based on:
    - Task parameters (e.g., distance to target parameters $D(T_i) = \|\psi_i - \psi_N\|$).
    - Heuristics (e.g., required steps, object counts).
    - Agent's performance (e.g., $1 - \text{SuccessRate}(T_i)$, or $-V^*(T_i)$).
    - Learning dynamics (e.g., learning progress (Graves, Bellemare, Menick, Munos, & Kavukcuoglu, 2017), model uncertainty).

    The curriculum typically orders tasks such that $D(T_i) \leq D(T_{i+1})$ on average for $T_i \sim Q_i, T_{i+1} \sim Q_{i+1}$.
3. *Sequencing strategy / Pacing:* A mechanism determining the progression through the curriculum stages $Q_1, \ldots, Q_N$. This can be:

- Fixed: Predetermined schedule or number of steps per stage.
- Adaptive: Transitioning from stage i to $i+1$ based on the agent's performance. For example, move to stage $i+1$ when the agent's average performance metric $\mathscr{M}_i$ (e.g., mean success rate or reward on tasks from Q_i) exceeds a threshold η_i.

3.3.3 Curriculum learning and transfer learning

A crucial aspect enabling the effectiveness of CL in RL is the implicit use of Transfer Learning between consecutive stages of the curriculum. As the agent masters tasks $T_i \sim Q_i$ at stage i, the knowledge encoded in its learned components (e.g., policy parameters, value function parameters, learned representations) is used to initialize the learning process for tasks $T_{i+1} \sim Q_{i+1}$ at stage $i+1$.

Let π_{θ_i} be the policy with parameters θ_i learned by the end of curriculum stage i. Similarly, let V_{ϕ_i} be a learned value function with parameters ϕ_i. The transfer process can be formalized as initializing the parameters for stage $i+1$ based on the final parameters from stage i:

$$\theta_{i+1}^{\text{initial}} = f(\theta_i^{\text{final}}) \tag{3.10}$$

$$\phi_{i+1}^{\text{initial}} = g(\phi_i^{\text{final}}) \tag{3.11}$$

Often, the transfer functions f and g are simply the identity function (i.e., $\theta_{i+1}^{\text{initial}} = \theta_i^{\text{final}}$), meaning the learning for the next stage starts exactly where the previous stage left off. In other cases, f or g might involve minor modifications, like adding small amounts of noise or freezing certain layers while retraining others.

This transfer of learned parameters is essential. Instead of learning each task T_{i+1} from a random initialization θ^{random}, the agent starts from a point θ_i^{final} that already encodes useful skills and knowledge relevant to the (presumably similar) tasks in Q_{i+1}. The hypothesis is that this warm start significantly reduces the number of samples or training iterations required to achieve proficiency on T_{i+1}, thereby accelerating convergence towards competence on the final target task T_N.

3.3.4 Automation and approaches

While curricula can be manually designed by experts, this requires significant effort and domain knowledge. Research increasingly focuses on Automated Curriculum Learning, where the task generation, difficulty assessment, and sequencing are performed algorithmically. Approaches include:

- *Goal generation:* Methods like Hindsight Experience Replay (HER) implicitly create curricula by treating failed attempts as successes for intermediate goals (Andrychowicz, Wolski, Ray, Schneider, Fong, Welinder, McGrew, Tobin, Abbeel, & Zaremba, 2017). Other methods generate achievable goals based on the agent's current state space coverage.
- *Task parameter sampling:* Explicitly sampling task parameters ψ from distributions Q_i that evolve over time, often guided by agent performance or learning progress (Graves, Bellemare, Menick, Munos, & Kavukcuoglu, 2017). The BCG method in this thesis falls broadly into this category, using BNs to structure the parameter space.
- *Teacher-student frameworks:* An external 'teacher' model learns to propose tasks that are maximally beneficial for the 'student' agent's learning progress (Matiisen, Oliver, Cohen, & Schulman, 2019).
- *Difficulty via self-competition:* Using self-play mechanisms where agents compete against past versions of themselves, naturally creating a curriculum of increasing opponent difficulty (Silver, Schrittwieser, Simonyan, Antonoglou, Huang, Guez, Hubert, Baker, Lai, Bolton, et al., 2017).

3.3.5 Benefits and challenges

When successful, CL offers significant benefits in RL:

- Faster convergence times.
- Improved final performance on the target task.

- Ability to solve complex, sparse-reward tasks intractable otherwise.
- Guided exploration towards relevant parts of the state-action space.

However, effective CL design faces challenges:

- Defining appropriate task spaces and meaningful difficulty metrics.
- Avoiding negative transfer if intermediate tasks are poorly chosen.
- Ensuring the curriculum ultimately leads to good performance on the *target* task.
- Potential computational overhead of managing the curriculum generation and sequencing.

This thesis contributes an automated approach (BCG) aiming to address some of these challenges by using probabilistic models for structured and adaptive curriculum generation.

3.4 Autoencoders

Autoencoders are a type of Artificial Neural Network (ANN) used primarily for unsupervised learning tasks, particularly dimensionality reduction and feature learning (Hinton & Salakhutdinov, 2006). The fundamental idea is to train the network to reconstruct its input, essentially learning an approximation of the identity function, but under constraints that force it to learn a compressed, salient representation of the data in an intermediate layer.

3.4.1 Architecture

An autoencoder consists of two main components connected sequentially: an encoder and a decoder. A typical architecture is visualized in Figure 3.3.

1. *Encoder:* This part of the network takes the input data $x \in \mathbb{R}^d$ and maps it to a hidden representation $z \in \mathbb{R}^{d'}$ in the latent space. Typically, the dimension of the latent space d' is much smaller than the input dimension d ($d' \ll d$), creating an

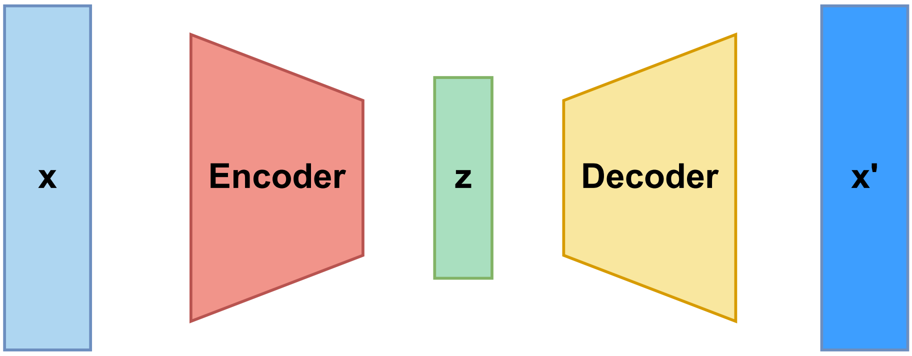


Figure 3.3: Typical architecture of an Autoencoder. The encoder maps the input x to a lower-dimensional latent code z. The decoder attempts to reconstruct the original input x' from the code z. The bottleneck structure forces the network to learn a compressed representation.

information bottleneck. The encoder can be represented as a function $e : \mathbb{R}^d \to \mathbb{R}^{d'}$, parameterized by weights and biases θ_e:

$$z = e(x; \theta_e) \tag{3.12}$$

The encoder often consists of one or more layers (e.g., fully connected, convolutional) with non-linear activation functions (e.g., ReLU, sigmoid).

2. *Decoder:* This part takes the latent representation z and maps it back to a reconstruction $x' \in \mathbb{R}^d$ in the original input space. The goal is for x' to be as close as possible to the original input x. The decoder can be represented as a function $d : \mathbb{R}^{d'} \to \mathbb{R}^d$, parameterized by θ_d:

$$x' = d(z; \theta_d) \tag{3.13}$$

The decoder architecture often mirrors the encoder's structure (e.g., using transposed convolutions if the encoder used convolutions). The activation function of the final decoder layer is chosen based on the nature of the input data (e.g., sigmoid for inputs normalized to [0,1], linear for unbounded real values).

The complete autoencoder function is the composition of the encoder and decoder: $x' = d(e(x; \theta_e); \theta_d)$.

3.4.2 Training objective

Autoencoders are trained by minimizing a loss function that measures the difference between the original input x and its reconstruction x'. This difference is often called the *reconstruction error*. Given a dataset $D = \{x^{(1)}, \dots, x^{(m)}\}$, the objective is to find the parameters (θ_e, θ_d) that minimize the average loss:

$$(\hat{\theta}_e, \hat{\theta}_d) = argmin_{\theta_e, \theta_d} \frac{1}{m} \sum_{j=1}^{m} L(x^{(j)}, d(e(x^{(j)}; \theta_e); \theta_d)) \tag{3.14}$$

The choice of the loss function $L(\cdot, \cdot)$ depends on the data type:

- For real-valued inputs (e.g., normalized image pixel intensities), the *Mean Squared Error (MSE)* is commonly used:

$$L_{MSE}(x, x') = \|x - x'\|_2^2 = \sum_{k=1}^{d} (x_k - x'_k)^2 \tag{3.15}$$

- For binary inputs or inputs interpreted as probabilities (e.g., pixel values in [0,1]), the *Binary Cross-Entropy (BCE)* loss is often preferred:

$$L_{BCE}(x, x') = -\sum_{k=1}^{d} [x_k \log(x'_k) + (1 - x_k) \log(1 - x'_k)] \tag{3.16}$$

  where x'_k is typically the output of a sigmoid activation in the final decoder layer.

Training is performed using standard backpropagation and gradient-based optimization algorithms (like Adam or SGD) to update the parameters θ_e and θ_d.

3.4.3 Use cases and variants

The primary purpose of training an autoencoder is often not the reconstruction x' itself, but the learned latent representation z or the learned encoder function $e(\cdot)$.

- *Dimensionality reduction:* The encoder $e(x)$ provides a lower-dimensional representation z of the input x, capturing its most salient features.

- *Feature learning:* The latent code z can be used as a learned feature vector for input into subsequent supervised learning models (e.g., classifiers, regressors) or, as in this thesis, for analyzing task characteristics.

- *Data denoising:* Denoising Autoencoders are trained to reconstruct clean inputs from corrupted versions, forcing them to learn robust features.
- *Generative modeling:* Variational Autoencoders (VAEs) (Kingma & Welling, 2014) are a probabilistic extension that learns a distribution over the latent space, allowing for the generation of new data samples similar to the training data.

Other variants include Sparse Autoencoders (which add sparsity penalties to the latent code) and Convolutional Autoencoders (which use convolutional layers, suitable for image data).

In the context of this thesis, we utilize the encoder part $e(\cdot)$ of a trained autoencoder (specifically, a convolutional autoencoder for visual inputs like MiniGrid states) to extract a compressed feature vector $z = e(x)$ representing the essential visual characteristics of a given task's state or configuration. This learned representation z is then used as input for downstream analysis, such as measuring task similarity or difficulty within the Curriculum Learning framework.

4. METHODOLOGY

4.1 Problem Formulation

We address curriculum learning within reinforcement learning (RL) environments, particularly those characterized by sparse rewards. The standard RL problem is modeled as a Markov Decision Process (MDP), defined by the tuple (S, A, P, R), where:

- S represents the state space: all possible configurations of the environment.
- A denotes the action space: all actions available to the agent.
- $P : S \times A \times S \to [0,1]$ is the state transition function, giving the probability $P(s'|s,a)$ of transitioning to state s' after taking action a in state s.
- $R : S \times A \to \mathbb{R}$ is the reward function, assigning a scalar reward to state-action pairs.

The agent's objective is to learn an optimal policy π^* that maximizes the expected discounted cumulative reward, represented by the value function:

$$V^{\pi}(s) = \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \mid s_0 = s\right], \tag{4.1}$$

where $\gamma \in [0,1)$ is the discount factor. The optimal policy π^* and its corresponding optimal value function V^* satisfy:

$$\pi^* = \arg\max_{\pi} V^{\pi}(s), \quad \forall s \in S, \tag{4.2}$$

$$V^*(s) = \max_{\pi} V^{\pi}(s) = V^{\pi^*}(s), \quad \forall s \in S. \tag{4.3}$$

In the context of curriculum learning, this MDP formulation is extended. A curriculum is defined as a sequence of related tasks $[T_1, T_2, \ldots, T_n]$, where each task T_i is an MDP, often sharing structure but varying in difficulty (e.g., different parameters within the same environment). The agent learns these tasks sequentially. A crucial aspect is transfer learning: the policy learned for task T_i is often used to initialize the learning process for the subsequent task T_{i+1}, aiming to accelerate convergence towards the final target task T_n. Our work focuses on generating this sequence $T_1,\ldots,T_n$ automatically and adaptively.

4.2 Bayesian Curriculum Learning Framework

This research proposes a novel Bayesian Curriculum Generation (BCG) (Akgün & Üre, 2025) approach for RL in sparse reward settings. The core idea is to integrate probabilistic modeling, unsupervised clustering, and adaptive task sequencing based on agent performance. This framework aims to create structured, efficient, and adaptive curricula tailored to the agent's learning progress. The complete algorithm is detailed in Section 4.3.

4.2.1 Probabilistic task modeling with bayesian networks

We utilize Bayesian Networks (BNs) to model the relationships and dependencies between key parameters defining the tasks within an environment (Koller & Friedman, 2009; Pearl, 1988). A BN is a directed acyclic graph (DAG) $G = (V, E)$, where nodes $V = \{V_1, \ldots, V_n\}$ represent environmental parameters (e.g., agent/goal positions, object properties) and edges E represent conditional dependencies. The structure allows factorization of the joint probability distribution:

$$P(V_1, \ldots, V_n) = \prod_{i=1}^{n} P(V_i \mid \mathrm{Pa}(V_i)), \tag{4.4}$$

where $\mathrm{Pa}(V_i)$ are the parents of node V_i. Each Conditional Probability Distribution (CPD), $P(V_i \mid \mathrm{Pa}(V_i))$, often denoted as $\theta_{ijk} = P(V_i = k \mid \mathrm{Pa}(V_i) = j)$, quantifies the influence of parent parameters on a child parameter. This probabilistic model serves as the basis for generating diverse yet structured tasks by sampling parameter configurations according to the learned dependencies. Tasks are generated by sampling parameters following the topological order of the BN.

4.2.2 Task representation and difficulty evaluation

Effective curriculum design requires quantifying task characteristics and difficulty. Tasks $\mathscr{T} = \{T_1, \ldots, T_n\}$ are mapped into a feature space using an embedding function ϕ.

- *Visual tasks:* For tasks represented by images (e.g., grid environments like MiniGrid (Chevalier-Boisvert, Dai, Towers, Perez-Vicente, Willems, Lahlou, Pal, Castro, & Terry, 2023)), an autoencoder $\mathscr{A}$ extracts a latent representation $z_i = \mathscr{A}(T_i)$. Dimensionality reduction $\mathscr{D}$ (e.g., t-SNE) may be applied subsequently: $y_i = \mathscr{D}(z_i)$.
- *Scalar tasks:* Tasks defined by scalar parameters (e.g., velocities in AeroRival Pursuit) are represented by their normalized parameter vectors.

The difficulty of a task T_i is initially assessed based on its distance to the final target task T_n in the feature space:

$$d(T_i) = \|\phi(T_i) - \phi(T_n)\|_2. \tag{4.5}$$

These distances are then normalized and potentially combined (using weights λ_1, λ_2) if multiple representations exist (e.g., visual and scalar), resulting in normalized difficulty scores D_i'.

4.2.3 Clustering and adaptive task sequencing

Based on the normalized difficulty scores D_i', tasks are grouped into N_{bins} distinct difficulty levels or clusters $\{C_1, \ldots, C_{N_{bins}}\}$ using an unsupervised clustering algorithm (e.g., K-means). This ensures tasks within a cluster are of similar difficulty and that clusters are ordered, forming the backbone of the curriculum sequence. Task selection during training is dynamic and probabilistic, often guided by the agent's performance $\mathscr{M}_t$ (e.g., mean episodic reward, success rate). The selection mechanism (detailed in Eq. 4.16) typically favors tasks that are appropriately challenging for the agent's current skill level. Furthermore, the framework adapts based on performance.

Task difficulty assessments and the underlying BN model can be updated using Bayesian inference (Eq. 4.18, 4.19, 4.20) incorporating observed agent performance data. Task solvability $\sigma(T_i)$ is evaluated against a threshold η (Eq. 4.17, 4.21), potentially triggering adjustments to the perceived difficulty of related tasks (Eq. 4.22, 4.23). Training transitions to the target task T_n once the agent demonstrates sufficient competence on prerequisite tasks, optimizing training time.

4.3 Bayesian Curriculum Generation (BCG) Algorithm

The BCG algorithm (Akgün & Üre, 2025), detailed in this section, operationalizes the framework described above. It systematically integrates task representation, difficulty assessment via clustering, probabilistic task generation using BNs, and dynamic adaptation based on agent performance. Figure 4.1 provides a visual overview of the workflow.

The algorithm proceeds through the following key steps:

4.3.1 Task representation processing

Tasks $\mathscr{T} = \{T_1, \ldots, T_n\}$ are mapped to a feature space via ϕ.

- *Visual tasks:* Use an autoencoder $\mathscr{A}$ for latent features $z_i = \mathscr{A}(T_i) \in \mathbb{R}^m$ (Eq. 4.6), potentially followed by dimensionality reduction $\mathscr{D}$ to get $y_i = \mathscr{D}(z_i) \in \mathbb{R}^k$ (Eq. 4.7).
- *Scalar tasks:* Use normalized parameter vectors directly via ϕ.

$$z_i = \mathscr{A}(T_i) \tag{4.6}$$

$$y_i = \mathscr{D}(z_i) \tag{4.7}$$

4.3.2 Difficulty assessment and normalization

Compute base difficulty relative to the target task T_n:

$$d(T_i) = \|\phi(T_i) - \phi(T_n)\|_2. \tag{4.8}$$

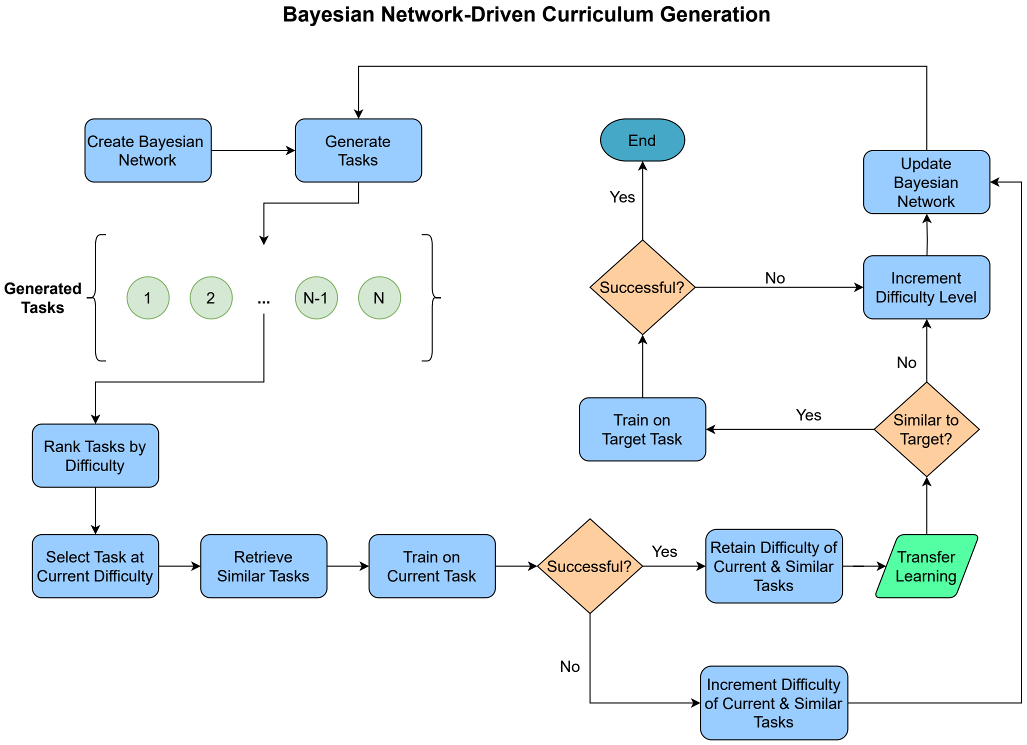


Figure 4.1: Overview of the Bayesian Curriculum Generation (BCG) Algorithm (adapted from (Akgün & Üre, 2025)): This flowchart illustrates the structured approach. It starts with building a Bayesian Network for curriculum modeling. Unsupervised clustering differentiates task complexity levels, and a cost function guides task sequencing. The curriculum evolves dynamically through iterative cycles of task generation, difficulty evaluation, and agent training, repeating until the target task is mastered. This enhances sample efficiency and reduces training burden.

Calculate a normalized similarity score $s_{norm}(T_i)$, potentially combining different feature types (e.g., encoded vs. direct parameters) using weights λ_1, λ_2:

$$s_{norm}(T_i) = \begin{cases} \lambda_1 \hat{s}_E + \lambda_2 (1 - \hat{s}_S), & \text{if encoder } \mathscr{A} \text{ is used} \\ 1 - \hat{s}_S, & \text{otherwise} \end{cases} \tag{4.9}$$

where $\hat{s}_E$ and $\hat{s}_S$ are normalized distances based on encoded features ($\mathscr{E} = \mathscr{D} \circ \mathscr{A}$) and direct features ($\phi$) respectively:

$$\hat{s}_E = \frac{\|\mathscr{E}(T_i) - \mathscr{E}(T_n)\|_2}{\max_j \|\mathscr{E}(T_j) - \mathscr{E}(T_n)\|_2}, \tag{4.10}$$

$$\hat{s}_S = \frac{\|\phi(T_i) - \phi(T_n)\|_2}{\max_j \|\phi(T_j) - \phi(T_n)\|_2}. \tag{4.11}$$

Derive the final normalized difficulty score D'_i using min-max scaling:

$$D'_i = \frac{d(T_i) - \min_j(d(T_j))}{\max_j(d(T_j)) - \min_j(d(T_j))}. \tag{4.12}$$

4.3.3 Difficulty clustering and task selection

Cluster tasks into $K = N_{bins}$ ordered difficulty groups $\{C_1, \ldots, C_K\}$ based on D'_i using K-means, such that:

$$\bigcup_{k=1}^{K} C_k = \mathscr{T}, \tag{4.13}$$

$$C_k \cap C_j = \emptyset \quad \forall k \neq j, \tag{4.14}$$

$$\forall T_i \in C_k, T_j \in C_{k+1} : \text{mean}(D'_{T \in C_k}) \leq \text{mean}(D'_{T \in C_{k+1}}). \tag{4.15}$$

Select the next task T_i based on its posterior probability given agent performance metrics $\mathscr{M}_t$:

$$P(T_i | \mathscr{M}_t) = \frac{P(\mathscr{M}_t | T_i) P(T_i)}{\sum_{j=1}^{n} P(\mathscr{M}_t | T_j) P(T_j)}, \tag{4.16}$$

where performance metrics $\mathscr{M}_t$ might include mean reward R_t and episode length l_t:

$$\mathscr{M}_t = \left\{ R_t = \frac{1}{N} \sum_{e=1}^{N} r_e, l_t = \frac{1}{N} \sum_{e=1}^{N} len_e \right\}. \tag{4.17}$$

4.3.4 Probabilistic task generation and bayesian updating

Leverage the BN $G=(V,E)$ to generate new task parameters by sampling from the CPDs $P(V_i|\mathrm{Pa}(V_i))$. Update the BN parameters (θ) based on observed agent performance data $\mathscr{D}_{\text{new}}$ using Bayesian inference:

$$P(\theta|\mathscr{D}_{\text{new}}) \propto P(\mathscr{D}_{\text{new}}|\theta)P(\theta) \propto P(\mathscr{D}_{\text{new}}|\theta)\prod_{i,j,k}\theta_{ijk}^{\alpha_{ijk}-1}. \tag{4.18}$$

Refine CPDs using methods like Maximum Likelihood Estimation (MLE):

$$P^*(V_i|\mathrm{Pa}(V_i)) = \arg\max_P \sum_{j=1}^{N} \log P(O_j,T_j|V_i,\mathrm{Pa}(V_i),G), \tag{4.19}$$

or Maximum A Posteriori (MAP) estimation:

$$P^*(V_i|\mathrm{Pa}(V_i)) = \arg\max_P \left(\sum_{j=1}^{N} \log P(O_j,T_j|V_i,\mathrm{Pa}(V_i),G)\right) + \log P(V_i|\mathrm{Pa}(V_i)). \tag{4.20}$$

4.3.5 Dynamic difficulty adjustment and thresholding

Assess task solvability $\sigma(T_i)$ using a performance threshold η:

$$\sigma(T_i) = (R_t \geq \eta \cdot R_{\max}(T_i)), \tag{4.21}$$

where R_t is the recent average reward on T_i and $R_{\max}(T_i)$ is the estimated maximum possible reward. Adjust the difficulty level $l(T_j)$ of tasks T_j similar to T_i based on whether T_i was solved, using a similarity threshold τ_t:

$$l'(T_j) = \begin{cases} l(T_j) & \text{if } s_{norm}(T_j,T_i) > \tau_t \wedge \sigma(T_i) \\ \min(l(T_j)+1, N_{bins}-1) & \text{if } s_{norm}(T_j,T_i) > \tau_t \wedge \neg\sigma(T_i) \\ l(T_j) & \text{otherwise} \end{cases} \tag{4.22}$$

The similarity threshold τ_t itself can be adaptive, e.g., based on the mean similarity $\mu_{s_{norm}}$ scaled by a factor β:

$$\tau_t = \mu_{s_{norm}} \cdot \beta. \tag{4.23}$$

This iterative process of generation, training, evaluation, and adaptation continues until the target task T_n is mastered.

4.3.6 Computational complexity analysis

The overall computational complexity of training an RL agent with BCG is the sum of the complexity of the underlying RL algorithm (e.g., PPO) and the overhead introduced by the BCG framework. The goal of BCG is that its overhead is significantly offset by a reduction in the sample complexity required by the RL agent to solve the tasks. The BCG overhead consists of three main components:

1. *Bayesian Network Learning (Offline Cost):* The initial learning of the Bayesian Network's Conditional Probability Distributions (CPDs) from data is an offline, one-time cost. If learning from a dataset of M tasks with N parameters, the complexity depends on the chosen structure learning and parameter learning algorithms. For a fixed BN structure, learning the parameters is generally efficient. However, structure learning can be computationally intensive, though this is performed only once before the main training loop.

2. *Bayesian Network Inference (Online Cost):* This is the most significant part of BCG's online overhead, performed periodically during RL training. The complexity of exact probabilistic inference in a BN is, in the worst case, exponential in the network's *treewidth*. The treewidth is a measure of a graph's structural complexity.

    - For simple structures like chains (`A -> B -> C`) or trees, the treewidth is low (1 or 2), and inference is linear in the number of nodes, making it very fast.
    - For densely connected graphs, where nodes have many parents, the treewidth can be large, leading to exponential complexity. This is the core of the trade-off mentioned in our Limitations (Section 6.5): a manually designed, simple BN structure is computationally cheap, while a more complex, highly-connected structure is expensive.

Therefore, the inference complexity can be expressed as $O(\exp(w))$, where w is the treewidth of the manually defined BN. This cost is incurred each time the curriculum is updated.

3. *Clustering and Task Selection (Online Cost):* After inferring difficulties for a pool of T candidate tasks, BCG performs clustering. Using an efficient algorithm like k-means on a one-dimensional difficulty score is highly efficient, typically with a complexity of $O(T \cdot k \cdot i)$, where k is the number of clusters (bins) and i is the number of iterations. This cost is generally negligible compared to BN inference and RL training.

In summary, the primary computational consideration for BCG is the complexity of inference in the user-defined Bayesian Network. By encouraging the use of causally-inspired, sparse network structures, the online overhead can be kept minimal, allowing the gains in RL sample efficiency to dominate the overall training time.

4.4 Hyperparameter Analysis and Configuration

The operational characteristics of the BCG framework are influenced by several key hyperparameters, summarized in Table 4.1. A detailed parametric study was conducted to understand their impact. Our findings suggest considerable stability, with overall performance metrics typically varying by only 5-10% when parameters are adjusted within their recommended ranges under the evaluated environmental conditions.

This observed robustness likely stems from several integral design aspects of the framework:

- The adaptive nature of the similarity threshold (τ_t), which self-calibrates based on observed task feature distances.
- The use of unsupervised clustering (N_{bins}) to group tasks by difficulty, providing inherent structure regardless of the precise bin count.
- The probabilistic task selection mechanism, which naturally balances exploring different task types and exploiting successful ones.

While the default values presented proved effective in our experiments on environments with predefined parameters, scenarios featuring greater complexity, dynamic elements, or temporal variations might require a more extensive hyperparameter search. Nonetheless, for settings similar to those tested, the values in Table 4.1 offer effective starting points. The following analysis explores key hyperparameter interactions visualized via heatmaps.

Table 4.1: BCG Framework Hyperparameters: Settings and Effects (adapted from (Akgün & Üre, 2025)).

Parameter	**Default value**	**Recommended range**	**Effect on curriculum**
β	0.4	[0.3, 0.5]	Influences the adaptiveness of difficulty adjustments based on task similarity; higher values make adjustments affect more similar tasks (related to τ_t).
N_{bins}	3	[2, 5]	Sets the number of distinct difficulty levels (clusters). More bins allow finer-grained progression but can potentially slow learning if tasks within bins are too similar or if too many stages are created.
λ_1, λ_2	0.6, 0.4	[0.5, 0.7], [0.3, 0.5]	Balances the contribution of different feature types in similarity calculations: λ_1 weights autoencoder-derived features, λ_2 weights state-based features. Values >0.5 indicate stronger reliance on the corresponding feature type.
η	0.7	[0.6, 0.9]	Defines the performance threshold for considering a task 'solved' or mastered. Higher values demand greater proficiency before progression or confirming task difficulty.

For the heatmap analysis, performance was quantified using a combined metric giving equal importance (50% weight each) to task success (measured by normalized reward) and learning efficiency (measured by normalized training timesteps required).

The interplay between hyperparameters is revealed in the heatmaps. Figure 4.2 examines the relationship between the number of difficulty clusters (N_{bins}) and the

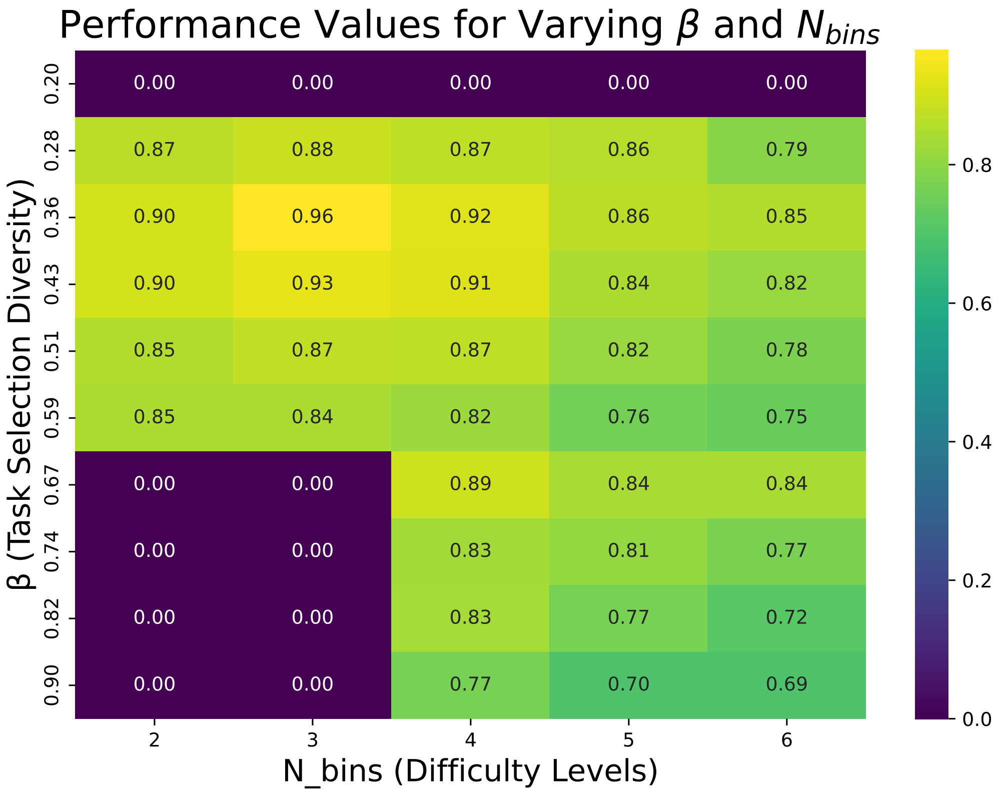
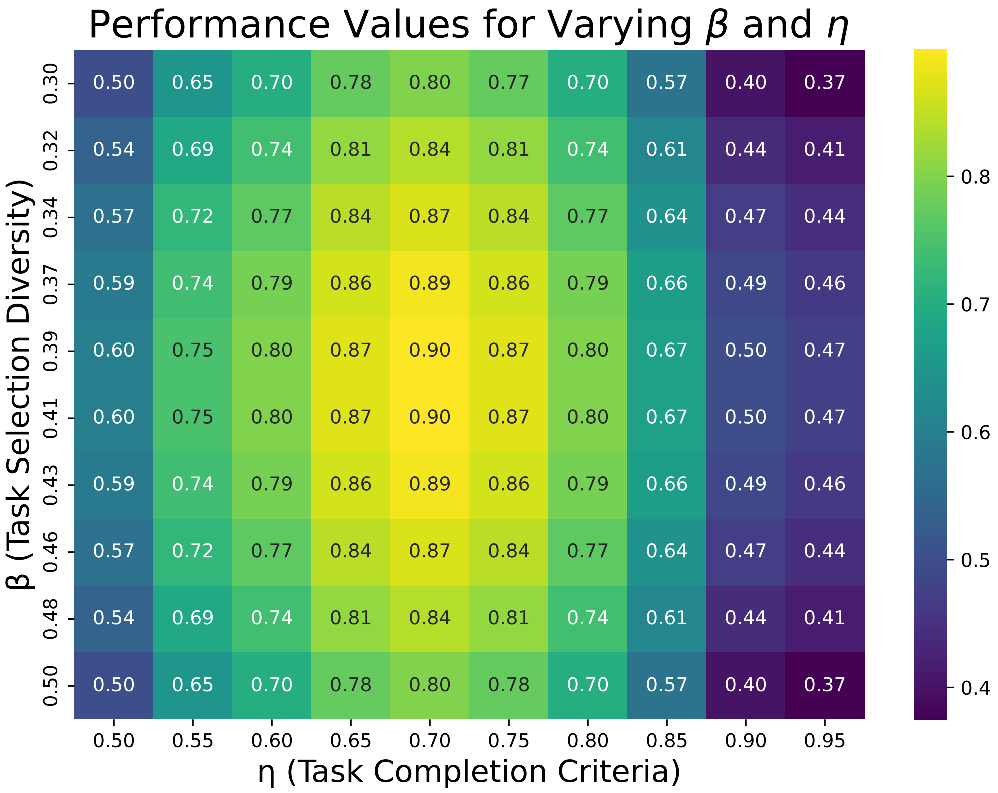


Figure 4.2: Performance variation with β and N_{bins}. Warmer colors indicate higher combined performance (adapted from (Akgün & Üre, 2025)).



	0.50	0.55	0.60	0.65	0.70	0.75	0.80	0.85	0.90	0.95
0.30	0.50	0.65	0.70	0.78	0.80	0.77	0.70	0.57	0.40	0.37
0.32	0.54	0.69	0.74	0.81	0.84	0.81	0.74	0.61	0.44	0.41
0.34	0.57	0.72	0.77	0.84	0.87	0.84	0.77	0.64	0.47	0.44
0.37	0.59	0.74	0.79	0.86	0.89	0.86	0.79	0.66	0.49	0.46
0.39	0.60	0.75	0.80	0.87	0.90	0.87	0.80	0.67	0.50	0.47
0.41	0.60	0.75	0.80	0.87	0.90	0.87	0.80	0.67	0.50	0.47
0.43	0.59	0.74	0.79	0.86	0.89	0.86	0.79	0.66	0.49	0.46
0.46	0.57	0.72	0.77	0.84	0.87	0.84	0.77	0.64	0.47	0.44
0.48	0.54	0.69	0.74	0.81	0.84	0.81	0.74	0.61	0.44	0.41
0.50	0.50	0.65	0.70	0.78	0.80	0.78	0.70	0.57	0.40	0.37

Figure 4.3: Performance variation with η and β. Warmer colors indicate higher combined performance (adapted from (Akgün & Üre, 2025)).

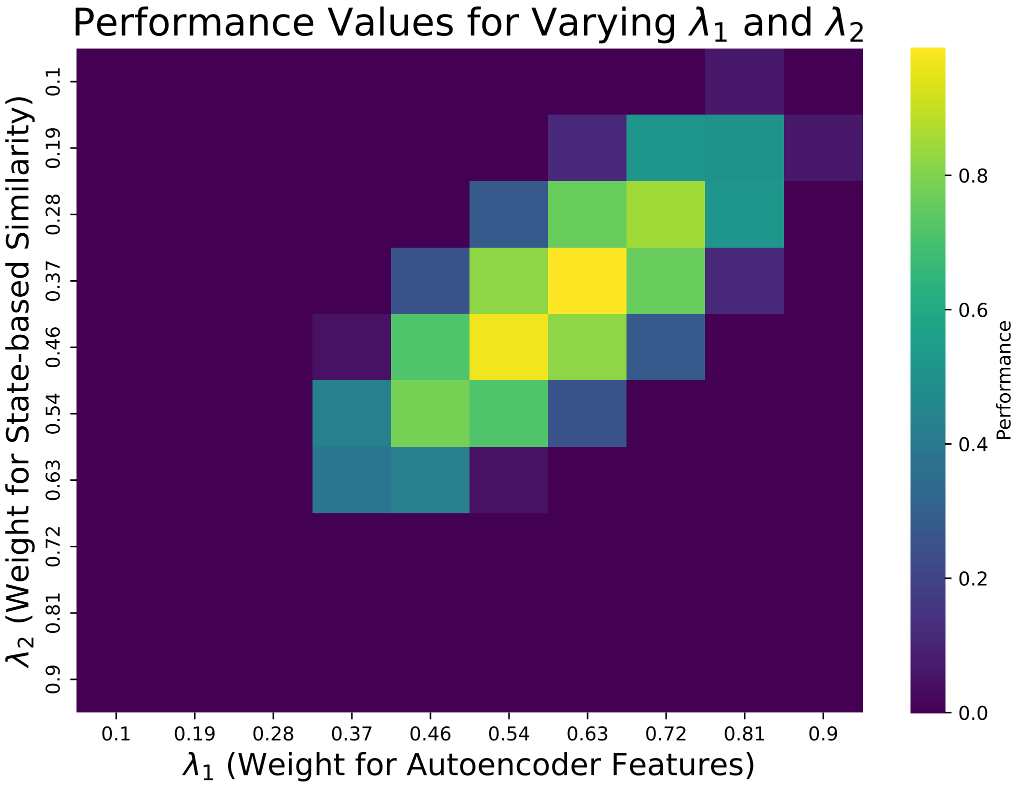


Figure 4.4: Performance variation with λ_1 and λ_2. Warmer colors indicate higher combined performance (adapted from (Akgün & Üre, 2025)).

similarity threshold factor (β). Optimal performance (score 0.96) was observed around $N_{bins} = 3$ and $\beta = 0.43$, suggesting a balance between moderate difficulty granularity and adaptive similarity assessment is beneficial. The framework shows robustness, maintaining scores above 0.80 for β between approximately 0.28 and 0.51. Performance declines sharply outside this band, especially when coarse granularity ($N_{bins} = 2$) combines with high β (>0.67), leading to failure. Conversely, very fine granularity ($N_{bins} \geq 5$) tended to slow progress, yielding lower scores than intermediate values ($N_{bins} = 3$ or 4). This highlights the need for appropriately scaled difficulty levels and task diversity control.

Figure 4.3 illustrates the interaction between the task completion threshold (η) and β. The highest performance (around 0.90) occurs in a region where β is near 0.4 (specifically 0.39–0.41) and η is approximately 0.70. This peak underscores the value of coupling moderate task selection diversity with a reasonably demanding success criterion. Performance tends to decrease symmetrically as β moves away from 0.4. Notably, very high η values (>0.85) significantly reduce performance irrespective of β, likely because excessively strict completion criteria hinder the agent's progression through the curriculum. Similarly, low η values (<0.5) allow agents to advance prematurely with insufficient skill mastery, which also leads to lower overall scores due to reduced final task success and inefficient learning. This emphasizes the sensitivity of the curriculum's effectiveness to the chosen success threshold.

The relationship between the weights for different similarity measures, λ_1 (autoencoder features) and λ_2 (state-based features), is shown in Figure 4.4. A diagonal band indicates strong performance when both measures contribute. The optimum lies near $\lambda_1 \approx 0.63$ and $\lambda_2 \approx 0.37$, suggesting a slight preference for using the abstract features learned by the autoencoder but confirming that incorporating state-based similarity is also crucial. Relying too heavily on only one measure (e.g., $\lambda_1 > 0.81$ or $\lambda_2 > 0.63$) leads to a marked decrease in performance, underlining the benefit of a combined approach for robust task representation in the curriculum.

In conclusion, this parametric analysis indicates that the BCG framework exhibits reliable performance within the recommended hyperparameter ranges for the tested environments, with the default settings providing a solid baseline. The heatmap visualizations offer valuable insights into parameter interactions, consistently pointing towards the benefits of balanced settings for granularity (N_{bins}), task selection/adaptation (β), success criteria (η), and feature weighting (λ_1, λ_2). These findings establish a foundation for applying the framework while suggesting areas for careful tuning when extending to more complex or dynamic scenarios.

4.5 Implementation Details

This section outlines the technical specifics of the framework's implementation, including the software environment, computational hardware, and the architecture used for processing visual task representations.

4.5.1 Software and hardware environment

The development and execution of the experiments were carried out using Python version 3.8. Core numerical operations and data handling leveraged NumPy (v1.21.0) and Pandas (v1.1.5). The deep learning components, including the base reinforcement learning agent and the autoencoder, were implemented using PyTorch (v1.10.2), utilizing its ‘torch.nn‘ module for network layers and ‘torch.optim‘ for optimization. Image preprocessing steps potentially involved torchvision (v0.11.3).

For constructing and manipulating Bayesian Networks, the ‘pgmpy‘ library (v0.1.24) was employed, specifically using its ‘BayesianNetwork‘ module for structure definition and learning, alongside its parameter estimation (MLE) and inference functionalities. Dimensionality reduction of latent representations was performed using the ‘openTSNE‘ library (v0.6.0), noted for its efficient FFT-accelerated implementation.

Clustering of tasks based on their representations utilized algorithms available in ‘scikit-learn‘ (v0.24.2), such as KMeans and potentially DBSCAN, along with its ‘StandardScaler‘ for feature normalization. Additional scientific computing tasks were supported by ‘scipy‘ (v1.5.4). Visualizations were generated using Matplotlib (v3.3.4), and experiment progress was monitored with ‘tqdm‘ (v4.64.1).

All computational experiments were performed on a high-performance computing cluster equipped with NVIDIA V100 GPUs, leveraging CUDA 11.3 for accelerated computation. The project's source code was managed using Git for version control, and code correctness was ensured through unit tests developed with 'pytest' (v6.2.5).

4.5.2 Visual task representation: autoencoder implementation

To handle visual inputs, such as the grid representations in the MiniGrid environment, an autoencoder was implemented to learn compressed latent features representative of the task's visual state. As introduced in Section 3.4, autoencoders learn to reconstruct their input through an encoder-decoder architecture, forcing data through an information bottleneck. This process facilitates the extraction of salient features for downstream analysis, such as assessing task difficulty within the BCG framework.

The specific architecture employed was a deep convolutional autoencoder, detailed in Table 4.2. The encoder component progressively reduces the spatial dimensions of the input image (assumed to be 224x224x3) while increasing the feature depth using a sequence of 2D convolutions (with ReLU activations and Batch Normalization) and MaxPooling layers. This results in a compact latent representation (7x7x512 in this specific architecture). The decoder component then reconstructs the image back to its original dimensions using corresponding ConvTranspose2D layers (also with ReLU/BatchNorm), culminating in a final Sigmoid activation to output pixel values typically normalized between 0 and 1. Figure 4.5 illustrates this architectural concept.

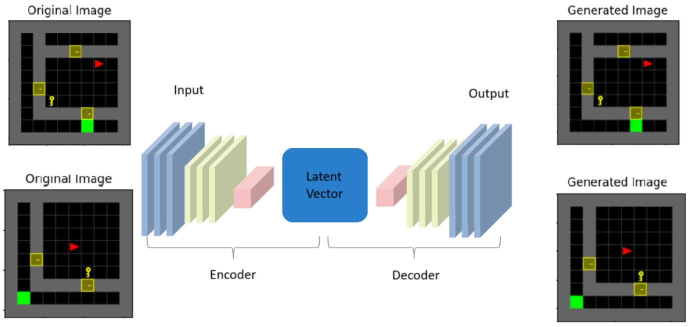


Figure 4.5: Conceptual illustration of the Convolutional Autoencoder architecture used for processing visual task inputs (e.g., from MiniGrid). Input images are compressed by the encoder into a latent feature space and subsequently reconstructed by the decoder (adapted from (Akgün & Üre, 2025)).

The autoencoder network was trained by minimizing the reconstruction error between the input task images and the decoder's output, typically using a loss function like Mean Squared Error (MSE) or Binary Cross-Entropy (BCE), optimized via stochastic gradient descent methods (e.g., Adam).

Following training, only the encoder part of the network (f in the background section's notation) was utilized during the curriculum generation phase. The encoder transforms a given visual task input x into its corresponding latent vector $z = f(x)$. These high-dimensional latent vectors, capturing essential task features, were then processed further. Specifically, to facilitate visualization and clustering based on task similarity, the t-SNE algorithm was applied to reduce the dimensionality of the latent vectors z down to a two-dimensional representation, aiming to preserve the local structure and reveal inherent groupings within the task space.

4.5.3 Dimensionality reduction for visualization (t-SNE)

To analyze and visualize the relationships between tasks based on the high-dimensional latent vectors (z) generated by the autoencoder's encoder component, we employed t-Distributed Stochastic Neighbor Embedding (t-SNE) (van der Maaten & Hinton, 2008). t-SNE is a non-linear technique particularly effective at revealing local structure and similarities within high-dimensional data by embedding it into a low-dimensional space, typically 2D for visualization.

Table 4.2: Detailed Architecture of the Deep Convolutional Autoencoder Network. The table specifies the layer type, output shape, spatial dimensions (HxW), feature depth, and filter size for each step in the encoder (compression) and decoder (reconstruction) paths (adapted from (Akgün & Üre, 2025)).

Block	**Operation**	**Output Shape**	**HxW**	**Depth**	**Filter HxW**
Encoder	Input	224x224x3	224	3	-
Encoder	Conv2D + ReLU + BatchNorm	224x224x32	224	32	3x3
Encoder	MaxPool2D	112x112x32	112	32	2x2
Encoder	Conv2D + ReLU + BatchNorm	112x112x64	112	64	3x3
Encoder	MaxPool2D	56x56x64	56	64	2x2
Encoder	Conv2D + ReLU + BatchNorm	56x56x128	56	128	3x3
Encoder	MaxPool2D	28x28x128	28	128	2x2
Encoder	Conv2D + ReLU + BatchNorm	28x28x256	28	256	3x3
Encoder	MaxPool2D	14x14x256	14	256	2x2
Encoder	Conv2D + ReLU + BatchNorm	14x14x512	14	512	3x3
Encoder	MaxPool2D	7x7x512	7	512	2x2
Decoder	ConvTranspose2D + ReLU + BatchNorm	14x14x256	14	256	2x2
Decoder	ConvTranspose2D + ReLU + BatchNorm	28x28x128	28	128	2x2
Decoder	ConvTranspose2D + ReLU + BatchNorm	56x56x64	56	64	2x2
Decoder	ConvTranspose2D + ReLU + BatchNorm	112x112x32	112	32	2x2
Decoder	ConvTranspose2D + Sigmoid	224x224x3	224	3	2x2

The ‘openTSNE‘ library (v0.6.0), noted for its efficient implementation, was utilized for this purpose. t-SNE computes pairwise similarities between the high-dimensional latent vectors (modeling them with a Gaussian distribution) and attempts to find a low-dimensional embedding where pairwise similarities (modeled with a Student’s t-distribution) closely match, minimizing the Kullback-Leibler divergence between the two distributions. A key parameter in t-SNE is perplexity, which influences the balance between preserving local versus global aspects of the data structure in the resulting embedding. Tuning perplexity (within ranges like 50-150, depending on dataset size) helps achieve meaningful visualizations where clusters represent groups of tasks deemed similar by the learned latent features. Figure 4.6 provides an example of such a 2D visualization obtained by applying t-SNE to the encoder outputs for a set of tasks.

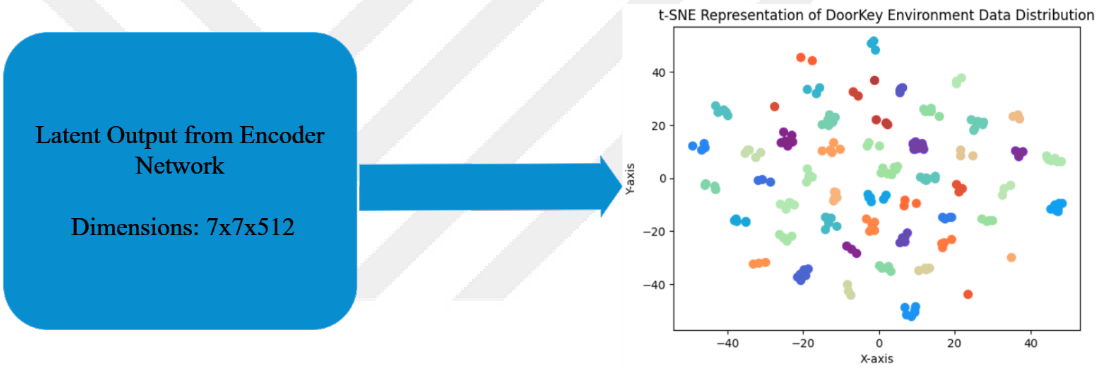


Figure 4.6: Example 2D visualization produced by applying t-SNE to the latent features generated by the autoencoder’s encoder network for various tasks. Each point corresponds to a task, and proximity suggests similarity in the learned latent space (adapted from (Akgün & Üre, 2025)).

This low-dimensional embedding serves primarily as a tool for visual inspection and qualitative analysis of the learned task representations, helping to confirm whether the autoencoder captures meaningful differences and similarities relevant to task difficulty. The quantitative difficulty assessment, however, was based on distances calculated from these embeddings as described next.

4.5.4 Visual difficulty assessment pipeline

For tasks represented by visual inputs (e.g., images I_i from MiniGrid), the implementation of the difficulty assessment (conceptually described in Section 4.2 and with formulas in Section 4.2.2) followed a specific pipeline utilizing the trained autoencoder and the subsequent t-SNE embedding:

1. *Encoding:* Each task image I_i in the set of considered tasks $\mathscr{T}$ (including the target task I_{target}) was passed through the pre-trained encoder network $\mathscr{E}$ (detailed in Section 4.5.2) to obtain its corresponding high-dimensional latent vector $L_i = \mathscr{E}(I_i)$.

2. *Dimensionality reduction:* The set of all obtained latent vectors $\{L_1, L_2, \ldots, L_n\}$ was processed using the t-SNE algorithm (implemented via 'openTSNE') to generate a 2D representation $\mathscr{T}' = \{y_1, y_2, \ldots, y_n\}$, where $y_i = \text{t-SNE}(L_i)$.

3. *Distance calculation:* The Euclidean distance (L_2 norm) was calculated in the 2D t-SNE space between the representation of each task y_i and the representation of the target task y_{target}. Let this distance be $d_i = \|y_i - y_{\text{target}}\|_2$. This distance served as the raw difficulty score relative to the target.

4. *Normalization:* These raw distances d_i were then normalized across all tasks to produce the final difficulty scores δ_i ranging from 0 to 1, using min-max scaling as defined previously in the methodology (specifically, $\delta_i = (d_i - \min(d))/(\max(d) - \min(d))$). A lower δ_i indicates higher similarity (lower difficulty relative) to the target task in this embedded space.

This pipeline provided a practical method to derive quantitative difficulty scores from complex visual inputs by leveraging learned features and dimensionality reduction, which were then used for clustering and task selection within the BCG algorithm.

4.5.5 Scalar difficulty assessment pipeline

For environments like AeroRival Pursuit where tasks are primarily defined by continuous or scalar parameters rather than visual inputs, a different pipeline was implemented to determine task difficulty relative to the target task ($\mathscr{T}_{\text{target}}$), following the principles outlined in Section 4.2 and using the distance/normalization formulas defined in Section 4.2.2.

Since image processing via autoencoders is not required, the focus shifts directly to the task-defining parameters (e.g., enemy velocity, rocket velocity, target distances). Let the set of relevant parameters for a task $\mathscr{T}_i$ be represented by a vector ψ_i. The implementation involved:

1. *Parameter representation:* Each task $\mathscr{T}_i$ was represented by its normalized parameter vector $\psi_i \in \mathbb{R}^k$. Normalization (e.g., min-max scaling across the range of parameters encountered) ensures that different parameters contribute appropriately to distance calculations.

2. *Distance calculation:* The similarity between a task $\mathscr{T}_i$ and the target task $\mathscr{T}_{\text{target}}$ was quantified by calculating the Euclidean distance (L_2 norm) between their respective normalized parameter vectors, ψ_i and ψ_{target}:

   $$d_i = \|\psi_i - \psi_{\text{target}}\|_2 \tag{4.24}$$

   A smaller distance d_i indicates higher similarity to the target task configuration.

3. *Normalization:* These raw distances were subsequently normalized using the min-max scaling formula (as defined in the general difficulty assessment methodology, e.g., Eq. 4.12) to produce the final difficulty score δ_i:

   $$\delta_i = \frac{d_i - \min_j(d_j)}{\max_j(d_j) - \min_j(d_j)} \tag{4.25}$$

   This maps the difficulty to the [0, 1] range, where 0 represents the task parametrically closest (least difficult relative) to the target among the considered set, and 1 represents the farthest (most difficult relative).

These normalized difficulty scores δ_i for scalar-parameter tasks were then used analogously to the scores derived from visual inputs for clustering tasks into difficulty bins (N_{bins}) and guiding the curriculum progression within the BCG framework.

5. RESULTS

This chapter presents the empirical findings concerning the performance of the proposed Bayesian Curriculum Generation (BCG) methodology. Comparative results were obtained through experiments conducted in two distinct reinforcement learning (RL) environments: the discrete, grid-based MiniGrid and the continuous, dynamic AeroRival Pursuit simulation. These testbeds were chosen to evaluate the algorithm's effectiveness across different environmental characteristics and challenges, particularly those involving sparse rewards and complex task structures.

5.1 Performance in the MiniGrid Environment

This section details the experimental setup and outcomes for the BCG algorithm and baseline methods within the MiniGrid environment suite. Key aspects include the use of Autoencoders with t-SNE for task representation and difficulty assessment, integrated with the BN-based adaptive curriculum generator.

5.1.1 MiniGrid environment characteristics and setup

The specific environment utilized was *MiniGrid-DoorKey* (Chevalier-Boisvert, Dai, Towers, Perez-Vicente, Willems, Lahlou, Pal, Castro, & Terry, 2023). In this setting, an agent operates within a grid world and must learn a sequence of actions: navigate to find a key, use the key to unlock a specific door, and finally proceed to a designated goal location (visualized generally in Figure 5.1). This task structure inherently requires multi-step planning and handling sparse rewards, as positive feedback is typically only provided upon reaching the final goal.

The agent perceives the environment through partial, egocentric visual observations and selects actions from a discrete set (e.g., move forward, turn left/right, pick up, toggle door). For the evaluation, four distinct task configurations were created based on the DoorKey setup. These tasks varied primarily in grid size (two 6x6, two 8x8)

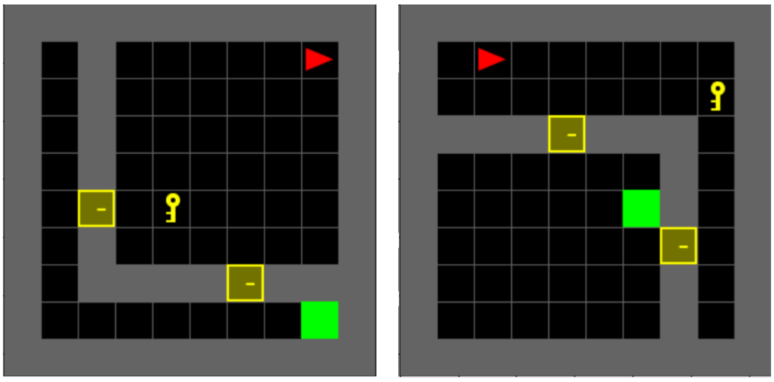


Figure 5.1: General illustration of a MiniGrid environment variant. The DoorKey tasks used involve navigating grids, finding a key (object), unlocking a door (obstacle), and reaching a goal (target square)

and the relative starting positions of the agent, the key, the door, and the goal location. These positional variations alter the required navigation path and interaction sequence, thereby modulating task complexity without changing the fundamental objective. The curriculum generated by BCG aims to sequence these or similar parameter variations to facilitate learning from simpler to more complex instances.

5.1.1.1 Evaluation metrics

Performance within MiniGrid was primarily assessed using the average episodic reward. In this environment setup, a successful episode (reaching the goal) yields a reward between 0 and 1, calculated based on the efficiency (number of steps taken). A reward closer to 1 indicates a faster, more optimal solution path. Failure to reach the goal within a time limit results in a low or zero reward. Averaging this reward over multiple evaluation episodes provides a measure of the learned policy's effectiveness and convergence speed. This metric implicitly reflects episode length, as higher rewards correlate with shorter successful episodes.

5.1.2 Analysis of visual task representation

Before presenting comparative results, we analyze the effectiveness of the Autoencoder (AE) and t-SNE pipeline (detailed in Section 4.5.2 and 4.5.3) in capturing task similarities for visual MiniGrid tasks. The goal was to verify if the learned latent representations, when visualized in 2D via t-SNE, grouped tasks in a way that reflects their inherent structural similarities or difficulties.

Figure 5.2 shows the 2D t-SNE projection of latent features extracted by the trained encoder from various MiniGrid-DoorKey task configurations. Each point represents a unique task instance. The spatial arrangement suggests that t-SNE preserves meaningful relationships present in the high-dimensional latent space.

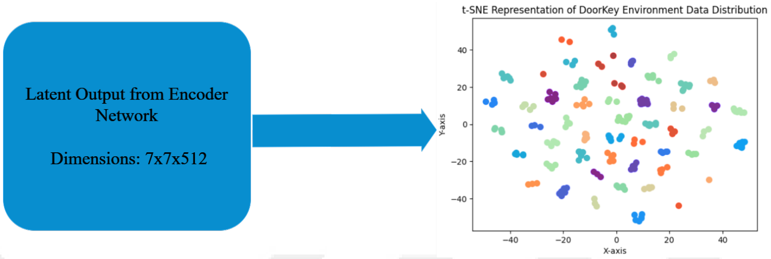


Figure 5.2: Example 2D t-SNE visualization of latent features extracted from MiniGrid tasks by the autoencoder's encoder. Proximity of points suggests similarity in the learned feature space, potentially correlating with task difficulty or structure (adapted from (Akgün & Üre, 2025)).

Further examination revealed distinct clusters emerging in the t-SNE space. Figures 5.3 and 5.5 highlight two such clusters, arbitrarily labeled A and B. Corresponding task examples are shown in Figures 5.4 and 5.6. Tasks within Cluster A (Figure 5.4) tend to share similar layouts or require analogous solution strategies, reflected by their proximity in the t-SNE plot (Figure 5.3). Similarly, tasks within Cluster B (Figure 5.6) form another group based on shared characteristics, spatially separated from Cluster A in the projection (Figure 5.5).

This visual analysis confirms that the implemented AE+t-SNE pipeline generates representations that capture inherent task similarities and differences based on the visual inputs. The emergence of distinct clusters supports the approach of using these representations for unsupervised task categorization and difficulty assessment within the BCG framework. The spatial arrangement reflects relative similarities, which forms the basis for the distance calculations used in the difficulty pipeline (Section 4.5.4).

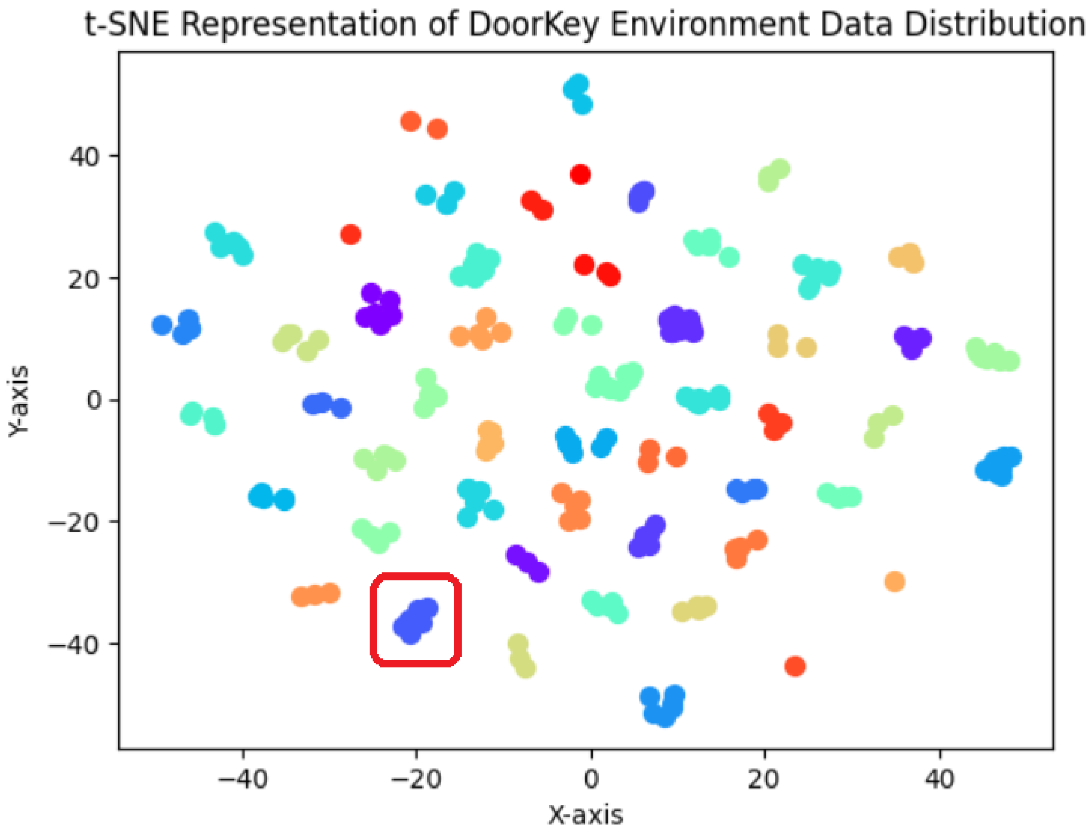


Figure 5.3: t-SNE projection highlighting tasks grouped into Cluster A (adapted from (Akgün & Üre, 2025)).

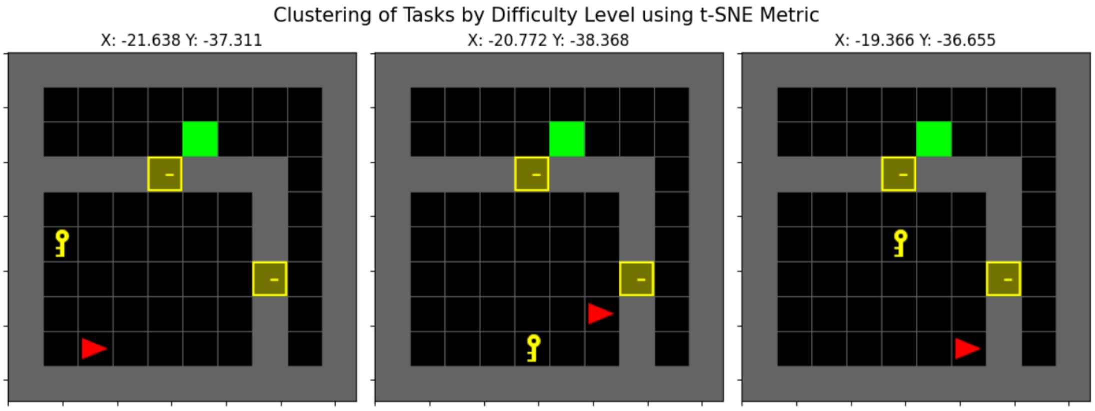


Figure 5.4: Example task configurations from Cluster A, illustrating shared structural features (adapted from (Akgün & Üre, 2025)).

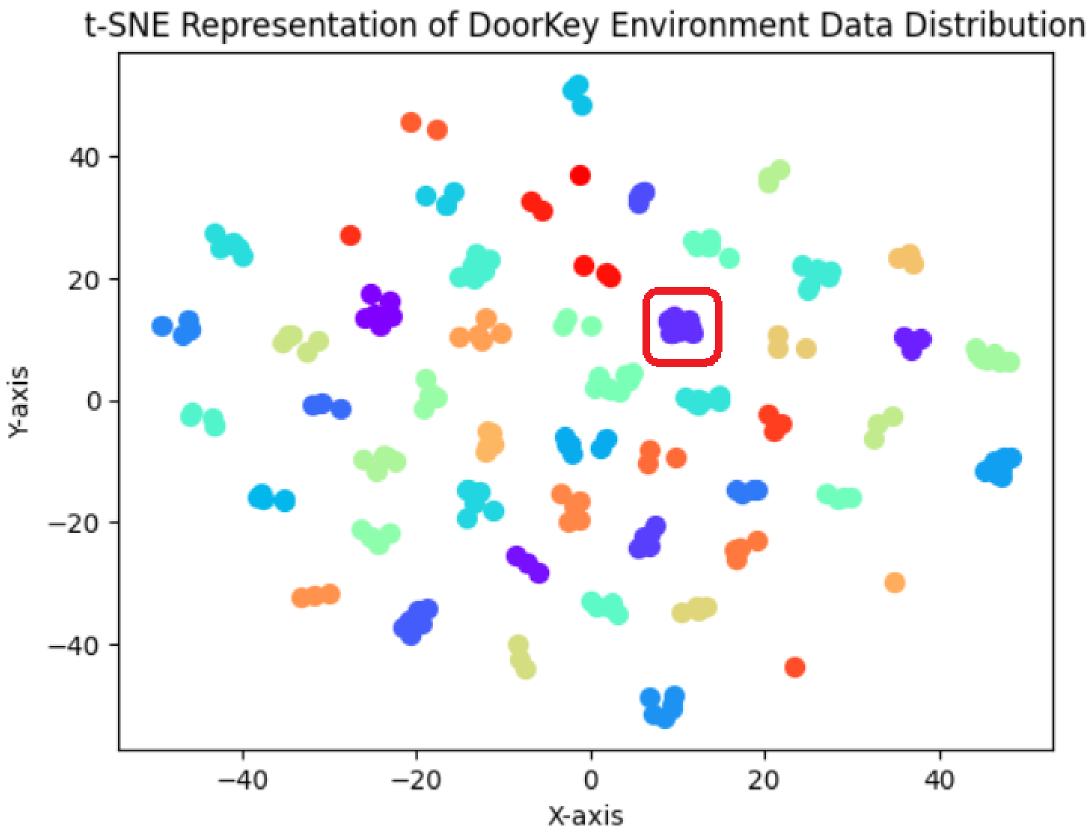


Figure 5.5: t-SNE projection highlighting tasks grouped into Cluster B, distinct from Cluster A (adapted from (Akgün & Üre, 2025)).

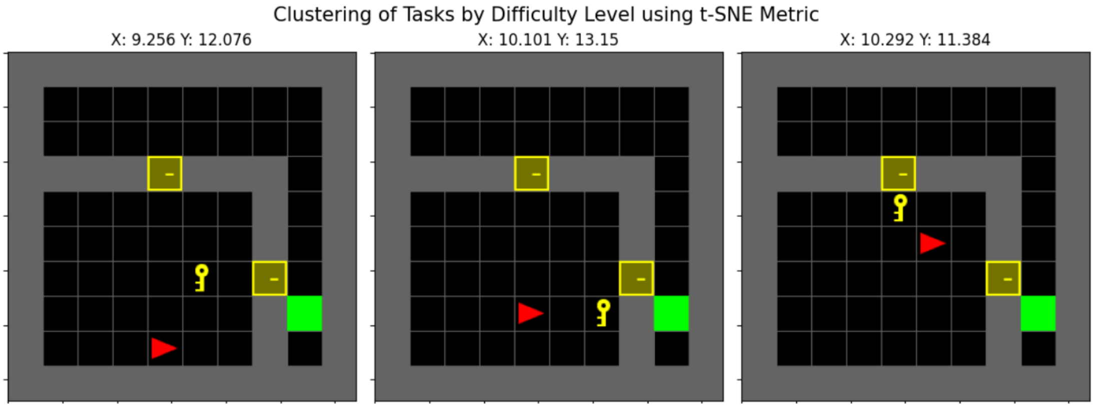


Figure 5.6: Example task configurations from Cluster B (adapted from (Akgün & Üre, 2025)).

5.1.3 Comparative performance evaluation

BCG's performance was benchmarked against several contemporary algorithms selected for their relevance to exploration and structured learning in RL: AGAC (Flet-Berliac, Ferret, Pietquin, Preux, & Geist, 2021), CBET (Parisi, Dean, Pathak, & Gupta, 2021), RAPID (Zha, Ma, Yuan, Hu, & Liu, 2021), DEIR (Wan, Tang, Tian, & Kaneko, 2023), AMIGO (Campero, Raileanu, Küttler, Tenenbaum, Rocktäschel, & Grefenstette, 2020), MASK (Ben-Iwhiwhu, Nath, Pilly, Kolouri, & Soltoggio, 2022), and PREFVEC (Szoke, Shperberg, Holtz, & Allievi, 2024). The rationale for selecting these baselines is further discussed in the Literature Review (Section 2). The evaluation framework consisted of a series of MiniGrid tasks with systematically increasing complexity, as depicted in Figure 5.7.

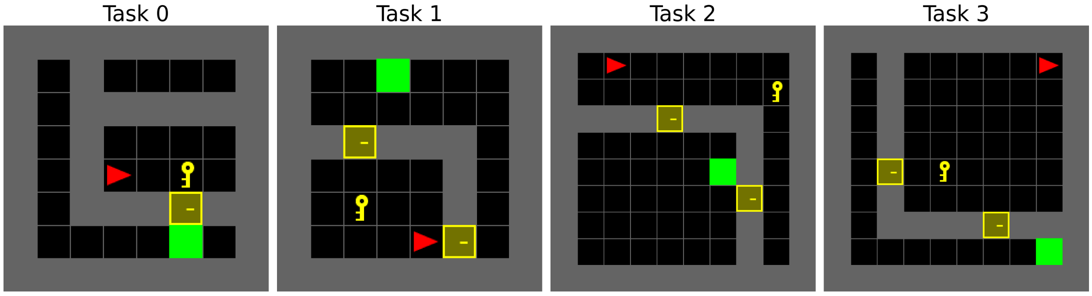


Figure 5.7: Illustration of Increasing Task Difficulty in MiniGrid Test Scenarios. (a) Task 0: Baseline map. (b) Task 1: Simple obstacles. (c) Task 2: Complex barriers, longer paths. (d) Task 3: Highest complexity map. This progression tests algorithm adaptability (adapted from (Akgün & Üre, 2025)).

The comparative analysis involved executing BCG and the eight baseline methods (AMIGO (Campero, Raileanu, Küttler, Tenenbaum, Rocktäschel, & Grefenstette, 2020), CBET (Parisi, Dean, Pathak, & Gupta, 2021), AGAC (Flet-Berliac, Ferret, Pietquin, Preux, & Geist, 2021), DEIR (Wan, Tang, Tian, & Kaneko, 2023), RAPID (Zha, Ma, Yuan, Hu, & Liu, 2021), PPO (Schulman, Wolski, Dhariwal, Radford, & Klimov, 2017), MASK (Ben-Iwhiwhu, Nath, Pilly, Kolouri, & Soltoggio, 2022), and PREFVEC (Szoke, Shperberg, Holtz, & Allievi, 2024)) on the MiniGrid tasks. Each algorithm ran three times per task with different random seeds for robustness. The PPO (Schulman, Wolski, Dhariwal, Radford, & Klimov, 2017) implementation from

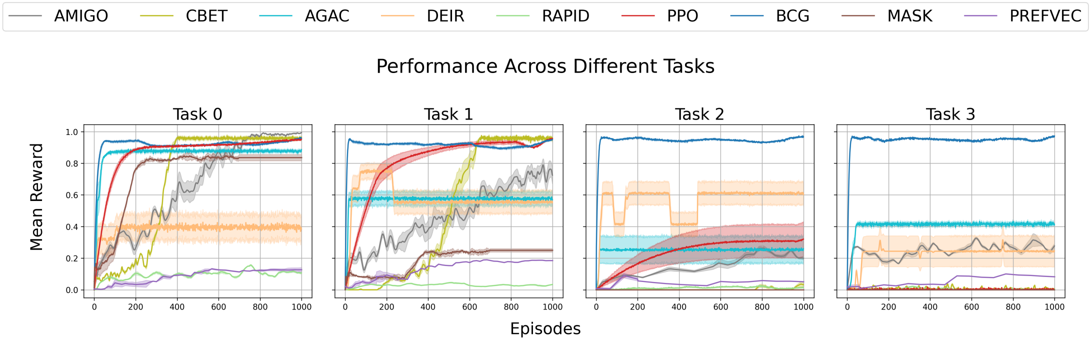


Figure 5.8: Learning Curves of RL Algorithms on MiniGrid Tasks. Shows mean reward ± std dev. BCG demonstrates consistent high performance and low variance. AMIGO/CBET excel early but decline. AGAC/DEIR show moderate, variable performance. RAPID/PPO lag. MASK fails to scale. PREFVEC struggles. Highlights BCG's adaptability (adapted from (Akgün & Üre, 2025)).

Table 5.1: Quantitative Comparison of Algorithm Performance on MiniGrid Tasks. Mean reward ± std dev over three trials. Higher is better. Bold indicates best per task (adapted from (Akgün & Üre, 2025)).

Method	**Task 0**	**Task 1**	**Task 2**	**Task 3**
BCG (Ours)	0.960 ± 0.002	0.950 ± 0.003	**0.967** ± **0.003**	**0.973** ± **0.003**
AMIGO	**0.994** ± **0.001**	0.725 ± 0.053	0.200 ± 0.009	0.277 ± 0.014
CBET	0.965 ± 0.010	**0.958** ± **0.007**	0.033 ± 0.013	0.007 ± 0.001
AGAC	0.886 ± 0.008	0.569 ± 0.041	0.260 ± 0.089	0.412 ± 0.012
DEIR	0.385 ± 0.084	0.551 ± 0.072	0.602 ± 0.073	0.245 ± 0.098
RAPID	0.107 ± 0.007	0.034 ± 0.001	0.014 ± 0.003	0.006 ± 0.001
PPO	0.952 ± 0.003	0.957 ± 0.003	0.321 ± 0.111	0.000 ± 0.000
MASK	0.836 ± 0.020	0.250 ± 0.011	0.000 ± 0.000	0.000 ± 0.000
PREFVEC	0.127 ± 0.014	0.185 ± 0.000	0.051 ± 0.000	0.083 ± 0.000

the Stable Baselines 3 library (Raffin, Hill, Gleave, Kanervisto, Ernestus, & Dormann, 2021) served as the base RL algorithm for BCG.

Figure 5.8 presents the learning curves. BCG consistently achieved high reward levels rapidly and demonstrated stable performance (narrow variance band). While AMIGO and CBET performed well on easier tasks, their effectiveness decreased significantly as complexity rose. AGAC and DEIR showed moderate but less stable performance. RAPID and the standard PPO baseline consistently underperformed. MASK showed initial promise but failed to scale, while PREFVEC struggled throughout, likely due to its design not being optimized for MiniGrid's sparse rewards.

Table 5.1 quantifies these observations. While AMIGO achieved the highest score on the simplest Task 0, and CBET on Task 1, BCG clearly registered the best performance on the more complex Tasks 2 (0.967) and 3 (0.973). Importantly, BCG maintained very low standard deviations on these tasks, indicating high reliability. The performance drop-off for algorithms like MASK and the consistent low scores for PREFVEC are also numerically evident.

5.1.4 BCG process analysis (Minigrid)

To provide further insight into the BCG algorithm's operation, we illustrate its adaptive curriculum generation process within the MiniGrid-DoorKey setting. Figure 5.9 shows an example sequence of tasks generated by the curriculum, demonstrating a gradual increase in complexity (e.g., spatial distance between key, door, and goal). The framework utilizes transfer learning between these stages.

The adaptive nature is highlighted when the agent struggles with a particular task difficulty level. The Bayesian Network component can adjust the task sampling probabilities, potentially selecting slightly easier or different tasks that better match the agent's current capabilities before re-attempting harder ones, as conceptualized in Figure 5.10.

The overall training efficiency is reflected in the learning progress over time. Figure 5.11 plots the average timesteps required to solve tasks at different stages of the curriculum. Initially, simple tasks are solved quickly. As complexity increases,

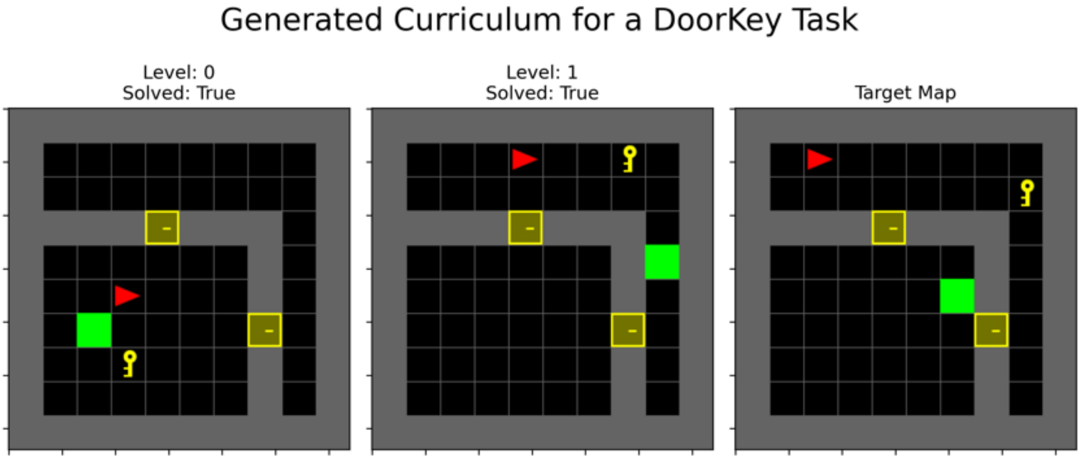


Figure 5.9: Example illustration of task progression generated by BCG in MiniGrid, showing increasing complexity over the curriculum stages (adapted from (Akgün & Üre, 2025)).

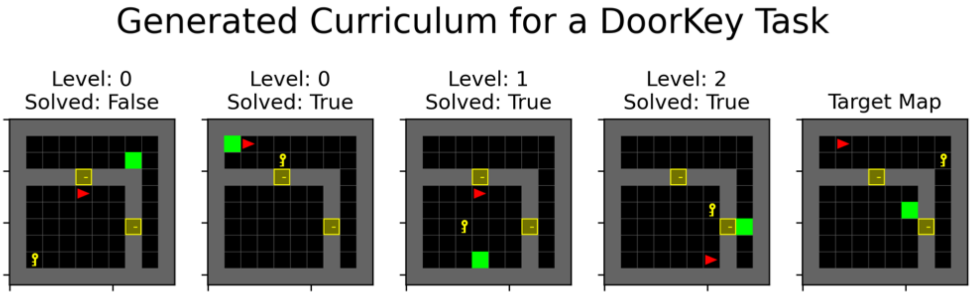


Figure 5.10: Conceptual depiction of adaptive task selection within BCG. If progress stalls, the framework can sample alternative tasks within or near the current difficulty level based on BN probabilities and performance feedback (adapted from (Akgün & Üre, 2025)).

completion time grows, but the agent successfully learns. Variations in completion time might also reflect the adaptive nature, where occasional harder tasks might be sampled or retried. Despite this, the BCG approach demonstrates effective task resolution compared to baseline methods in these settings.

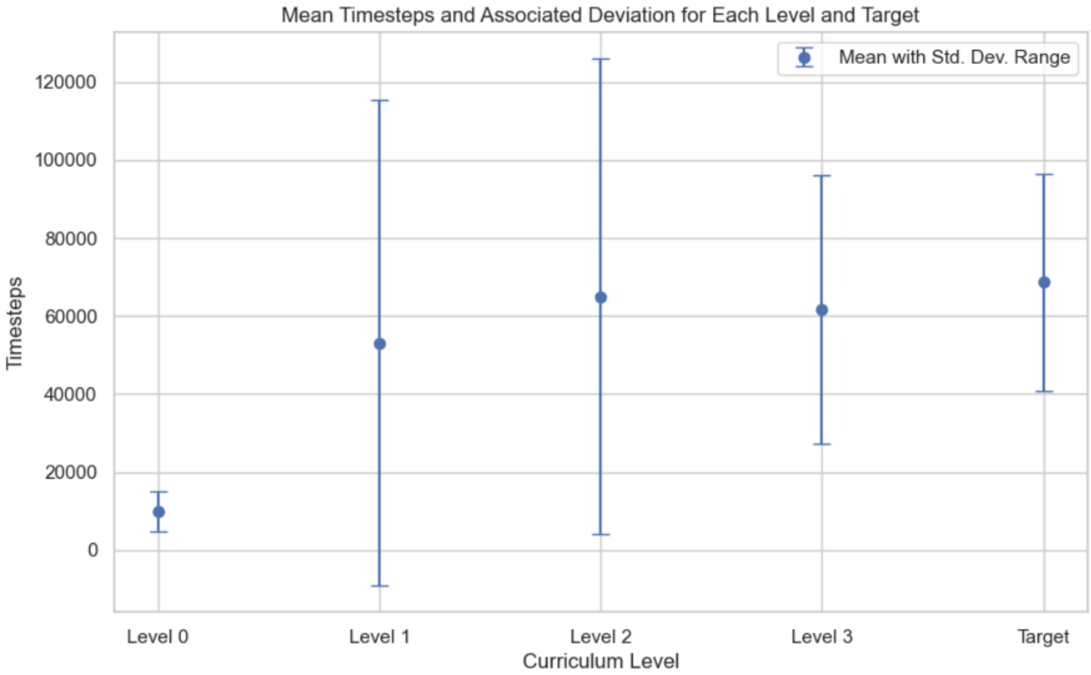


Figure 5.11: Training progress showing mean (and standard deviation) timesteps to task completion across curriculum stages in MiniGrid. Illustrates initial speed on easy tasks and successful learning despite increasing time on harder tasks (adapted from (Akgün & Üre, 2025)).

5.1.5 Minigrid summary

The results from the MiniGrid experiments, encompassing both learning dynamics (Figure 5.8) and final performance metrics (Table 5.1), demonstrate the effectiveness of the BCG algorithm in this environment. BCG exhibited superior adaptability to increasing task complexity, consistently achieving high rewards with low variance compared to the evaluated baselines. This suggests BCG is a robust method for tackling challenges typical of partially observable grid worlds with sparse rewards.

5.2 Performance in the AeroRival Pursuit Environment

This section reports on the evaluation of the BCG algorithm within the AeroRival Pursuit Environment, focusing on continuous control, a continuous-space 2D simulation featuring navigation, hazard avoidance, and adversarial elements.

5.2.1 Evaluation metrics

As in MiniGrid, performance was evaluated using the average episodic reward, but the scale differs. Here, rewards range from -1 (failure) to +1 (optimal success), with values between 0 and 1 reflecting faster successful completions. This metric effectively captures both mission success and agent efficiency.

5.2.2 AeroRival pursuit environment characteristics

In AeroRival Pursuit, an agent-controlled aircraft aims to reach a target before a competing adversary, while navigating hazardous zones that trigger defensive rockets upon entry. Success requires reaching the correct target (potentially inferred from the adversary's trajectory if multiple targets exist) safely and efficiently. The environment combines continuous control with strategic decision-making under threat and competition.

5.2.3 Comparative performance evaluation

Performance was evaluated on two AeroRival configurations: Task 0 (baseline with one hazard zone, Figure 5.12) and Task 1 (complex with three hazard zones, Figure 5.13). Rewards ranged from -1 (failure) to +1 (optimal success), with sparsity adding challenge. BCG, using PPO (Schulman, Wolski, Dhariwal, Radford, & Klimov, 2017) from Stable Baselines 3 (Raffin, Hill, Gleave, Kanervisto, Ernestus, & Dormann, 2021) as its base, was compared against DDPG, TD3, PPO, SAC, MASK, and COHER in this setting.

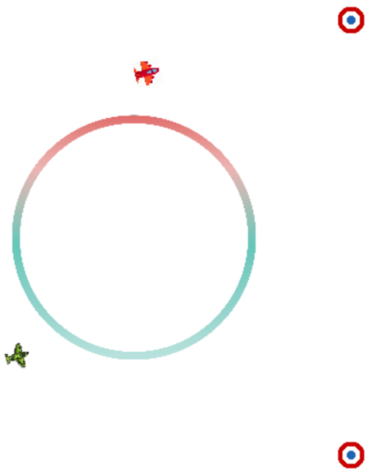


Figure 5.12: AeroRival Pursuit Baseline Setup (Task 0). Single threat zone (adapted from (Akgün & Üre, 2025)).

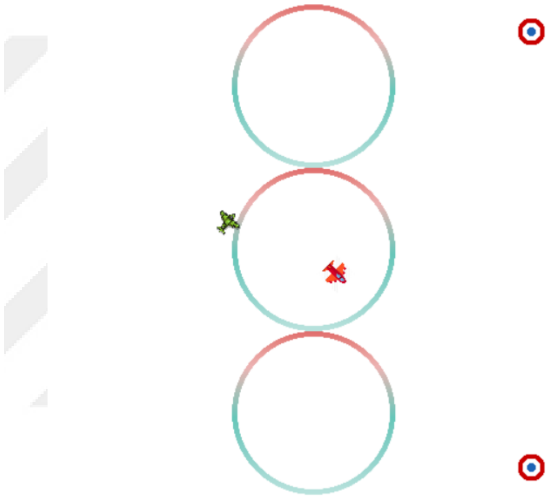


Figure 5.13: AeroRival Pursuit Complex Setup (Task 1). Three threat zones (adapted from (Akgün & Üre, 2025)).

Table 5.2: Quantitative Comparison of Algorithm Performance on AeroRival Pursuit Tasks. Mean reward ± std dev over three trials. Positive values indicate success. Bold indicates best per task (adapted from (Akgün & Üre, 2025)).

Method	Task 0	Task 1
BCG (Ours)	**0.668 ± 0.070**	**0.726 ± 0.016**
DDPG	-0.493 ± 0.110	-0.672 ± 0.110
TD3	-0.980 ± 0.002	-0.981 ± 0.002
PPO	-0.740 ± 0.121	-0.974 ± 0.002
SAC	-0.450 ± 0.118	-0.988 ± 0.003
MASK	-0.647 ± 0.016	-1.000 ± 0.000
COHER	-0.823 ± 0.002	-0.964 ± 0.000

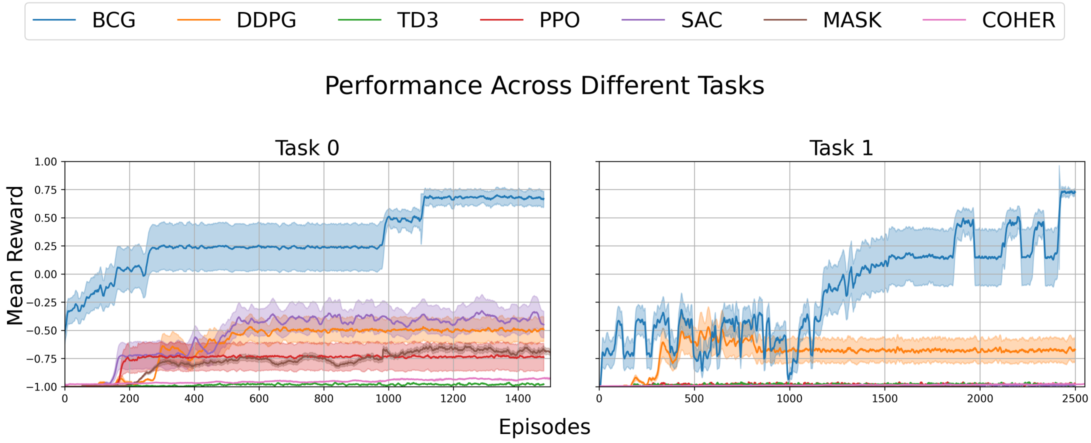


Figure 5.14: Learning Curves of RL Algorithms on AeroRival Pursuit Tasks. Mean reward ± std dev. BCG shows rapid convergence to high positive rewards in Task 0 and maintains strong performance in Task 1, benefiting from curriculum structure. Baselines struggle significantly, mostly failing to achieve positive rewards, especially in Task 1 (adapted from (Akgün & Üre, 2025)).

The learning curves in Figure 5.14 show BCG rapidly reaching high positive rewards in Task 0, indicating effective learning of navigation and hazard avoidance. The jumps correspond to curriculum stage transitions. In contrast, baseline algorithms largely failed to achieve positive rewards. In the more complex Task 1, BCG maintained its superior performance, achieving positive rewards while most baselines (PPO, SAC, TD3, MASK, COHER) failed completely. DDPG showed minimal capability but was still significantly outperformed by BCG. Fluctuations in BCG's Task 1 curve are indicative of the curriculum adaptation process.

Table 5.2 provides the numerical comparison. BCG achieved substantial positive rewards in both Task 0 (0.668) and Task 1 (0.726), demonstrating successful task completion and consistency (especially in Task 1 with low std dev). All baseline algorithms registered negative average rewards across both tasks, indicating consistent failure. MASK notably scored -1.000 in Task 1, signifying complete failure on all runs.

5.2.4 AeroRival summary

The experimental results in the AeroRival Pursuit environment consistently highlight the effectiveness of the BCG algorithm. As shown visually (Figure 5.14) and numerically (Table 5.2), BCG successfully solved both task configurations, achieving positive rewards and demonstrating stability, particularly in the more complex scenario. This contrasts sharply with the benchmark algorithms, which failed to cope with the environment's demands. These findings suggest BCG is a capable approach for RL problems involving continuous control, dynamic interactions, and sparse rewards.

6. DISCUSSIONS

This chapter synthesizes and interprets the empirical findings presented in Chapter 5. We analyze the performance patterns of the Bayesian Curriculum Generation (BCG) algorithm across different environments, relate these outcomes to the initial research objectives and questions defined in Chapter 1, and compare them with existing literature discussed in Chapter 2. Furthermore, we discuss the broader implications of the work and acknowledge its limitations.

6.1 Synthesis of Key Findings

The empirical results presented in Chapter 5 consistently highlight the efficacy of the proposed Bayesian Curriculum Generation (BCG) algorithm. Across two distinct and challenging environments – the discrete, partially observable MiniGrid-DoorKey tasks and the continuous, adversarial AeroRival Pursuit simulation – BCG demonstrated a marked advantage in learning performance compared to standard Proximal Policy Optimization (PPO) and a range of contemporary baseline algorithms. Key findings include:

- *Superior performance:* BCG consistently achieved higher average rewards and demonstrated greater success rates, particularly in the more complex task configurations within both environments (Tables 5.1 and 5.2). In AeroRival, BCG was notably the only method to achieve consistent positive rewards, indicating successful task completion where others failed systemically.
- *Enhanced adaptability:* The framework proved robust to increasing task complexity. In MiniGrid, BCG maintained high performance across Tasks 0 through 3, whereas many baselines exhibited significant performance degradation (Figure 5.8). Its success in both the grid-based MiniGrid and the continuous AeroRival highlights its versatility across different environment dynamics and state/action spaces.

- *Improved stability and efficiency:* BCG generally exhibited lower variance in performance across multiple runs compared to several baselines (indicated by shaded areas in Figures 5.8 and 5.14). While direct sample efficiency comparison was complex, the faster convergence to high reward levels, especially evident in AeroRival Task 0, suggests improved learning efficiency facilitated by the curriculum structure. The characteristic jumps in BCG's learning curves likely correspond to effective knowledge transfer between curriculum stages (Figure 5.14).
- *Effective task representation:* The analysis of the Autoencoder and t-SNE pipeline for MiniGrid (Section 5.1.2) indicated that the learned visual features effectively captured task similarities, leading to meaningful clusters (Figures 5.3-5.6) that could be leveraged for difficulty assessment.

These findings collectively suggest that the structured, adaptive curriculum generated by BCG successfully guides the learning process in challenging, sparse-reward settings.

6.2 Relation to Research Objectives and Questions

The empirical findings presented in Chapter 5 provide insights into the research questions that motivated this work. We now explicitly address each question based on the performance and behavior observed for the Bayesian Curriculum Generation (BCG) framework.

Addressing RQ1: How can Bayesian networks be effectively utilized to create adaptive curriculum learning frameworks that systematically structure task progressions based on agent capabilities?

The results demonstrate that Bayesian Networks (BNs) serve as an effective foundation for structuring and adapting RL curricula. As detailed in the Methodology (Section 4.2), BNs were used to model dependencies between task-defining parameters. This probabilistic structure enabled the systematic generation of related, yet diverse, task instances (Section 4.3.4). The successful learning trajectories observed, particularly the ability to solve complex final tasks where direct RL failed (Chapter 5), indicate that the curricula generated possessed a meaningful structure facilitating skill acquisition (e.g., Figure 5.9). Furthermore, the framework integrates agent capabilities (performance metrics $\mathscr{M}_t$) into the task selection process (Equation 4.16) and allows for dynamic updates to perceived task difficulty based on performance (Equation 4.22). This linkage creates an adaptive system where the curriculum structure, rooted in the BN, evolves in response to the agent's learning state. Therefore, the evidence suggests BNs are effectively utilized by BCG to provide both systematic structure and capability-driven adaptation in curriculum generation.

Addressing RQ2: What advantages might probabilistic curriculum generation offer compared to existing approaches such as teacher-student frameworks, intermediate goal creation, and self-play methods?

BCG's probabilistic generation, centered on the BN, demonstrated clear advantages in the tested scenarios compared to the likely outcomes of alternative approaches discussed in the literature (Section 6.3).

- Compared to some Teacher-Student frameworks (Graves, Bellemare, Menick, Munos, & Kavukcuoglu, 2017; Matiisen, Oliver, Cohen, & Schulman, 2019) that primarily react to student progress, BCG's BN provides an explicit, generative model of the task space itself. This may offer more structured control over task distribution and difficulty, potentially leading to the observed robust performance even when agent progress signals might be noisy or delayed in sparse reward settings.

- Unlike Intermediate Goal Creation methods (Florensa, Held, Geng, & Abbeel, 2018; Florensa, Held, Wulfmeier, Zhang, & Abbeel, 2017) that typically define sub-goals within a fixed environment, BCG directly manipulates the underlying task parameters governed by the BN. This parameter-space curriculum proved highly effective for environments like MiniGrid and AeroRival where difficulty is intrinsically tied to layout or physics parameters, offering perhaps a more direct control mechanism than goal setting alone.

- While Self-Play (Duan, Chen, Houthooft, Schulman, & Abbeel, 2016; Sukhbaatar, Lin, Kostrikov, Synnaeve, Szlam, & Fergus, 2017; Sukhbaatar, Szlam, Synnaeve, Chintala, & Fergus, 2015) generates curricula based on opponent modeling, BCG provides explicit control over environmental task difficulty through its probabilistic parameter generation. This makes it directly applicable to single-agent tasks and potentially more sample-efficient initially, as it doesn't require training competent opponents.

The significant performance margin between BCG and various baselines in Chapter 5 suggests that its specific approach—combining probabilistic task modeling via BNs with adaptive sequencing—offered tangible advantages in effectiveness and adaptability for these challenging sparse reward problems.

Addressing RQ3: How can a curriculum learning system effectively balance between task difficulty and agent capability in environments with sparse rewards or complex skill hierarchies?

BCG employs several interacting mechanisms to achieve this balance. Firstly, task difficulty is explicitly quantified relative to the target task using learned representations or parameter distances, followed by normalization (δ_i, Section 4.2.2, 4.5.4, 4.5.5). Secondly, tasks are grouped into discrete difficulty levels via unsupervised clustering (N_{bins}, Section 4.2.3), providing a structured scaffold. Thirdly, progression between these levels and selection within levels are gated by agent performance, using success thresholds (η, Equation 4.21) and probabilistic selection biased by current capabilities (Equation 4.16). The results demonstrate this balance was effective: agents successfully learned complex, sparse-reward tasks in both MiniGrid and AeroRival (Chapter 5). They progressed through increasing difficulties (Figure 5.9) without evidence of insurmountable steps (getting stuck) or premature advancement leading to failure (supported by hyperparameter analysis, Figure 4.3). This indicates the system successfully matched task challenge to the agent's evolving competence.

Addressing RQ4: To what extent can a generalized curriculum learning framework maintain effectiveness across diverse reinforcement learning algorithms and environmental contexts?

The results provide strong evidence for BCG's effectiveness across diverse environmental contexts. Its success in both the discrete, grid-based MiniGrid and the continuous, adversarial AeroRival Pursuit environment (Chapter 5) demonstrates the framework's applicability to fundamentally different state spaces, action spaces, and task dynamics. The methodology for handling both visual (via AE+tSNE) and scalar parameters further supports this versatility. Regarding different reinforcement learning algorithms, BCG was intentionally designed to be modular (Section 4.2). The experiments successfully validated its use with PPO (Schulman, Wolski, Dhariwal, Radford, & Klimov, 2017), a standard and effective actor-critic algorithm. However, the current study did not empirically evaluate BCG with other algorithm families (e.g., off-policy methods like SAC/DDPG, or value-based methods). Therefore, while designed for generality and proven effective with PPO, definitive conclusions about its universal effectiveness across *all* RL algorithms require further investigation, as noted in Limitations (Section 6.5).

Addressing RQ5: What design principles enable curriculum learning systems to adapt dynamically to changing agent capabilities throughout the training process?

The BCG framework incorporates several key design principles enabling its dynamic adaptation, validated by the successful learning observed:

1. *Continuous performance monitoring:* Tracking metrics like reward (R_t) provides real-time feedback on agent capability (Equation 4.17).
2. *Quantified and structured task space:* Representing tasks ($\phi(T_i), \psi_i$) allows measuring difficulty/similarity (d_i, δ_i, s_{norm}), and clustering (C_k) provides discrete stages reflecting capability levels.
3. *Performance-gated progression:* Using explicit thresholds (η) ensures mastery before advancing, linking progression directly to demonstrated capability (Equation 4.21).
4. *Adaptive task selection:* Probabilistic selection (Equation 4.16), potentially biased by current performance and task characteristics, allows focusing on appropriately challenging tasks within or across stages.

5. *Dynamic difficulty adjustment:* Mechanisms like updating solvability indicators ($\sigma(T_i)$) and using adaptive similarity thresholds (τ_t) allow the system's perception of task difficulty itself to evolve based on agent interaction (Equation 4.22).

6. *Model updating:* The framework allows for Bayesian updating of the BN parameters θ (Equation 4.18), enabling the underlying task generation model to adapt based on data, though the extent of online updating might vary by implementation.

These principles, implemented within BCG (Chapter 4, Section 4.3), collectively create a system that dynamically adjusts the learning pathway in response to the agent's evolving competence.

6.3 Comparison with Existing Literature

The performance of BCG relative to the baselines provides valuable context within the existing literature (Section 2). While methods like AMIGO and CBET showed promise in specific (often simpler) MiniGrid tasks, they lacked the consistent adaptability of BCG across the full range of complexities. Algorithms focused purely on exploration enhancement or intrinsic motivation (represented implicitly by some baselines' design philosophies) struggled significantly in the sparse reward settings, highlighting the benefits of a structured curriculum. Methods like MASK, potentially incorporating knowledge reuse, succeeded initially but failed to scale, suggesting BCG's probabilistic and adaptive sequencing offers a more robust approach to managing task difficulty progression. PREFVEC's struggles underscore that even successful algorithms from other benchmarks may not be suited for these specific sparse reward challenges without adaptation, further motivating curriculum-based approaches like BCG. Compared to manually designed curricula, BCG offers automation and adaptation, while compared to purely goal-generation methods, the BN provides more structure based on underlying task parameters.

6.4 Implications and Significance

The success of BCG carries several implications. Firstly, it demonstrates the viability of integrating probabilistic graphical models (BNs) into the core of curriculum generation, offering a principled way to manage task parameterization and dependencies. Secondly, it reinforces the importance of adaptive curricula that respond to agent progress, rather than relying on fixed sequences. Thirdly, the versatility shown across MiniGrid and AeroRival suggests the underlying principles of BCG could be applicable to a broader range of challenging RL domains, such as robotics, game AI, and complex simulations, where sparse rewards and intricate task structures are common. This work contributes a concrete algorithmic framework that balances structured task generation with dynamic adaptation, potentially inspiring similar hybrid approaches in automated machine learning and AI training.

6.5 Limitations

Despite its promising results, BCG has several limitations that provide clear avenues for future work:

- *Manual Design and Computational Trade-off of the Bayesian Network:* A primary limitation is that the BN structure, which models the dependencies between task parameters, must be manually designed. This design process involves a critical trade-off between model fidelity and computational cost. For example, an expert might define a simple, computationally efficient causal chain, such as `Parameter A > Parameter B > Difficulty`. However, a more complex and potentially more accurate model—where multiple parameters are modeled as joint causes influencing a single outcome, resulting in a more densely connected graph—would demand significantly more computational resources for probabilistic inference. This manual, expert-driven design not only requires domain knowledge but also directly dictates the computational overhead of the curriculum generation process.

- *Static Parameter Relationships:* The current implementation assumes the relationships between task parameters, as encoded in the BN, remain static. This may not hold in non-stationary environments where task dynamics change over time, potentially limiting BCG's effectiveness in such scenarios without periodic re-evaluation of the network structure.

- *Hyperparameter Sensitivity:* While our analysis showed robustness within tested ranges (Table 4.1), BCG's performance can still depend on the careful tuning of key hyperparameters, including the binning number for discretization (N_{bins}) and the weighting factors for curriculum selection (λ_1, λ_2).

- *Scope of Evaluation:* While tested in two distinct and challenging environments, evaluation across a wider variety of domains, especially those with continuous parameter spaces or different underlying causal structures, is necessary to fully establish the generalizability of BCG.

6.6 Concluding Remarks on Discussion

In summary, the empirical results strongly support the Bayesian Curriculum Generation algorithm as an effective method for improving reinforcement learning in sparse reward environments. By leveraging Bayesian Networks for structured task modeling and incorporating adaptive mechanisms based on agent performance, BCG consistently outperformed various baseline methods across different task complexities and environmental dynamics. The findings align with the research objectives and demonstrate clear advantages over methods struggling with exploration or lacking adaptability. While limitations related to parameterization, environmental dynamics, and computation exist, the work provides significant insights and a robust foundation for future development in automated and adaptive curriculum learning.

7. CONCLUSION AND FUTURE WORK

7.1 Conclusion

This thesis addressed the significant challenge of efficient learning in reinforcement learning (RL) environments characterized by sparse rewards, where traditional methods often struggle due to inadequate exploration and slow convergence. We introduced the Bayesian Curriculum Generation (BCG) algorithm (Akgün & Üre, 2025), a novel framework for automated and adaptive curriculum learning. The core principle of BCG is the integration of probabilistic modeling using Bayesian Networks (BNs) with adaptive sequencing mechanisms. BNs are employed to model the underlying structure and parameter dependencies of the task space, enabling the structured generation of diverse tasks. Task difficulty is quantified based on learned representations (using autoencoders for visual inputs) or parameter vectors, and tasks are grouped into difficulty levels via clustering. Crucially, the curriculum adapts dynamically to the agent's learning progress, utilizing performance metrics to guide probabilistic task selection and update difficulty assessments, while leveraging transfer learning between curriculum stages.

The effectiveness and versatility of BCG were empirically demonstrated through comprehensive experiments in two distinct environments: the discrete, partially observable MiniGrid-DoorKey navigation task and the continuous, adversarial AeroRival Pursuit simulation. In both settings, BCG, using PPO as its base RL learner, consistently and significantly outperformed standard PPO and a range of contemporary baseline algorithms designed for exploration or related curriculum concepts. Key findings indicated BCG's superior final performance, greater learning stability, and robust adaptability to increasing task complexity and diverse environmental dynamics (Chapter 5). The results validated the core hypothesis that combining structured probabilistic task modeling with online adaptation provides a powerful approach to overcoming the hurdles of sparse reward RL.

The main contributions of this work include (1) The development of the novel BCG framework, demonstrating the successful integration of Bayesian Networks for structured, automated curriculum generation in RL. (2) The implementation of adaptive mechanisms, including performance-based task selection and dynamic difficulty assessment using learned features or parameters, enabling the curriculum to respond to agent capabilities. (3) Empirical validation of BCG's effectiveness and versatility across different types of challenging, sparse-reward environments (discrete/visual and continuous/parameter-based).

Despite these promising results, the current BCG framework has several limitations, as discussed in Chapter 6. The reliance on manually identified key task parameters for the BN structure can be a bottleneck in highly complex or poorly understood environments. The assumption of static relationships between these parameters limits applicability in non-stationary settings where task dynamics evolve over time. Additionally, the computational overhead associated with BN learning/inference, feature extraction, clustering, and curriculum management requires consideration, especially for scaling to very large problems. Finally, performance remains dependent on careful hyperparameter tuning, and broader evaluation across more diverse tasks and RL algorithms is needed to fully assess generalizability.

7.2 Future Research Directions

Addressing the identified limitations and extending the capabilities of BCG provides several exciting avenues for future research:

- *Automated parameter identification and BN structure learning:* A key priority is reducing the reliance on manual parameter selection. Future work will investigate methods to automatically identify curriculum-relevant parameters directly from agent interaction data or environment specifications. Techniques from feature selection, dimensionality reduction, or even applying reinforcement learning itself to optimize the choice of BN nodes and structure could be explored. This would significantly enhance the autonomy and applicability of BCG to novel environments where key difficulty factors are not immediately obvious.

- *Extension to dynamic and non-stationary environments:* The current assumption of static task parameter relationships needs to be relaxed. We plan to extend BCG to explicitly handle dynamic systems where task characteristics or dependencies might change over time. This could involve incorporating temporal dependencies into the BN structure (e.g., using Dynamic Bayesian Networks), developing mechanisms for online updating of both BN parameters (θ) and structure (G) based on recent agent experience, and designing adaptive sequencing strategies that are more sensitive to non-stationarity and environmental shifts. Evaluating such extensions in environments with evolving dynamics would be crucial.

- *Application and evaluation in complex robotics domains:* To further validate BCG's utility and explore its potential impact, we aim to implement and test the framework in more complex and realistic robotic scenarios. This includes areas such as Multi-Agent Systems: applying adaptive curricula in cooperative or competitive tasks, such as multiple drone navigation and coordination, where inter-agent dynamics add complexity that might be captured by the BN; Long-Horizon Manipulation/Assembly: structuring the learning of complex manipulation sequences with sparse success signals by generating curricula based on object properties, configurations, or sub-task dependencies; and Navigation in Unstructured/Dynamic Environments: creating curricula that adapt not only to agent skill but also to changes in a real-world environment for mobile robots. This involves addressing challenges in defining appropriate task parameters and representations for these richer domains.

- *Improving computational efficiency:* Investigating methods to reduce the computational overhead of the BCG framework is important for scalability. This could involve exploring more efficient approximate inference techniques for BNs, faster clustering algorithms suitable for online updates, or optimizing the feature extraction and difficulty calculation pipeline.

By pursuing these directions, we aim to develop BCG into an even more robust, autonomous, and widely applicable tool for accelerating reinforcement learning in complex, real-world problems. Continued research will help solidify the understanding of how structured, probabilistic, and adaptive curricula can effectively address fundamental challenges in artificial intelligence.

REFERENCES

Akgün, **O.**, & **Üre**, **N. K.** (2025). Bayesian curriculum generation in sparse reward reinforcement learning environments. *Engineering Science and Technology, an International Journal*, *66*, 102048.

Andrychowicz, **M.**, **Wolski**, **F.**, **Ray**, **A.**, **Schneider**, **J.**, **Fong**, **R.**, **Welinder**, **P.**, **McGrew**, **B.**, **Tobin**, **J.**, **Abbeel**, **P.**, & **Zaremba**, **W.** (2017). Hindsight experience replay. *Advances in Neural Information Processing Systems*, 5048–5058.

Andrychowicz, **M.**, **Baker**, **B.**, **Chociej**, **M.**, **Jozefowicz**, **R.**, **McGrew**, **B.**, **Pachocki**, **J.**, **Petron**, **A.**, **Plappert**, **M.**, **Powell**, **G.**, **Ray**, **A.**, et al. (2020). Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, *39*(1), 3–20.

Bajaj, **V.**, **Sharon**, **G.**, & **Stone**, **P.** (2023). Task phasing: Automated curriculum learning from demonstrations. *Proceedings of the International Conference on Automated Planning and Scheduling*, *33*(1), 542–550.

Bengio, **Y.**, **Louradour**, **J.**, **Collobert**, **R.**, & **Weston**, **J.** (2009). Curriculum learning. *Proceedings of the 26th Annual International Conference on Machine Learning*, 41–48. https://doi.org/10.1145/1553374.1553380

Ben-Iwhiwhu, **E.**, **Nath**, **S.**, **Pilly**, **P. K.**, **Kolouri**, **S.**, & **Soltoggio**, **A.** (2022). Lifelong reinforcement learning with modulating masks. *arXiv preprint arXiv:2212.11110*.

Brockman, **G.**, **Cheung**, **V.**, **Pettersson**, **L.**, **Schneider**, **J.**, **Schulman**, **J.**, **Tang**, **J.**, & **Zaremba**, **W.** (2016). Openai gym. *arXiv preprint arXiv:1606.01540*.

Campero, **A.**, **Raileanu**, **R.**, **Küttler**, **H.**, **Tenenbaum**, **J. B.**, **Rocktäschel**, **T.**, & **Grefenstette**, **E.** (2020). Learning with amigo: Adversarially motivated intrinsic goals. *arXiv preprint arXiv:2006.12122*.

Castelletti, **A.**, & **Soncini-Sessa**, **R.** (2007). Bayesian networks and participatory modelling in water resource management. *Environmental Modelling & Software*, *22*(8), 1075–1088.

Chen, **S. H.**, **Jakeman**, **A. J.**, & **Norton**, **J. P.** (2008). Artificial intelligence techniques: An introduction to their use for modelling environmental systems. *Mathematics and computers in simulation*, *78*(2-3), 379–400.

Chen, **S. H.**, & **Pollino**, **C. A.** (2012). Good practice in bayesian network modelling. *Environmental Modelling & Software*, *37*, 134–145.

Chevalier-Boisvert, **M.**, **Dai**, **B.**, **Towers**, **M.**, **Perez-Vicente**, **R.**, **Willems**, **L.**, **Lahlou**, **S.**, **Pal**, **S.**, **Castro**, **P. S.**, & **Terry**, **J.** (2023). Minigrid & miniworld: Modular & customizable reinforcement learning environments for goal-oriented tasks. *Advances in Neural Information Processing Systems 36, New Orleans, LA, USA*.

Clement, **B.**, **Roy**, **D.**, **Oudeyer**, **P.-Y.**, & **Lopes**, **M.** (2015). Multi-armed bandits for intelligent tutoring systems. *Journal of Educational Data Mining*, *7*(2), 20–48.

Cooper, **G. F.** (1990). The computational complexity of probabilistic inference using Bayesian belief networks. *Artificial Intelligence*, *42*(2-3), 393–405. https://doi.org/10.1016/0004-3702(90)90060-D

Dechter, **R.** (1996). Bucket elimination: A unifying framework for probabilistic inference. In **E. Horvitz** & **F. Jensen** (Eds.), *Proceedings of the twelfth conference on uncertainty in artificial intelligence (uai '96)* (pp. 211–219). Morgan Kaufmann Publishers Inc.

Dempster, **A. P.**, **Laird**, **N. M.**, & **Rubin**, **D. B.** (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B (Methodological)*, *39*(1), 1–38.

Diaz, **M.**, **Paull**, **L.**, & **Tacchetti**, **A.** (2024). Rethinking teacher-student curriculum learning through the cooperative mechanics of experience. *arXiv preprint arXiv:2404.03084*.

Duan, **Y.**, **Chen**, **X.**, **Houthooft**, **R.**, **Schulman**, **J.**, & **Abbeel**, **P.** (2016). Benchmarking deep reinforcement learning for continuous control. *International conference on machine learning*, 1329–1338.

Finn, **C.**, **Abbeel**, **P.**, & **Levine**, **S.** (2017). Model-agnostic meta-learning for fast adaptation of deep networks. *International Conference on Machine Learning*, 1126–1135.

Flet-Berliac, **Y.**, **Ferret**, **J.**, **Pietquin**, **O.**, **Preux**, **P.**, & **Geist**, **M.** (2021). Adversarially guided actor-critic. *arXiv preprint arXiv:2102.04376*.

Florensa, **C.**, **Held**, **D.**, **Geng**, **X.**, & **Abbeel**, **P.** (2018). Automatic goal generation for reinforcement learning agents. *Proceedings of the 35th International Conference on Machine Learning*, 1515–1528.

Florensa, **C.**, **Held**, **D.**, **Wulfmeier**, **M.**, **Zhang**, **M.**, & **Abbeel**, **P.** (2017). Reverse curriculum generation for reinforcement learning. *Conference on robot learning*, 482–495.

Geiger, **D.**, & **Heckerman**, **D.** (1994). Learning gaussian networks. In **R. L.** de **Mantaras** & **D. Poole** (Eds.), *Proceedings of the tenth conference on uncertainty in artificial intelligence (uai '94)* (pp. 235–243). Morgan Kaufmann Publishers

Geman, **S.**, & **Geman**, **D.** (1984). Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *PAMI-6*(6), 721–741. https://doi.org/10.1109/TPAMI.1984.4767596

Graves, **A.**, **Bellemare**, **M. G.**, **Menick**, **J.**, **Munos**, **R.**, & **Kavukcuoglu**, **K.** (2017). Automated curriculum learning for neural networks. *Proceedings of the 34th International Conference on Machine Learning*, 1311–1320.

Gu, **S.**, **Holly**, **E.**, **Lillicrap**, **T.**, & **Levine**, **S.** (2017). Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. *2017 IEEE international conference on robotics and automation (ICRA)*, 3389–3396.

Gupta, **K.**, & **Najjaran**, **H.** (2021). Curriculum-based deep reinforcement learning for adaptive robotics: A mini-review. *Int. Journal of Robotic Engineering*, *6*.

Gupta, **K.**, **Mukherjee**, **D.**, & **Najjaran**, **H.** (2022). Extending the capabilities of reinforcement learning through curriculum: A review of methods and applications. *SN Computer Science*, *3*, 1–18.

Heckerman, **D.** (1995). *A tutorial on learning with Bayesian networks* (tech. rep. No. MSR-TR-95-06). Microsoft Research. Redmond, WA, USA.

Hinton, **G. E.**, & **Salakhutdinov**, **R. R.** (2006). Reducing the dimensionality of data with neural networks. *Science*, *313*(5786), 504–507. https://doi.org/10.1126/science.1127647

Jensen, **F. V.** (1996). *An introduction to bayesian networks*. Springer-Verlag.

Jordan, **M. I.**, **Ghahramani**, **Z.**, **Jaakkola**, **T. S.**, & **Saul**, **L. K.** (1999). An introduction to variational methods for graphical models. *Machine Learning*, *37*(2), 183–233. https://doi.org/10.1023/A:1007665907178

Justesen, **N.**, **Torrado**, **R. R.**, **Bontrager**, **P.**, **Khalifa**, **A.**, **Togelius**, **J.**, & **Risi**, **S.** (2018). Illuminating generalization in deep reinforcement learning through procedural level generation. *NeurIPS 2018 Workshop on Deep Reinforcement Learning*.

Kahn, **G.**, **Villaflor**, **A.**, **Ding**, **B.**, **Abbeel**, **P.**, & **Levine**, **S.** (2018). Self-supervised deep reinforcement learning with generalized computation graphs for robot navigation. *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 5129–5136.

Kingma, **D. P.**, & **Welling**, **M.** (2014). Auto-encoding variational Bayes. *Proceedings of the 2nd International Conference on Learning Representations (ICLR)*. https://arxiv.org/abs/1312.6114

Koller, **D.**, & **Friedman**, **N.** (2009). *Probabilistic graphical models: Principles and techniques*. MIT press.

Lauritzen, **S. L.** (1995). The EM algorithm for graphical association models with missing data. *Computational Statistics & Data Analysis*, *19*(2), 191–201. https://doi.org/10.1016/0167-9473(93)E0056-A

Lee, **S.**, **Cho**, **D.**, **Park**, **J.**, & **Kim**, **H. J.** (2024). Cqm: Curriculum reinforcement learning with a quantized world model. *Advances in Neural Information Processing Systems*, *36*.

Levine, **S.**, **Finn**, **C.**, **Darrell**, **T.**, & **Abbeel**, **P.** (2016). End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, *17*(1), 1334–1373.

Li, **Q.**, **Zhai**, **Y.**, **Ma**, **Y.**, & **Levine**, **S.** (2023). Understanding the complexity gains of single-task rl with a curriculum. *International Conference on Machine Learning*, 20412–20451.

Linden, **R.**, **Lopes**, **R.**, & **Bidarra**, **R.** (2013). Designing procedurally generated levels. *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, *9*, 41–47.

Marcot, **B. G.**, **Steventon**, **J. D.**, **Sutherland**, **G. D.**, & **McCann**, **R. K.** (2006). Guidelines for developing and updating bayesian belief networks applied to ecological modeling and conservation. *Canadian Journal of Forest Research*, *36*(12), 3063–3074.

Margolis, **G. B.**, **Yang**, **G.**, **Paigwar**, **K.**, **Chen**, **T.**, & **Agrawal**, **P.** (2024). Rapid locomotion via reinforcement learning. *The International Journal of Robotics Research*, *43*(4), 572–587.

Matiisen, **T.**, **Oliver**, **A.**, **Cohen**, **T.**, & **Schulman**, **J.** (2019). Teacher-student curriculum learning. *IEEE Transactions on Neural Networks and Learning Systems*, *31*(9), 3732–3740. https://doi.org/10.1109/TNNLS.2019.2934906

Mnih, **V.**, **Kavukcuoglu**, **K.**, **Silver**, **D.**, **Rusu**, **A. A.**, **Veness**, **J.**, **Bellemare**, **M. G.**, **Graves**, **A.**, **Riedmiller**, **M.**, **Fidjeland**, **A. K.**, **Ostrovski**, **G.**, et al. (2015). Human-level control through deep reinforcement learning. *Nature*, *518*(7540), 529–533. https://doi.org/10.1038/nature14236

Narvekar, **S.**, **Peng**, **B.**, **Leonetti**, **M.**, **Sinapov**, **J.**, **Taylor**, **M. E.**, & **Stone**, **P.** (2020). Curriculum learning for reinforcement learning domains: A framework and survey. *Journal of Machine Learning Research*, *21*(181), 1–50.

Newton, **A. C.** (2010). Use of a bayesian network for red listing under uncertainty. *Environmental Modelling & Software*, *25*(1), 15–23.

Niu, **Y.**, **Jin**, **S.**, **Zhang**, **Z.**, **Zhu**, **J.**, **Zhao**, **D.**, & **Zhang**, **L.** (2023). Goats: Goal sampling adaptation for scooping with curriculum reinforcement learning. *2023 IEEE/RSJ IROS*, 1023–1030.

OpenAI, **Berner**, **C.**, **Brockman**, **G.**, **Chan**, **B.**, **Cheung**, **V.**, **Dębiak**, **P.**, **Dennison**, **C.**, **Farhi**, **D.**, **Fischer**, **Q.**, **Hashme**, **S.**, **Hesse**, **C.**, **Józefowicz**, **R.**, **Gray**, **S.**, **Olsson**, **C.**, **Pachocki**, **J.**, **Petrov**, **M.**, **Pinto**, **H. P. d. O.**, **Raiman**, **J.**, **Salimans**, **T.**, … **Zhang**, **S.** (2019). Dota 2 with large scale deep reinforcement learning.

Parisi, **S.**, **Dean**, **V.**, **Pathak**, **D.**, & **Gupta**, **A.** (2021). Interesting object, curious agent: Learning task-agnostic exploration. *Advances in Neural Information Processing Systems*, *34*, 20516–20530.

Pathak, **D.**, **Agrawal**, **P.**, **Efros**, **A. A.**, & **Darrell**, **T.** (2017). Curiosity-driven exploration by self-supervised prediction. *Proceedings of the 34th International Conference on Machine Learning*, 2778–2787.

Pearl, **J.** (1988). *Probabilistic reasoning in intelligent systems: Networks of plausible inference*. Morgan Kaufmann.

Portelas, **R.**, **Colas**, **C.**, **Hofmann**, **K.**, & **Oudeyer**, **P.-Y.** (2020a). Automatic curriculum learning for deep RL: A short survey. *International Joint Conference on Artificial Intelligence*, 4819–4825. https://doi.org/10.24963/ijcai.2020/673

Portelas, **R.**, **Colas**, **C.**, **Hofmann**, **K.**, & **Oudeyer**, **P.-Y.** (2020b). Teacher algorithms for curriculum learning of deep rl in continuously parameterized environments. *Conference on Robot Learning*, 835–853.

Raffin, **A.**, **Hill**, **A.**, **Gleave**, **A.**, **Kanervisto**, **A.**, **Ernestus**, **M.**, & **Dormann**, **N.** (2021). Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, *22*(268), 1–8. http://jmlr.org/papers/v22/20-1364.html

Rusu, **A. A.**, **Rabinowitz**, **N. C.**, **Desjardins**, **G.**, **Soyer**, **H.**, **Kirkpatrick**, **J.**, **Kavukcuoglu**, **K.**, **Pascanu**, **R.**, & **Hadsell**, **R.** (2016). Progressive neural networks. *Advances in Neural Information Processing Systems*, *29*.

Sayar, **E.**, **Iacca**, **G.**, & **Knoll**, **A.** (2024). Curriculum learning for robot manipulation tasks with sparse reward through environment shifts. *IEEE Access*.

Schulman, **J.**, **Levine**, **S.**, **Abbeel**, **P.**, **Jordan**, **M.**, & **Moritz**, **P.** (2015). Trust region policy optimization. *Proceedings of the 32nd International Conference on Machine Learning*, 1889–1897.

Schulman, **J.**, **Wolski**, **F.**, **Dhariwal**, **P.**, **Radford**, **A.**, & **Klimov**, **O.** (2017). Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Schulman, **J.**, **Moritz**, **P.**, **Levine**, **S.**, **Jordan**, **M. I.**, & **Abbeel**, **P.** (2018). High-dimensional continuous control using generalized advantage estimation.

Schwarz, **G. E.** (1978). Estimating the dimension of a model. *The Annals of Statistics*, *6*(2), 461–464. https://doi.org/10.1214/aos/1176344136

Shortreed, **S. M.**, **Laber**, **E.**, **Lizotte**, **D. J.**, **Stroup**, **T. S.**, **Pineau**, **J.**, & **Murphy**, **S. A.** (2011). Informing sequential clinical decision-making through reinforcement learning: An empirical study. *Machine learning*, *84*(1-2), 109–136.

Silver, **D.**, **Schrittwieser**, **J.**, **Simonyan**, **K.**, **Antonoglou**, **I.**, **Huang**, **A.**, **Guez**, **A.**, **Hubert**, **T.**, **Baker**, **L.**, **Lai**, **M.**, **Bolton**, **A.**, et al. (2017). Mastering the game of Go without human knowledge. *Nature*, *550*(7676), 354–359. https://doi.org/10.1038/nature24270

Silver, **D.**, **Hubert**, **T.**, **Schrittwieser**, **J.**, **Antonoglou**, **I.**, **Lai**, **M.**, **Guez**, **A.**, **Lanctot**, **M.**, **Sifre**, **L.**, **Kumaran**, **D.**, **Graepel**, **T.**, et al. (2018). A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, *362*(6419), 1140–1144.

Spirtes, **P.**, **Glymour**, **C. N.**, & **Scheines**, **R.** (2000). *Causation, prediction, and search* (Second). MIT Press.

Sukhbaatar, **S.**, **Lin**, **Z.**, **Kostrikov**, **I.**, **Synnaeve**, **G.**, **Szlam**, **A.**, & **Fergus**, **R.** (2017). Intrinsic motivation and automatic curricula via asymmetric self-play. *arXiv preprint arXiv:1703.05407.*

Sukhbaatar, **S.**, **Szlam**, **A.**, **Synnaeve**, **G.**, **Chintala**, **S.**, & **Fergus**, **R.** (2015). Mazebase: A sandbox for learning from games. *arXiv preprint arXiv:1511.07401.*

Sutton, **R. S.**, & **Barto**, **A. G.** (2018). *Reinforcement learning: An introduction*. MIT Press.

Szoke, **L.**, **Shperberg**, **S. S.**, **Holtz**, **J.**, & **Allievi**, **A.** (2024). Adaptive curriculum learning with successor features for imbalanced compositional reward functions. *IEEE Robotics and Automation Letters*, *9*(6), 5174–5181. https://doi.org/10.1109/LRA.2024.3387134

Tabibian, **B.**, **Upadhyay**, **U.**, **De**, **A.**, **Zarezade**, **A.**, **Schölkopf**, **B.**, & **Gomez-Rodriguez**, **M.** (2019). Enhancing human learning via spaced repetition optimization. *Proceedings of the National Academy of Sciences*, *116*(10), 3988–3993.

Uchendu, **I.**, **Xiao**, **T.**, **Lu**, **Y.**, **Zhu**, **B.**, **Yan**, **M.**, **Simon**, **J.**, **Bennice**, **M.**, **Fu**, **C.**, **Ma**, **C.**, **Jiao**, **J.**, et al. (2023). Jump-start reinforcement learning. *International Conference on Machine Learning*, 34556–34583.

van der **Maaten**, **L.**, & **Hinton**, **G.** (2008). Visualizing data using t-sne. *Journal of Machine Learning Research*, *9*(Nov), 2579–2605.

Vinyals, **O.**, **Babuschkin**, **I.**, **Czarnecki**, **W. M.**, **Mathieu**, **M.**, **Dudzik**, **A.**, **Chung**, **J.**, **Choi**, **D. H.**, **Powell**, **R.**, **Ewalds**, **T.**, **Georgiev**, **P.**, et al. (2019). Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature*, *575*(7782), 350–354. https://doi.org/10.1038/s41586-019-1724-z

Voinov, **A.**, & **Bousquet**, **F.** (2010). Modelling with stakeholders. environmetal modelling and software, 25, 1268-1281. *Refereed journal papers*.

Wan, **S.**, **Tang**, **Y.**, **Tian**, **Y.**, & **Kaneko**, **T.** (2023). Deir: Efficient and robust exploration through discriminative-model-based episodic intrinsic rewards. *arXiv preprint arXiv:2304.10770*.

Wang, **X.**, **Chen**, **Y.**, & **Zhu**, **W.** (2021). A survey on curriculum learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Yu, **C.**, **Liu**, **J.**, & **Nemati**, **S.** (2021). Reinforcement learning in healthcare: A survey. *ACM Computing Surveys (CSUR)*, *55*(1), 1–36.

Zha, **D.**, **Ma**, **W.**, **Yuan**, **L.**, **Hu**, **X.**, & **Liu**, **J.** (2021). Rank the episodes: A simple approach for exploration in procedurally-generated environments. *arXiv preprint arXiv:2101.08152*.

YÖK

CURRICULUM VITAE

Name SURNAME: Onur AKGÜN

EDUCATION:

- **B.Sc.:** 2015, Kocaeli University, Department of Mechatronics Engineering
- **M.Sc.:** 2018, Yıldız Technical University, Department of Control and Automation Engineering

PROFESSIONAL EXPERIENCE AND REWARDS:

- 2017-present Research Assistant, Department of Mechatronics Engineering, Turkish-German University, Istanbul, Türkiye
- 2015-2016 Engineer, Research and Development Department, Elektek Ltd., Istanbul, Türkiye

PUBLICATIONS, PRESENTATIONS AND PATENTS ON THE THESIS:

- **Akgün, O.** & **Üre, N. K.** (2025). Bayesian curriculum generation in sparse reward reinforcement learning environments. *Engineering Science and Technology, an International Journal*, vol. 66, p. 102048.