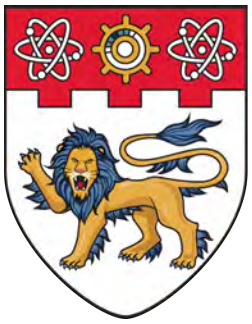


NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

SOLVING LARGE-SCALE PLANNING AND DEEP LEARNING PROBLEMS

AYE PHYU PHYU AUNG

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

AUGUST, 2022

SOLVING LARGE-SCALE PLANNING AND DEEP LEARNING PROBLEMS

AYE PHYU PHYU AUNG

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

A thesis submitted to the Nanyang Technological University

in partial fulfillment of the requirement for the degree of

Doctor of Philosophy

AUGUST, 2022

Statement of Originality

I hereby certify that the work embodied in this thesis is the result of original research, is free of plagiarised materials, and has not been submitted for a higher degree to any other University or Institution.

August 30, 2022

Date

Aye Phyu Phyu Aung

Supervisor Declaration Statement

I have reviewed the content and presentation style of this thesis and declare it is free of plagiarism and of sufficient grammatical clarity to be examined. To the best of my knowledge, the research and writing are those of the candidate except as acknowledged in the Author Attribution Statement. I confirm that the investigations were conducted in accord with the ethics policies and integrity standards of Nanyang Technological University and that the research data are presented honestly and without prejudice.

Type text here

August 30, 2022

Date

Bo An

Authorship Attribution Statement

This thesis contains materials from the four papers that are published/ under-review in the following peer-reviewed conferences and journals where I was the first and/or corresponding author.

Chapter 3 is published as **Aye Phyu Phyu Aung**, Xinrun Wang, Bo An, Xiaoli Li. We mind your well-being: Preventing depression in uncertain social networks by sequential interventions. *Proceedings of the 30th International Conference on Automated Planning and Scheduling* (**ICAPS'20**), pp.499-507. The contributions of the co-authors are as follows:

- Prof. Bo An provided the initial idea of this work;
- Dr. Xinrun Wang and I investigated the problems and built the model;
- I justified the model, proposed and implemented the algorithms;
- I wrote the manuscript and Dr. Xinrun Wang revised the manuscript;
- Prof. Bo An, Dr. Xiaoli Li provided critical comments to the paper.

Chapter 4 is published as **Aye Phyu Phyu Aung**, Senthilnath Jayavelu, Xiaoli Li, Bo An. Planning sequential interventions to tackle depression in large uncertain social networks using deep reinforcement learning. *Neurocomputing 481*: 182-192 (2022) The contributions of the co-authors are as follows:

- I provided the idea to extend the work in Chapter 3 to this work;
- Dr. Senthilnath Jayavelu and I investigated the problems, built and justified the model;
- I proposed and implemented the algorithms and conducted the experiments;
- I wrote the manuscript and Prof. Bo An and Dr. Xiaoli Li revised the paper.

Chapter 5 is published as **Aye Phyu Phyu Aung**, Xinrun Wang, Runsheng Yu, Bo An, Senthilnath Jayavelu, Xiaoli Li. DO-GAN: A double oracle framework for generative adversarial networks. *Proceedings of the 2022 IEEE Conference on Computer Vision and Pattern Recognition* (**CVPR'22**). The contributions of the co-authors are as follows:

- Dr. Xinrun Wang proposed the problem and the model;
- Dr. Runsheng Yu provided critical comments to the model and I justified the model;

- I proposed, implemented the algorithms and wrote the manuscript;
- Dr. Xinrun Wang and Dr. Senthilnath Jayavelu revised the algorithm and provided suggestions.
- Prof. Bo An, Dr. Runsheng Yu and Dr. Xinrun Wang provided insightful comments on the manuscript;
- Prof. Bo An, Dr. Senthilnath Jayavelu and Dr. Xiaoli Li revised the paper.

Chapter 6 under the review as **Aye Phyu Phyu Aung**, Xinrun Wang, Hau Chan, Bo An, Senthilnath Jayavelu, Xiaoli Li. DONAS: Double Oracle and Neural Architecture Search for Adversarial Machine Learning. *36th Conference on Neural Information Processing Systems* (**NEURIPS'22**). The contributions of the co-authors are as follows:

- Dr. Xinrun Wang and I proposed the problem and the model;
- Dr. Hau Chan provided critical comments to the model and I justified the model;
- I proposed, implemented the algorithms, ran the experiments and wrote the manuscript;
- Dr. Xinrun Wang and Dr. Senthilnath Jayavelu revised the algorithm and provided suggestions.

- Prof. Bo An, and Dr. Xinrun Wang, Dr. Senthilnath Jayavelu provided insightful comments on the manuscript;
- Prof. Bo An, Dr. Senthilnath Jayavelu and Dr. Xiaoli Li revised the paper.

August 30, 2022

Date

Aye Phyu Phyu Aung

Acknowledgements

This four-year Ph.D. journey is an eventful one for both my career and personal life. During these four years, a lot of eventful moments taught me to be a professional researcher, a more grown-up who can properly work with other wonderful researchers and teammates, a mature person who can make life-changing decisions calmly and when I had to let some things go.

First and foremost, I am greatly indebted to my NTU supervisor, Professor Bo An. Without his help and suggestion to work as a research officer first and apply for the scholarship, I would not even be doing a Ph.D. in the first place. Both during my time as a research officer and four-year Ph.D., he has provided me with insightful discussions as well as suggestions and allowed to explore interesting research problems. Moreover, for a student from Myanmar who does not have the best language ability, he suggested a lot of different solutions so that I can communicate with other researchers and seniors from my team. I am very grateful for his patience in helping me until I can show results. His great ability in research and sense of important topics inspire me to work on important problems and present them effectively.

Secondly, I would like to thank my A*STAR I2R supervisors Dr. Xiaoli Li and Dr. Senthilnath Jayavelu for providing the suitable research environment at A*STAR as well as for their insightful guidance and comments on my research proposals, writing/ presentation styles and official matters so that my four-year Ph.D. would go smoothly.

Thirdly, I would like to especially thank my senior Dr. Xinrun Wang for being very patient being my collaborator in my research work. I initially had several difficulties discussing and starting my research work. It is for his guidance and feedback that I finally am able to learn how to professionally do research work. I would like to express my sincere gratitude to him.

I am also very fortunate to meet and learn from a group of talented people: Qingyu Guo, Wanyuan Wang, Youzhi Zhang, Xinrun Wang, Lei Feng, Aye Phyu, Hongxin Wei, Rundong Wang, Wei Qiu, Yakub Cerny, Runsheng Yu, Yanchen Deng, Shuxin Li, Wanqi Xue, Fei Fei Lin, Zhuyun Yin, Xinyu Cai, Shuo Sun, Pengdeng Li, Shenggong Ji and Shufeng Kong. I would like to thank all of them for the warm support they have given me.

I would also like to thank my high school Physics teacher Dr. Htay Htay Naing for inspiring me by showing her expansive knowledge of her subject. Her meticulous answers to all my questions made me want to pursue a Ph.D. degree like her.

I want to thank my friends and sister who were around me in my difficult times. Wai Yan, Su Wai Yee Aung, Thawtar Lynn, Myat Kay Khine Tun, Ya Min Nyi Nyi, Shwe Soe Chun, Cho Zin Tun, Phone Myint Han, Thin Thin Aung and Su Myat Naing Aung.

Finally, I would like to thank my parents, Aung Lwin and Soe Soe Myint, who are away but always sending their love and support to me. This thesis is dedicated to all of you. Thank you very much.

List of Figures

1.1 High-level hierarchy of the thesis structure 9

3.1 An illustrative example with 5 students. At round t, UCC knows the influence between the students w_{12} and w_{23}, represented by the solid lines. Influence unknown by UCC are represented by the dashed lines. Student 3 is picked by UCC to be intervened at t. w_{34} and w_{35} are known by UCC and students 4 and 5 are also influenced according to Eq. (3.1). . . 29

3.2 An illustrative example of reasoning process. When UCC intervenes student i, UCC observes the value of her mental state and influence between her neighbours shown in red. Using this observation, we predict the ranges of j, k, l in blue with observed v_i and the ranges from their other in-neighbours in magenta. The predicted ranges are in green. Using these ranges, we calculate $[lb_i, ub_i]$ for i, the j, k, l's ranges are modified if $v_i \notin [lb_i, ub_i]$. The final ranges for j, k, l are shown in blue. . . 37

3.3 Scalability comparison of MLPRAP variants with DC, HEAL (Figure 3.3a and 3.3b are plotted in log-scale) 47

3.4 Solution quality comparison of MLPRAP variants with DC, HEAL 48

3.5 Experiments on real networks 50

4.1 DRLPSO Architecture 54

4.2 DRLPSO components represented as Markov Model commonly used in RL 56

4.3 Peak convergence by varying the number of particles 64

4.4 Performance of learning algorithms in 100k iterations 65

4.5 Overview of the implementation plan 68

4.6 Intervention process: Step 1,6 are planned by the algorithm while Step 2-5 are performed by the counsellors 70

5.1 Training images with fixed noise for SGAN and DO-SGAN/P (pruning) until termination. Both during training and after convergence, DO-SGAN/P can generate better quality images with FID score of 6.32 against SGAN's FID score of 6.98. . . . 73
5.2 An illustration of DO-GAN. Figure adapted from [48]. 75
5.3 Comparison of GAN and DO-GAN/P on 2D synthetic Gaussian Mixture Dataset 86
5.4 Full comparison of GAN and DO-GAN/P on 2D Synthetic Gaussian Dataset 87
5.5 Training evolution on 2D Gaussian Dataset with $s = 5, 10, 15$. 88
5.6 Taxonomy of GAN Architectures from [99] 89
5.7 Training images with fixed noise for DCGAN and DO-DCGAN/P until termination. 90
5.8 Training images with fixed noise for SNGAN and DO-SNGAN/P until termination. 92
5.9 FID score vs. Epochs for SGAN and DO-SGAN trained on CIFAR-10 94

6.1 The CIFAR-10 images randomly generated from DONAS for GAN 116
6.2 (a): Linear CKA between layers of the individual searched networks of DONAS-GAN trained on TinyImageNet dataset. The downtrend of similarity for later layers is observed. (b): Linear CKA averaging between the same layers of the searched networks of DONAS-GAN trained on TinyImageNet. 117
6.3 The search process of DONAS-GAN obtaining diverse architectures. 118

List of Tables

2.1 Two-player two-action normal form game [12] 20
2.2 Normal form game with no pure strategy equilibrium 21
2.3 Comparison of Terminologies between Game Theory and GAN 23

4.1 Reward Comparison for MLPRAP 64
4.2 Reward Comparison for DRLPSO 67

5.1 Hyperparameters used in DO-GAN for different base architectures 85
5.2 Runtime of DO-GAN/P on 2D Gaussian Dataset with $s = 5, 10, 15$ 87
5.3 Inception scores (higher is better) and FID scores (lower is better) of DO-GAN with pruning (DO-GAN/P) and continual learning (DO-GAN/C). The mean and standard deviation are drawn from running 10 splits on 10000 generated images. . . . 93
5.4 Wall-clock time comparison of SGAN variants 95

6.1 Generative results for DONAS-GAN. 114
6.2 Robust accuracy under FGSM and PGD attacks. 119

Contents

1 Introduction **1**
- 1.1 Student Counselling Problem as Large-scale Games 3
  - 1.1.1 Planning Approach 3
  - 1.1.2 Learning Approach 4
- 1.2 Generative Adversarial Networks (GAN) as Large-scale Games 5
  - 1.2.1 Double Oracle Framework for GANs 6
  - 1.2.2 Double Oracle Framework for Adversarial Machine Learning (AML) 7
- 1.3 Thesis Overview 8

2 Literature Review **10**
- 2.1 Correlation of Stress and Risk of Depression 11
- 2.2 Influence Maximization 13
- 2.3 Adversarial Machine Learning 14
  - 2.3.1 Generative Adversarial Networks (GAN) 14
  - 2.3.2 Adversarial Training (AT) 16
  - 2.3.3 Neural Architecture Search (NAS) for AML 17
- 2.4 Online POMDP Solvers 18
- 2.5 Deep Recurrent Q-learning Architectures 19
- 2.6 Game Theory 19
  - 2.6.1 Normal-Form Game and Nash Equilibrium 19
  - 2.6.2 Double Oracle Algorithm 22

3 We Mind Your Well-being: Preventing Depression in Uncertain Social Networks by Sequential Intervention **24**
- 3.1 Motivation Scenario 27
- 3.2 Model 27
  - 3.2.1 Network and Dynamics 28

3.2.2 Uncertainties . . . 30
3.2.3 POMDP Formulation . . . 31
3.3 MLPRAP . . . 33
3.3.1 Reasoning . . . 35
3.3.2 Abstraction of POMDP States . . . 40
3.3.3 Multi-Level Partitions . . . 43
3.4 Experimental Evaluation . . . 44
3.4.1 Experiment Setup . . . 45
3.4.2 Experiment Results . . . 47
3.5 Chapter Summary . . . 51

4 Planning Sequential Interventions to Tackle Depression in Large Uncertain Social Networks Using Deep Reinforcement Learning 52
4.1 Deep Reinforcement Learning with Particle Swarm Optimization 54
4.1.1 DRLPSO Architecture . . . 55
4.1.2 DRLPSO Algorithm . . . 57
4.2 Experiment Results . . . 61
4.2.1 Experiment Setup . . . 62
4.2.2 Evaluation . . . 63
4.3 Discussion . . . 68
4.4 Chapter Summary . . . 70

5 A Double-Oracle Framework for Generative Adversarial Networks (DO-GAN) 72
5.1 DO-GAN . . . 75
5.1.1 General Framework of DO-GAN . . . 75
5.1.2 Linear Program for Meta-matrix Game . . . 79
5.1.3 Termination Check . . . 79
5.2 Practical Implementations . . . 80
5.2.1 Meta-matrix Pruning (DO-GAN/P) . . . 81
5.2.2 Continual Learning (DO-GAN/C) . . . 82
5.3 Experiments . . . 84
5.3.1 Value of λ . . . 85
5.3.2 Evaluation on 2D Gaussian Mixture Dataset . . . 86
5.3.3 Full Training Process of 2D Gaussian Dataset . . . 87

5.3.4 Choice of GAN Architectures for Experiments 88
5.3.5 Evaluation on Real-world Datasets 89
5.4 Chapter Summary 95

6 DONAS: Double-Oracle and Neural Architecture Search for Adversarial Machine Learning 97
6.1 DONAS: The General Framework 100
6.2 Spices of DONAS 103
6.2.1 DONAS for GAN 103
6.2.2 DONAS for AT 109
6.3 Experiments 112
6.3.1 Experimental Setup 112
6.3.2 Experiments on DONAS for GAN 113
6.3.3 Experiments on DONAS for AT 118
6.4 Chapter Summary 120

7 Conclusion 122
7.1 Conclusion 122
7.2 Future Work 125
7.2.1 Reducing Training Time for Neural Architecture Search 126
7.2.2 DO-GAN/ DONAS for Time Series Datasets 127
7.2.3 Hybrid (discrete-continuous) Action Space 127

Solving Large-scale Planning and Deep Learning Problems

Aye Phyu Phyu Aung

Supervisors: Prof. Bo An, Dr. Xiaoli Li

Abstract

Game theory has been researched and applied in many scenarios. However, the state, action space and time of most games are set as discrete to find the optimal strategy. Hence, the primary focus of the research will be on solving problems with large-scale action space as the direct usage of existing small or discrete solutions limits the solution quality and brings less resemblance to the increasingly complex real-life situations. In particular, we approach planning: student counselling problem with large discrete action space and deep learning problem: GAN with continuous action space. Then, we propose two solutions for the counselling problem: 1) Planning Approach and 2) Learning Approach as well as two solutions for GAN: 1) Double Oracle framework for GAN (DO-GAN) and 2) Double Oracle and Neural Architecture Search for Adversarial Machine Learning (DONAS). Finally, we conduct extensive experiments to show significant improvement of our solution quality against state-of-the-art algorithms.

Chapter 1

Introduction

Planning and game theory research have been applied in many scenarios to important problems in modern society such as security games, financial markets, resource allocation, cyber security and mechanism designs [1, 84, 92, 97]. To apply game theory algorithms to real-life scenarios, the early research works approached the problems with small or discrete state and action space and find the optimal strategy [16, 105]. However, as the real-life situations are becoming increasingly complex, direct usage of existing solutions limits the solution quality of the equilibrium and brings less resemblance to the actual situations [98].

Therefore, it is important to model the problems with large state and action space to properly reflect the real-life situations in planning and game theory domains. By doing so, the algorithm will be able to optimally or efficiently choose the strategy to allocate the limited resources. This is challenging mainly due to the complexity from the large action space (exponential

growth of available strategies with several choices or continuous action space) of the players in the problems. This needs the complex, high-performance computing capabilities to efficiently choose the optimal strategy. Some problems might even need large memory allocations, making the problems very difficult to solve. Several other issues such as various uncertainties in the real world make the problem even more challenging.

Hence, this thesis proposes to adapt and improve multiple algorithms and approaches from planning, learning, and game theory disciplines to not only train the large discrete state and action space but also extend to the environments with continuous state and action space.

Firstly, we approach a large-discrete problem i.e., the student counselling problem which has a large number of participants. The action space grows exponentially with the number of added participants in the network and thus, makes the problem very difficult to solve. Moreover, the solution does not have complete information on the initial states of the participants. This uncertainty makes the problem more difficult to converge to the Nash Equilibrium (NE) strategy.

Secondly, we approach a continuous large-scale problem: Generative Adversarial Networks (GANs). GANs have been applied in various domains such as image and video generation, image-to-image translation and text-to-image synthesis [58, 79, 80, 81]. Various architectures are proposed to generate more realistic samples [65, 75, 77] as well as regularization techniques [2, 67].

GANs can be seen as a two-player zero-sum game of infinite state and action choices with generator and discriminator as the two players [24]. They are alternately trained to maximize their respective utilities till convergence corresponding to a pure Nash Equilibrium (NE). In this setting, GANs are challenging as a pure Nash equilibrium may not exist [17, 63] and even finding the mixed Nash equilibrium is difficult as GANs have a large strategy space. In the rest part of this chapter, we present a brief review of the problems, the models, and the contributions made in this thesis.

1.1 Student Counselling Problem as Large-scale Games

The first two parts of this thesis focus on the student counselling problem which has a large-discrete action set that grows exponentially with the number of students in the network. Along with mental health becoming a major public health concern worldwide, we aim to help the social support organizations (e.g., university counselling centers) sequentially select the participants for interventions. Our two major contributions are as follows.

1.1.1 Planning Approach

Firstly, we adapt a planning approach to this problem. SOTA approaches like HEAL only handle the state values as $[0, 1]$. Thus, solutions adapting SOTA cannot solve networks with more than 30 students whose mental states are represented by a range of many values. The uncertainties in the network

also cause a loss in solution quality. Thus, we propose a model to address the sequential intervention of participants while considering the emotion propagation and formulate it as a Partially Observable Markov Decision Process (POMDP) to handle uncertainties about their mental states and the influence between them. We then apply reasoning to refine belief to improve solution quality for the lack of initial information on the mental state values. We improve the scalability by the abstraction of states to reduce the number of states by representing the mental states with an abstracted discrete set. We further improve the scalability by multi-level partitioning to get smaller POMDPs. We also conduct extensive experiments on both synthetic and real networks to show that our algorithm significantly improves scalability with comparable solution quality compared to the state-of-the-art algorithms.

1.1.2 Learning Approach

Secondly, we adapt a learning approach to the same problem by proposing an algorithm with Deep Reinforcement Learning (DRL). However, the partitioning step in the planning approach breaks up the student network causing information loss. Using RL to prevent information loss, the traditional DRL approaches cannot handle belief and observation as well as LSTM cannot handle the complex large action space. Adding Discrete Particle Swarm Optimization (DPSO) to optimize the action space solves the issue. Hence, we propose DRL with Particle Swarm Optimization (DRLPSO) to plan the op-

timal sequential actions in the uncertain dynamic environment with a large action space. In particular, DRLPSO consists of two stages: the Discrete Particle Swarm Optimization (DPSO) and Deep Q-learning integrated with Long Short-Term Memory (DQ-LSTM). In the first stage, we apply DPSO by initializing n particles that converge to multiple optimal actions for each belief state. In the second stage, the action with the best Q-value from the DPSO action set is executed to obtain belief and observation (history of action). We evaluated the proposed method empirically with the simulated student networks with mental state propagation compared to the state-of-the-art algorithms.

1.2 Generative Adversarial Networks (GAN) as Large-scale Games

The following two parts of this thesis focus on Generative Adversarial Networks (GANs) which can be seen as a two-player continuous state/action space problem. From the game-theoretic perspective, GANs can be viewed as a two-player game where the generator samples the data and the discriminator classifies the data as real or generated. Training GANs is challenging as a pure Nash equilibrium (NE) may not exist [17, 63] and even finding the mixed Nash equilibrium is difficult as GANs have a large strategy space. This also leads to unstable training in GANs depending on the data and

the hyperparameters i.e., we have to tune the hyperparameters with every change of datasets for each architecture. To remedy these limitations, mixed NE is a more suitable solution concept [33]. As a mixed NE approach, we consider Double Oracle (DO) algorithm [62] which is a powerful framework to compute mixed NE in large-scale games that has been applied in various disciplines [7, 37, 48]. Our two major contributions are as follows.

1.2.1 Double Oracle Framework for GANs

Inspired by successful applications of the DO framework, we propose DO-GAN by extending the **D**ouble **O**racle framework to **GAN**s. We first generalize the players' strategies as the trained models of generator and discriminator from the best response oracles. We then compute the meta-strategies using a linear program. For scalability of the framework where multiple generators and discriminator best responses are stored in the memory, we propose two solutions: 1) pruning the weakly-dominated players' strategies to keep the oracles from becoming intractable; 2) applying continual learning to retain the previous knowledge of the networks. We apply our framework to established GAN architectures such as vanilla GAN, Deep Convolutional GAN, Spectral Normalization GAN and Stacked GAN. Finally, we conduct experiments on MNIST, CIFAR-10 and CelebA datasets and show that DO-GAN variants have significant improvements in both subjective qualitative evaluation and quantitative metrics, compared with their respective GAN architectures.

1.2.2 Double Oracle Framework for Adversarial Machine Learning (AML)

As the next step, we expand our investigation to Adversarial Machine Learning (AML) which consists of two main schemes: i) generative adversarial networks (GAN) [9, 24, 39], and ii) adversarial training (AT) [25, 61]. Both GAN and AT are essentially two-player zero-sum games between generator/discriminator and classifier/attacker, respectively which are alternately trained to maximize their respective utilities till convergence corresponding to a Nash Equilibrium (NE) [3]. Mixed-architecture approaches such as MIX+GAN [3], MGAN [32] and DO-GAN in Section 1.2.1 as well as NAS approaches such as Adversarial-NAS [20] and AdvRush [69] have also shown promising results for optimizing the network architecture in AML. However, mixed-architectures just use predefined networks, while NAS techniques select only one final network from the search space, thus, limiting the effectiveness of both.

Thus, we propose DONAS, which combines **D**ouble **O**racle from game theory and **N**eural **A**rchitecture **S**earch to allow the multi-model search to mitigate mode-collapse, generating finer results as well as robustness. We extend the DO framework to NAS for GAN and AT without inducing any restriction to NAS to use a mixed-architecture approach with searched networks from NAS. In particular, we first propose a general algorithm for DONAS to adapt the DO framework followed by DONAS for GANs with the respective

oracles to search multiple discriminators and generators and DONAS for AT with the classifier/attacker oracles. Finally, we conduct experiments to evaluate the performance of DONAS for GAN and AT against different existing NAS algorithms. Experiment results show that our approach gives significant improvements to Adversarial-NAS for GANs as well as in its robustness against white-box attacks.

1.3 Thesis Overview

The organization of this thesis is as follows. Chapter 2 discusses the background material and related works of this thesis. Chapter 3 considers the problem of preventing Depression in Uncertain Social Networks by Sequential Interventions planned by Partially Observable Markov Decision Process (POMDP). Chapter 4 solves the same problem with Deep Reinforcement Learning (DRL) approach. Chapter 5 considers the problem of adapting the Double Oracle Framework for Generative Adversarial Networks (GANs). Chapter 6 discusses the Double Oracle for Adversarial Machine Learning, emphasizing Adversarial-Neural Architecture Search (Adversarial-NAS). Chapter 7 concludes this thesis and discusses future directions. The high-level hierarchy of the thesis structure is shown in Figure 1.1.

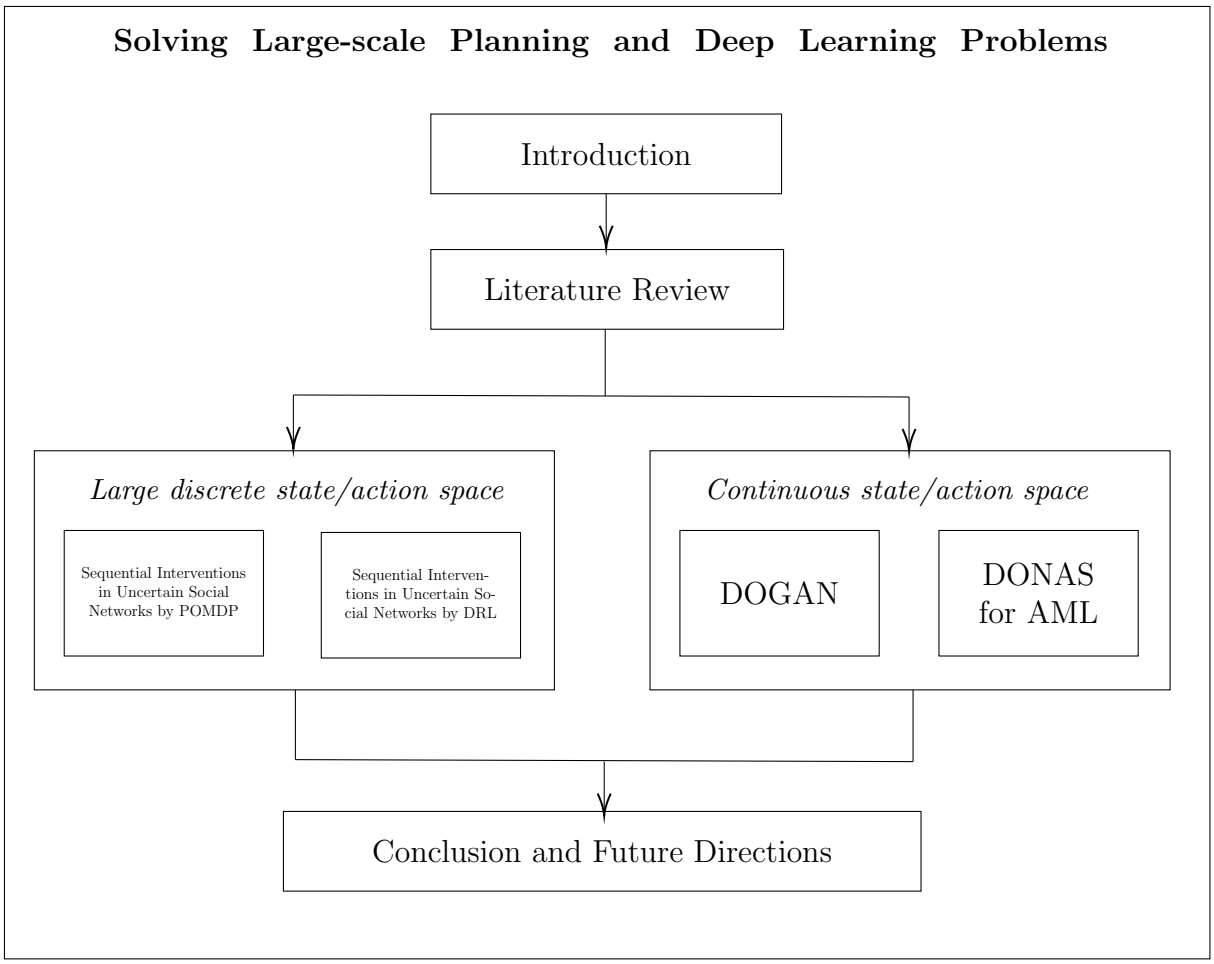


Figure 1.1: High-level hierarchy of the thesis structure

Chapter 2

Literature Review

This chapter reviews the background knowledge which supports this thesis with fundamental theorems, models, and algorithms. The topics discussed in this thesis fall into two main categories: a) the fundamental theorems and research works of the real-world environments to be considered as large-scale problems and b) fundamental models and algorithms used to propose the new algorithms to solve the considered large-scale problems.

We begin with the basic models and theorems related to student counselling problem: Correlation of Stress and Risk of Depression in Section 2.1 and Influence Maximization in Section 2.2. We then discuss the basic models related to Adversarial Machine Learning: Generative Adversarial Networks in Section 2.3.1 and Adversarial Training in Section 2.3.2. Finally, we conclude the fundamentals for considered problems with Neural Architecture Search (NAS) for AML in Section 2.3.3.

Next, we discuss the basic models, theorems, and algorithms of our so-

lutions. We first discuss online POMDP solvers in Section 2.4 to plan the sequential interventions for the student counselling problem. We then discuss Deep Recurrent Q-learning Architectures in Section 2.5 to solve the counselling problem with a deep reinforcement learning algorithm. Afterwards, we discuss basic models and theorems of game theory in Section 2.6 which includes normal-form game, concepts of pure and mixed Nash-equilibrium in Section 2.6.1. Finally, we discuss the double oracle algorithm in Section 2.6.2.

2.1 Correlation of Stress and Risk of Depression

Since our main concern is to prevent depression in a student network of a university, we found that a major factor leading to depression among university students is stress [29]. An epidemiological study also suggests that the most common factors leading to major depression episodes are high stress and low social support [6]. The research works such as [43, 60] support the view that chronic stress leads to anxiety and depression due to stress-induced behavioural and neuroendocrine responses. Thus, to reduce the stress of students and spread the happy mood within the student network, an emotion propagation model will be needed.

It is stated that an increased level of stress with growing economic and social demands in the modern age leads to a rapid rise in depressive symptoms. There has been evidence shown by the studies that emotions spread from person to person via a process known as social contagion, i.e., emotion propaga-

tion. The emotion propagation process is properly established in the following studies. The studies in [19] concluded that a person has a higher probability of being happy if he/she has more friends who are happy. Similarly, another research on a social network of adolescents finds that when a participant has more friends with a worse mental state, it has a higher probability of worsening the mental state of the participant [15]. Moreover, the study also stated that although happiness/stress spreads over social networks, depression does not. Thus, we only model the emotion (happy/ stressed) propagation in the network and do not consider depression in our propagation model.

Hence, using this well-established process as supported by the mentioned literature, we construct our emotion propagation model where a person, after the intervention, spreads his happiness through emotion propagation in the social network and reduces the stress level of others. Since the neighbors' mental states affect a person's mental state both positively and negatively, we considered the happy/ stress emotions of each neighbor in the propagation model. This propagation is one degree from the seed node since the influence propagation does not normally go beyond that in real-world networks [22]. Since the neighbours' mental states affect a person's mental state both positively and negatively [78], we consider the happy/stressed emotions of each neighbour in the propagation model. We assume that the mental states of intervened students are reduced with certainty considering that unforeseen external factors would not arise while being monitored during intervention [95].

2.2 Influence Maximization

Influence maximization on social networks has been modelled as Independent Cascade (IC) or Linear Threshold (LT) models [41]. There are uncertainties in social networks such as uncertainty in the edge influence probability and uncertainty in the initial values of the nodes. There are two lines of works that address such uncertainty: (1) IC is extended to choose the seed set to optimally spread influence in a graph [10]; (2) the influencer selects several seed sets sequentially to intervene the participants and the influencer receives observations about the participants' immediate social circles. To sequentially select the intervention participants, PSINET [106] and HEAL [105] formulate the problem as POMDP to handle the uncertainties of connection existence that are observed in each intervention [105, 106]. As another approach, DOSIM formulates the problem as a zero-sum game between the algorithm which picks the optimal policy, and the adversary (nature) which selects the connection probabilities with uncertainty [102]. However, all adopt IC model and only consider the spread from the influencer but not the effect of influencee's neighbours. Moreover, they only consider the node values as binary and set initial values as 0, i.e., they do not consider uncertainty on the initial values of the nodes. However, HEAL cannot scale up to realistic networks due to the huge state and action sets in our model. Hence, we propose abstraction and multi-level partitioning to improve scalability.

2.3 Adversarial Machine Learning

2.3.1 Generative Adversarial Networks (GAN)

Given some random distribution $p_{\mathbf{z}}(\mathbf{z})$ on the input noise $\mathbf{z}$ and a real data distribution $p_{data}(\mathbf{x})$ on training data $\mathbf{x}$, GAN is set up as a two-player zero-sum game between G and D as follows:

$$\min_G \max_D \; \mathbb{E}_{\mathbf{x}\sim p_{data}(\mathbf{x})}[\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}(\mathbf{z})}[\log(1 - D(G(\mathbf{z}))]. \tag{2.1}$$

During training, the parameters of G and D are updated alternately until we reach the global optimal solution $D(G(\mathbf{z})) = 0.5$. Next, we let Π_g and Π_d be the set of parameters for G and D, considering the probability distributions σ_g and σ_d. Then the mixed strategy formulation [33] is defined as:

$$\min_{\sigma_g} \max_{\sigma_d} \; \mathbb{E}_{\pi_d\sim\sigma_d}\mathbb{E}_{\mathbf{x}\sim p_{data}(\mathbf{x})}[\log D(\mathbf{x}, \pi_d)] + \mathbb{E}_{\pi_d\sim\sigma_d}\mathbb{E}_{\pi_g\sim\sigma_g}\mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}(\mathbf{z})}[\log(1 - D(G(\mathbf{z}, \pi_g), \pi_d)].$$

Various GAN architectures have been proposed to improve the performance of GANs. Deep Convolutional GAN (DCGAN) [77] replaces fully-connected layers in the generator and the discriminator with the deconvolution layer of Convolutional Neural Networks (CNN). Weight normalization techniques such as Spectral Normalization GAN (SNGAN) [66] stabilize the training of the discriminator and reduce the intensive hyperparameters tuning. There

are also multi-model architectures such as Stacked Generative Adversarial Networks (SGAN) [35] that consist of a top-down stack of generators and a bottom-up discriminator network. Each generator is trained to generate lower-level representations conditioned on higher-level representations that can fool the corresponding representation discriminator. Training GANs is very hard and unstable as pure NE for GANs might not exist and cannot be reliably reached by the existing approaches [63]. Considering mixed NE, MIX+GAN [3] maintains a mixture of generators and discriminators with the same network architecture but have their own trainable parameters. However, training a mixture of networks without parameter sharing makes the algorithm computationally expensive. Mixture Generative Adversarial Nets (MGAN) [32] propose to capture diverse data modes by formulating GAN as a game between a classifier, a discriminator, and multiple generators with parameter sharing. However, MIX+GAN and MGAN cannot converge to mixed NE. Mirror-GAN [33] finds the mixed NE by sampling over the infinite-dimensional strategy space and proposes provably convergent proximal methods. The sampling approach may be inefficient to compute mixed NE as the mixed NE may only have a few strategies with positive probabilities in the infinite strategy space.

2.3.2 Adversarial Training (AT)

As neural architectures become widely researched, investigations on their intrinsic vulnerability and enhancement of robustness have also increased. Numerous approaches have been proposed but Adversarial Training (AT) based methods [61] still remain the strongest defense approach.

Given a fixed classifier with parameters θ, a dataset X (e.g., an image x with true label y), and classification loss l, a bounded non-targeted attack is done by

$$\max_{\delta} l(x+\delta, y, \theta), \delta \in S, \tag{2.2}$$

where $S = \{\delta \mid ||\delta||_p \leq \epsilon\}$ is the space for the perturbation, $||\cdot||_p$ is the p-norm distance metric of adversarial perturbation and ϵ is the manipulation budget. We leverage Projected Gradient Descent (PGD) which uses the sign of the gradients and projects the perturbations into the set S, to construct an adversarial example for each iteration t with step size ϵ.

$$x^t = \Pi_{x+S}(x^{t-1} + \epsilon \cdot \text{sgn}(\nabla_x l(x^{t-1}, y, \theta))). \tag{2.3}$$

K-PGD [61] is the iterative attack of the adversarial examples constructed in Eq. (2.3) with uniform random noise as initialization, i.e., $x^0 = x + \zeta$ where ζ is from noise distribution. The strength of PGD attacks to generate adversarial examples depends on the number of iterations. It has an inner loop that constructs adversarial examples, while its outer loop updates the

model using mini-batch stochastic gradient descent on the generated examples. However, it is generally slow. Thus, we adapt Free AT [89] as the AT algorithm of our work. It computes the ascent step by re-using the backward pass needed for the descent step. To allow for multiple adversarial updates to the same image without performing multiple backward passes, Free AT trains on the same mini-batch several times in a row. Among the defense architectures, RobNet [26], randomly samples architectures from a search space and adversarially trains each of them. However, this is very computationally expensive. Moreover, evolutionary algorithm architectures such as R-NAS [45] and RoCo-NAS [21] are restricted to only black-box attacks. Lastly, AdvRush [69] proposes an adversarial robustness-aware neural architecture search algorithm using the input loss landscape of the neural networks to represent their intrinsic robustness.

2.3.3 Neural Architecture Search (NAS) for AML

With Automatic Machine Learning (AutoML) [57], Neural Architecture Search (NAS) has become one of the most important directions for machine learning. The goal of NAS is to automatically search for an effective architecture that satisfies certain demands. The DARTS in one-shot literature is the first approach that relaxes the search space to be continuous and conducts searching in a differentiable way. The architecture parameters and network weights can be trained simultaneously. However, they are extremely

time-consuming to obtain the generators. In the context of generative networks, Adversarial-Neural Architecture Search (Adversarial-NAS) proposes to search the generator and discriminator architectures simultaneously in a differentiable manner [20] . However, in both GAN and AT, NAS only samples the architectures from search space by picking the operations with maximum weights. To improve the modes that are missed for training, we propose DONAS for GANs and AT by sampling and finetuning multiple networks instead of just selecting the maximum [20]. DONAS does not induce any restriction to NAS.

2.4 Online POMDP Solvers

We focus on online algorithms for solving POMDPs that are more scalable and suitable for our problem with huge state and action sets than offline algorithms. Monte Carlo Sampling-based online solvers, POMCP [90] and DESPOT [108], use Monte Carlo tree search and maintain a search tree for all sampled histories. Thus, they have better solution quality but reduce scalability. HEAL does not maintain a search tree and uses QMDP heuristics [56] to find the best action.

2.5 Deep Recurrent Q-learning Architectures

DRL is widely used to solve large sequential decision-making problems. In particular, for uncertain environments, Deep Recurrent Q-learning (DRQN) has shown significant performance over the existing approaches. Hence, we focus on DRQN methods. The problem is formulated as a POMDP to represent the uncertain environment, and the solution is obtained using DRL. For instance, Egorov (2015) solved the POMDP problem with a deep neural network and represented the Q-function of POMDP with belief and action instead of state and action. Later works such as [27] proposed a recurrent model with convolutional layers and an LSTM layer to solve the POMDP environments of ALE. This model represents the Q-function with observation as a parameter instead of using the state. More recently, DDRQN [18], and ADRQN [111] improved the work by adding a history of actions to the Q-function's parameters. However, these works have either considered belief or observation (history of action) individually but not both together. They also do not consider the large action space of dynamic environments.

2.6 Game Theory

2.6.1 Normal-Form Game and Nash Equilibrium

Game theory helps people understand by representing competitive or cooperative situations in which participants make decisions rationally [35, 36]. The

most famous representation in game theory is the normal form game.

Definition 2.1: A normal form game is defined by:

1. the set of players $\mathcal{N} = \{1, 2, \cdots, N\}$,
2. the set of pure strategies (actions) $\mathcal{S} = \{S_1, S_2, \cdots S_N\}$
3. payoff functions for the players: $u_i : S_1 \times S_2 \times \cdots S_N \to \mathbb{R}$

Table 2.1: Two-player two-action normal form game [12]

	c_1	c_2
r_1	2,1	4,0
r_2	1,0	3,1

In a normal form game, all players aim to choose from their strategies in such a way as to maximize their utilities. To represent a two-player normal form game, consider $N = 2$ i.e., column player c and the row player r; $S_1 = \{r_i | 1 \leq i \leq m\}$ and $S_2 = \{c_j | 1 \leq j \leq n\}$ then the representation of the game considered is shown in Table 2.1. To analyze the possible outcomes of the game, we need to determine other additional information such as the sequence of the player's moves and the reasonability of the players. In the simultaneous-move case, each player assumes that other players are perfectly rational and choose the optimal action to maximize their own payoffs.

For this game, c will determine that r will always play r_1 because r_1's payoff is always higher than r_2 no matter what c plays. Therefore, the column player c will play c_1 to maximize her own payoff. Thus, the final outcome $\{r_1, c_1\}$ is an equilibrium of the game where no player wants to deviate unilaterally.

However, the equilibrium with pure strategies does not exist in all games. An example without any pure strategy equilibrium is shown in Table 2.2 [12].

Table 2.2: Normal form game with no pure strategy equilibrium

	c_1	c_2
r_1	1,-1	-1,1
r_2	-1,1	1,-1

In this game, we allow the player to randomize his action, which is called *mixed strategy* to get the equilibrium. The *mixed strategy* $\sigma_i \in \Delta_i$ is a probability distribution over all strategies/actions in the strategy set S_i where Δ_i is the set of all mixed strategies and $\sigma_i(s_j)$ is the probability that player i will play $s_j \in S_i$. We denote the strategy profile of all players as $\langle \sigma_i, \sigma_{-i} \rangle$ where σ_{-i} is the strategy profile of the players except player i. The common solution concept in simultaneous-move and the non-cooperative game is Nash equilibrium where no player has an incentive to deviate by playing a different (mixed) strategy [72]. This work by Nash also proved that every normal-form game with finite actions for N players has at least a mixed strategy Nash equilibrium. Hence, the Nash Equilibrium is defined by:

Definition 2.2: A strategy profile $\langle \sigma_i, \sigma_{-i} \rangle$ forms a Nash equilibrium if and only if:

$$U_i(\sigma_i^*, \sigma_{-i}^*) \geq U_i(\sigma_i, \sigma_{-i}^*); \forall \sigma_i \in \Delta_i \tag{2.4}$$

where $U_i(\sigma_i, \sigma_{-i})$ is the utility, i.e., expected payoff, of the strategy profile $\langle \sigma_i, \sigma_{-i} \rangle$.

2.6.2 Double Oracle Algorithm

Double Oracle (DO) algorithm starts with a small restricted game between two players and solves it to get the players' strategies at Nash Equilibrium (NE) of the restricted game. The algorithm then exploits the respective best response oracles for additional strategies of the players. The DO algorithm terminates when the best response utilities are not higher than the equilibrium utility of the current restricted game, hence, finding the NE of the game without enumerating the entire strategy space. Moreover, in two-player zero-sum games, DO converges to a min-max equilibrium [62]. DO framework is used to solve large-scale normal-form and extensive-form games such as security games [37, 94], poker games [100] and search games [8]. DO framework is also used in MARL settings [48, 71]. Policy-Space Response Oracles (PSRO) generalize the double oracle algorithm in a multi-agent reinforcement learning setting [48]. PSRO treats the players' policies as the best responses from the agents' oracles, builds the meta-matrix game, and computes the mixed NE but it uses Projected Replicator Dynamics (PRD) that updates the changes in the probability of each player's policy at each iteration. Since PRD needs to simulate the update for several iterations, the use of PRD takes a longer time to compute the meta-strategies and does not guarantee to compute an exact NE of the meta-matrix game. However, in DO-GAN, we can use a linear program to compute the players' meta-strategies in polynomial time since GAN is a two-player zero-sum game [86]. We present the corresponding

terminologies between GAN and game theory in the following table.

Table 2.3: Comparison of Terminologies between Game Theory and GAN

Game Theory terminology	GAN terminology
Player	Generator/ discriminator
Strategy	The parameter setting of generator/ discriminator, e.g., π_g and π_d
Policy	The sequence of parameters (strategies) till epoch t, e.g., $(\pi_g^1, \pi_g^2, ..., \pi_g^t)$ Note: Not used in DO-GAN.
Game	The min-max game between generator and discriminator
Meta-game/ meta-matrix	The min-max game between generator & discriminator with their respective set of strategies at epoch t of DO framework
Meta-strategy	The mixed NE strategy of generator/discriminator at epoch t

Chapter 3

We Mind Your Well-being: Preventing Depression in Uncertain Social Networks by Sequential Intervention

Nowadays, depression and other mood disorders have become major public health concerns worldwide. The World Health Organization (WHO) estimates that 350 million people are affected by depression throughout the world, reducing their ability to work and socialize, as well as increasing the rate of mortality from suicides [15]. Since counselling and social support help mitigate this problem by reducing people's stress [78], counselling services are emerging to provide interventions where a counsellor conducts dialogue sessions with several participants, finds out about their mental states and provides therapy. In this work, we use University Counselling Center (UCC)

[1] as an example. UCC is set up to monitor the stress level and well-being of students and decrease their stress level by interventions.

However, there are several challenges faced. Firstly, according to 2016 AUCCCD Survey, only $6-7\%$ of the students seek for counselling. Although we need to invite all the students for counselling to provide more effective social support, there are some limitations in the capacity of intervention such as the availability of counsellors in a university. Secondly, UCC cannot obtain the complete information about the students' mental states and the relationship between them at the beginning of the intervention. Thirdly, the large number of students makes it difficult to scale up to the real-world networks. Hence, it is not easy for UCC to efficiently decide the intervention plan to invite the students to counselling.

There are existing works to deal with such kind of problems using sequential planning algorithms. PSINET [106], HEAL [105] and CAIMS [107] maximize the HIV information spread in uncertain networks by formulating as POMDPs addressing the uncertainty of the influence between the participants. DOSIM [102] formulates the problem as a zero-sum game between the influencer and the adversary (uncertainty). However, these algorithms cannot be directly extended to solve our problem due to two main issues. Firstly, they do not consider the initial uncertainty of mental state values which leads to poor solution quality. Secondly, the extensions of these algorithms fail to

[1] https://www.american.edu/ocl/counseling/

scale up to realistic networks due to the huge state space of our problem.

This chapter highlights four key contributions. Firstly, we propose a novel model for intervention to prevent depression in an uncertain network. The model considers the different values of nodes and the propagation which is not only affected by the influencer but also by influencee's neighbours. We formulate the problem as POMDP to address the uncertainties of mental state values and influence which we observe along the interventions. Secondly, we propose a reasoning algorithm on the students' mental states upon observation to refine the belief of the POMDP. Refining the belief with reasoning reduces the loss of solution quality by the initial uncertainty of students' mental states. Thirdly, we propose MLPRAP (Multi-Level Partition algorithm with Reasoning and Abstracted Planning) with the following novelties: (i) we abstract the POMDP states to reduce the state space; (ii) we provide theoretical bounds on uniform networks for solution quality loss due to abstraction; (iii) we partition the graph into smaller partitions by two folds: (1) balanced partitioning (MLP-B); (2) cluster partitioning (MLP-C) so that the algorithm scales up to plan interventions for a network of at least 1000 students. Finally, we provide extensive experimental analysis on scalability and solution quality of MLPRAP compared to the state-of-the-art algorithms.

3.1 Motivation Scenario

Though our model can be applied to many scenarios such as mental health care for a public sector and counselling services for employees, we motivate our model by a specific case where UCC conducts a series of interventions to students. At each round of intervention, UCC invites a group of students for the counselling session. Through the intervention, UCC knows the mental states of the intervened students and the students within one degree of them [83], i.e., the influence between the counselled students and their friends. Since the counsellors have the student registry and the collected information in the previous interventions, UCC has initial mental states and influence estimation of the students.

3.2 Model

We consider Q rounds of UCC's interventions of a group of students and at each intervention, UCC selects K students. For each intervention, UCC obtains the observations about mental states of the selected students and influence between the selected students and their neighbours. Belief for the next intervention is updated according to the newly obtained observations. The objective of UCC is to decrease the global stress level of all students to prevent depression.

3.2.1 Network and Dynamics

The connection network of N students is represented by a directed graph $G = \langle V, E \rangle$ with the node set $V(|V| = N)$ and the edge set E. Every $i \in V$ represents a student in the connection network. Moreover, every $e = \{(i,j)|i,j \in V\} \in E$ represents that student i is a friend of student j and is associated with real value w_{ij}, which terms the *influence* that i induces to j. Since the friendship between a pair of students is mutual [88], if $(i,j) \in E$ then $(j,i) \in E$. However, different w_{ij} and w_{ji} values indicate the different influence i and j have on each other and we set $w_{ii} = 0$. For the sake of description, let $\mathcal{N}^{in}(i)$ be student i's in-neighbours where $(j,i) \in E$ with $0 < w_{ji} \leq 1$ for $j \in \mathcal{N}^{in}(i)$ and $\mathcal{N}^{out}(i)$ as out-neighbours where $(i,j) \in E, 0 < w_{ij} \leq 1$ for $j \in \mathcal{N}^{out}(i)$, respectively. The mental state of a student is one of the values in the discrete set $\mathcal{M} = \{0, 1, 2, \cdots, \mu\}$ [2] in which 0 represents the least stressed mental state and μ represents the most stressed mental state. Therefore, the students' mental states are represented by $\mathbf{v} = \langle v_1, ..., v_N \rangle$ where $v_i \in \mathcal{M}, i \in V$ is the mental state of student i.

We assume that in each intervention, UCC reduces the selected student's stress level by δ [3]. Due to her mental state change, her emotion propagates to her friends $j \in \mathcal{N}^{out}(i)$ by the propagation where the extent of influence varies by *influence* w_{ij}. The process is illustrated in Figure 3.1.

[2] In current literature, mental states can only be roughly evaluated inexplicitly as *mild*, *moderate* and *severe* [101].

[3] $\delta = f(v_i)$. If $v_i < \delta$, we assign $v_i = 0$ after decrease. This also applies to Δj in Eq (3.2).

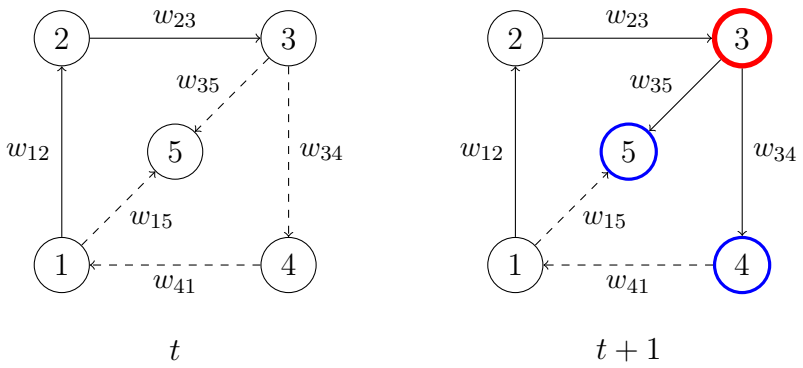


Figure 3.1: An illustrative example with 5 students. At round t, UCC knows the influence between the students w_{12} and w_{23}, represented by the solid lines. Influence unknown by UCC are represented by the dashed lines. Student 3 is picked by UCC to be intervened at t. w_{34} and w_{35} are known by UCC and students 4 and 5 are also influenced according to Eq. (3.1).

The extent of influence on j by i is represented by $\Delta_{i\to j}$ which is defined as:

$$\Delta_{i\to j} = \lfloor \frac{w_{ij}(\mu - v_i)}{w_{ij}(\mu - v_i) + \sum_{k\in\mathcal{N}^{in}(j)\setminus\{i\}} v_k \cdot w_{kj}} \cdot \delta \rfloor \tag{3.1}$$

which implies that when v_i is smaller, i.e., the influencer i is less stressed, $\Delta_{i\to j}$ is larger. When $\sum_{k\in\mathcal{N}^{in}(j)\setminus\{i\}} v_k \cdot w_{kj}$ is larger, i.e., her other neighbours have more stressed mental states, $\Delta_{i\to j}$ is smaller. This is inspired by the studies that the happiness/stress of the neighbours also affect the extent of influence [78, 83].

Hence, the total mental state value decrease on j is:

$$\Delta j = a_j \cdot \delta + \sum_{a_i=1, i\in V\setminus\{j\}} \Delta_{i\to j} \tag{3.2}$$

where $\mathbf{a} = \langle a_i \rangle, \forall i \in V$ such that $a_i = 1$ if student i is selected and $a_i = 0$ otherwise. The first term $a_j \cdot \delta$ is the influence induced by UCC and the second term is the influence induced by the propagation from the intervened

neighbours of j such that $\Delta_{i \to j}$ are aggregated for all neighbours i of j who are intervened.

3.2.2 Uncertainties

UCC does not have complete information of the students' mental states and influence initially. Hence, we model the uncertainty of students' mental states at the t^{th} intervention by defining an $N \times (\mu + 1)$ matrix $\hat{P}_{t-1}$ and each row $\hat{\mathbf{p}}_i^{t-1} = \langle \hat{p}_i^{t-1}(m) \rangle$ is the probability distribution over the discrete set $\mathcal{M}$ of student i. $\hat{p}_i^{t-1}(m)$ expresses the probability of student i being evaluated as mental state value m at t. For the uncertain influence, we also define an $N \times N$ matrix $\hat{W}_0$ which represents the estimates of influence between each pair of students. The values in $\hat{W}_0$ are estimated by the counsellors based on the information collection before the intervention process.

Initially, UCC has mental state estimates $\hat{P}_0$ and influence estimates $\hat{W}_0$. In each intervention, the mental states of the selected students and influence are observed. Hence, in t^{th} intervention, UCC derives $\hat{P}_t$ from the belief which is updated during the intervention. The rule for belief update is described in POMDP formulation section. $\hat{W}_t$ is also updated by assigning $\hat{w}_{ij} = w_{ij}, \forall j \in \mathcal{N}^{out}(i)$ and $\hat{w}_{ji} = w_{ji}, \forall j \in \mathcal{N}^{in}(i)$ for each intervened student i.

3.2.3 POMDP Formulation

POMDPs are sequential decision making models under uncertainty [76]. Formally, a POMDP is defined as $\mathcal{P} = \langle S, A, O, T, \Omega, R, b^0 \rangle$.

States and Initial Belief. S is the state set. A state is defined as $\mathbf{s} = \langle \mathbf{v}, \hat{W} \rangle$ where $\mathbf{v}$ denotes the students' mental states and $\hat{W}$ is defined as $\hat{w}_{ij} = w_{ij}$ if the influence of student i on j is known by UCC and $\hat{w}_{ij} = \hat{w}^0_{ij}$ otherwise where $\hat{w}^0_{ij}$ is the initial estimation of w_{ij} by UCC. UCC has an initial belief over states b^0 which is a distribution over S and $b^0_{\mathbf{s}}$ is the probability that the POMDP is at $\mathbf{s}$ at the beginning of the interventions.

Actions and Observations. UCC's selection of K students at each intervention is defined as action $\mathbf{a}$ where $a_i = 1$ means student i is selected and $a_i = 0$ otherwise, given the constraint $\sum_{i \in V} a_i = K$. All actions belong to the set A. UCC's observation by taking the action $\mathbf{a} \in A$ at state $\mathbf{s}$ is defined as $o(\mathbf{s}, \mathbf{a}) = \{v_i, w_{ij}, w_{ji} | \forall a_i = 1, j \in V, \mathbf{v} \in \mathbf{s}\}$, i.e., the mental states and the associated influence of the intervened students. All observations belong to the set O. Ω is the observation function of the POMDP which is uniquely defined by the action $\mathbf{a}$ and the state $\mathbf{s}$:

$$\Omega(o, \mathbf{s}, \mathbf{a}) = \begin{cases} 1, & \text{if } o = o(\mathbf{s}, \mathbf{a}); \\ 0, & \text{otherwise.} \end{cases} \tag{3.3}$$

Transition Probabilities Heuristic. $T(\mathbf{s}, \mathbf{a}, \mathbf{s}')$ is the transition probability of reaching $\mathbf{s}'$ from $\mathbf{s}$ by taking action $\mathbf{a}$. UCC takes action $\mathbf{a}$ and the

change in students' mental states is calculated as by Eq (3.2). Therefore, we can define $T(\mathbf{s}, \mathbf{a}, \mathbf{s}')$ as:

$$T(\mathbf{s}, \mathbf{a}, \mathbf{s}') = \begin{cases} 1, & \text{if } \mathbf{s}' = \langle \mathbf{v}', \hat{W}' \rangle; \\ 0, & \text{otherwise.} \end{cases} \tag{3.4}$$

where $\mathbf{v}'$ and $\hat{W}'$ are students' mental states and influence of the new state $\mathbf{s}'$ that are updated as:

$$v'_j = v_j - \Delta j; \tag{3.5}$$

$$\hat{w}'_{ij} = \begin{cases} w_{ij}, & \text{if } a_i = 1 \text{ or } a_j = 1; \\ \hat{w}_{ij}, & \text{otherwise.} \end{cases} \tag{3.6}$$

where $v'_j \in \mathbf{v}', \hat{w}'_{ij} \in \hat{W}', \forall i, j \in V$.

Reward and Policy. The reward $R(\mathbf{s}, \mathbf{a})$ of taking action $\mathbf{a} \in A$ in state $\mathbf{s} = \langle \mathbf{v}, \hat{W} \rangle$ is defined by:

$$R(\mathbf{s}, \mathbf{a}) = \sum\nolimits_{\mathbf{s}' \in S} T(\mathbf{s}, \mathbf{a}, \mathbf{s}') \left(\sum\nolimits_{i \in V} (v_i - v'_i) \right) \tag{3.7}$$

We define the history at intervention t as a sequence of past actions and observations $H_t = \langle a_1, o_1, a_2, \ldots, a_t, o_t \rangle$. We denote $\mathcal{H}_t$ as the set of all possible histories at t. The policy is defined as $\pi : \mathcal{H}_t \to A$ which takes history H_t as input and outputs the action $\mathbf{a}$. The expected reward for π starting from b^0 is defined as $V^\pi(b^0) = \sum_{t=1}^{Q} \mathbb{E}[R(\mathbf{s}, \mathbf{a}) | b^0, \pi]$. $\mathbb{E}[\cdot]$ outputs the expected value

of the input. The optimal policy π^* is the policy that maximizes $V^\pi(b^0)$. Formally, $\pi^* = \arg\max_\pi V^\pi(b^0)$.

Belief Update. Since in each state $\mathbf{s}$, we have the deterministic value of $\hat{W}$ where each element is either $\hat{w}_{ij}$ or w_{ij}, the initial belief b^0 can be defined by $\hat{P}_0$ and $\hat{W}_0$ such that for $\mathbf{s} = \langle \mathbf{v}, \hat{W} \rangle$:

$$b^0_{\mathbf{s}} = \begin{cases} \prod_{v_i \in \mathbf{v}} \hat{p}^0_i(v_i), & \text{if } \hat{W} = \hat{W}_0 \\ 0, & \text{otherwise.} \end{cases} \tag{3.8}$$

At intervention t, each state $\mathbf{s} = (\mathbf{v}, \hat{W})$ with belief $b^{t-1}_{\mathbf{s}}$ transits to $s' = (\mathbf{v}', \hat{W}')$ upon taking action $\mathbf{a}$. UCC observes $o \in O$ with the probability of $\Omega(o, \mathbf{s}', \mathbf{a})$. Hence we update the belief by

$$b^t_{\mathbf{s}'} = \gamma \cdot \Omega(o, \mathbf{s}', \mathbf{a}) \cdot \sum\nolimits_{\mathbf{s} \in S} T(\mathbf{s}, \mathbf{a}, \mathbf{s}') \cdot b^{t-1}_{\mathbf{s}} \tag{3.9}$$

where γ is the normalizing constant: $\gamma = 1/(\sum_{\mathbf{s}' \in S} \Omega(o, \mathbf{s}', \mathbf{a}) \cdot \sum_{\mathbf{s} \in S} T(\mathbf{s}, \mathbf{a}, \mathbf{s}') \cdot b^{t-1}_{\mathbf{s}})$. After that, we update $\hat{P}_t$ based on the belief update with $\hat{p}^t_j(m) = \sum_{\mathbf{s}' \in S, v'_j = m} b^t_{\mathbf{s}'}$.

3.3 MLPRAP

To solve the formulated problem, we first tried online POMDP solvers such as DESPOT [108] and POMCP [90] which can scale up to large networks. These solvers, however, limit the size of initial belief set. This makes them

not suitable for our setting since the initial belief set may be as large as the state set.

We propose MLPRAP (Multi-Level Partition algorithm with Reasoning and Abstracted Planning), extended from the algorithms for the dynamic influence maximization in social networks to improve scalability and solution quality. MLPRAP sequentially selects the students in an uncertain large network with three novelties: reasoning on the estimated mental states of students to refine the belief before each intervention, abstraction of the POMDP states to solve large POMDPs and multi-level partitioning of the graph.

Algorithm 3.1 describes the flow of MLPRAP. We partition the graph G with multi-level partitioning by Algorithm 3.4 or 3.5 to obtain a map $\mathbb{P}$ of partition *par* and k (line 1). Then, we generate abstracted POMDPs for each *par* (line 2). Next, we refine belief b^{t-1} with reasoning by Algorithm 3.2 (line 4). Each POMDP is solved to find the optimal one-node *action* with QMDP Heuristics (lines 6-8). We choose K actions from $\mathbb{A}$ according to multi-level partitioning variants to get $\mathbf{a}$ which is added to policy π and belief b^t is updated (lines 9-10).

In the rest of this section, we will discuss all the components of MLPRAP: (i) reasoning on the estimated mental states; (ii) abstraction of states; and (iii) multi-level partitioning of the graph.

Algorithm 3.1: `MLPRAP`(G)

Obtain $\mathbb{P}$ with multi-level partitioning (Algorithm 3.4/ 3.5) ; // $\mathbb{P}$ is a map of *par* and k
Generate abstracted POMDPs $\mathcal{P}'_k$ and assign k for $\langle par, k\rangle \in \mathbb{P}$;
for $t = 1 : Q$ **do**
    Reason to refine b^{t-1} of each $\mathcal{P}'_k$ (Algorithm 3.2);
    Initialize $\mathbb{A} = \emptyset$; // $\mathbb{A}$ is a map of *action* and k
    for $\mathcal{P}'_k$ **do**
        $action =$ `FindBestAction`$(\mathcal{P}'_k)$;
        add $\langle action, k\rangle$ to $\mathbb{A}$;
    Choose K actions from $\mathbb{A}$ and assign to $\mathbf{a}$;
    Add $\mathbf{a}$ to π, and update b^t of $\mathcal{P}'_k$ according to $(b^{t-1}, \mathbf{a})$;
return π

3.3.1 Reasoning

In our POMDP, the initial students' mental states and *influence* ($\hat{P}_0$ and $\hat{W}_0$) are inaccurate since they are estimated by UCC without evaluation. This leads to poor solution quality as the algorithm sequentially selects the participants from inaccurate initial beliefs. At every intervention t, as we get information about selected students' mental states and real influence between the students, we do reasoning on the estimated students' mental states and refine b^t so that the belief estimates are closer to the true mental states.

The change in her emotions propagates to her neighbours in the social network as emotions can spread from one person to another. Moreover, her mental state is also affected by the mental states of her neighbours. There are two main conclusions that describe the relationship about the mental states in a social network: (i) the close friends in the network have similar mental state values [31]; (ii) the mental state of a student is not just affected by one of

her friends independently but affected by the mental states of all her friends [15]. These existing findings can be represented by the State Relationship Rule (SRR).

State Relationship Rule (SRR). *A student's mental state lies within the range defined by the weighted average of all her neighbours' mental state ranges. Given each neighbour i's mental state range as* $[lb_i, ub_i]$ *where* $lb_i \leq v_i \leq ubi$*, j's range* $[lb_j, ub_j]$ *is defined as:*

$$lb_j = \lfloor \frac{\sum_{i \in \mathcal{N}^{in}(j)} lb_i \cdot \hat{w}_{ij}}{\sum_{i \in \mathcal{N}^{in}(j)} \hat{w}_{ij}} \rfloor; \tag{3.10}$$

$$ub_j = \lceil \frac{\sum_{i \in \mathcal{N}^{in}(j)} ub_i \cdot \hat{w}_{ij}}{\sum_{i \in \mathcal{N}^{in}(j)} \hat{w}_{ij}} \rceil \tag{3.11}$$

Accordingly, the prediction of the mental state depends not only on the mental state of the neighbours but also on the influence of the neighbours. We also allow uncertainty of the range predicted by SRR. Therefore, there will be a probability to predict a student to be stressed although she is surrounded by all neighbours in happy mental states.

So far, we assume that all mental state values can occur in our POMDP. There might be some states in the belief with the mental state values that are inconsistent with SRR. During the reasoning process, we take SRR into account and set the probabilities of the states in the belief that violate SRR to 0. This process can reduce the uncertainty on mental state values and thus, improve the solution quality. A concrete example is shown in Figure 3.2 to

describe how the belief is refined during the reasoning process according to the State Relationship Rule (SRR).

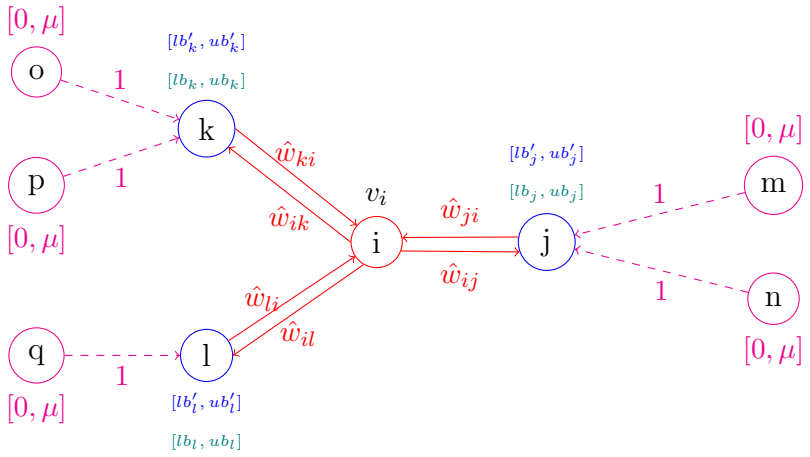


Figure 3.2: An illustrative example of reasoning process. When UCC intervenes student i, UCC observes the value of her mental state and influence between her neighbours shown in red. Using this observation, we predict the ranges of j, k, l in blue with observed v_i and the ranges from their other in-neighbours in magenta. The predicted ranges are in green. Using these ranges, we calculate $[lb_i, ub_i]$ for i, the j, k, l's ranges are modified if $v_i \notin [lb_i, ub_i]$. The final ranges for j, k, l are shown in blue.

Example 1. *We consider an example in Figure 3.2 with* $\mu = 10$ *where* $\{v_i = 5, w_{ij} = 1, w_{ji} = 1, w_{ik} = 0.5, w_{ki} = 1, w_{il} = 0.25, w_{li} = 0.25\}$ *is observed. First, with Eq. (3.11), we can predict the mental state ranges for* j, k, l *as* $[1, 9], [0, 10], [0, 10]$ *respectively. Using these ranges, we find that we get* $ub_i = 4$ *and* $lb_i = 6$ *after calculating* i*'s range with* $v_k = 0$ *and* $v_k = 10$*. Hence,* 0 *and* 10 *are removed from* k*'s range. Thus, we get* $[1, 9], [1, 9], [0, 10]$ *as the refined ranges for* j, k, l*. During reasoning process, we set* $b_{\mathbf{s}}$ *as zero if* $\mathbf{s}$ *contains inconsistent mental states that are not in the refined ranges and violates SRR.*

According to SRR, we do the reasoning as follows. When UCC intervenes

student i, UCC gets an observation $\{v_i, w_{ij}, w_{ji} | j \in V\}$. With this observation, we predict the range for student j's mental state where $j \in \mathcal{N}^{out}(i)$. We set the mental state range for other unobserved students as $[0, \mu]$ and influence as the maximum, i.e., 1 to make the predicted range wider to compensate for uncertain values. For the robust prediction of mental states, UCC defines that the predicted range for j's mental state has the width of at least ω i.e., $ub_j - lb_j \geq \omega$. Hence, if the width of the range predicted by Eq. (3.11) is less than ω, we modify the range as $[\theta_j - \omega/2, \theta_j + \omega/2]$ [4] where $\theta_j = \lfloor (lb_j + ub_j)/2 \rfloor$. On the other hand, we need to make sure that the observed value v_i is in the range $[lb_i, ub_i]$ which is calculated by the predicted ranges for $v_j, \forall j \in \mathcal{N}^{in}(i)$. For each value α in range $[lb_j, ub_j]$, we use $[lb_k, ub_k]$ for $k \in \mathcal{N}^{in}(i) \setminus \{j\}$ to find $[lb_i, ub_i]$ and α is removed from the range if $ub_i < v_i$ or $lb_i > v_i$. We run the refining process until all the observed v_i values are in the range $[lb_i, ub_i]$ which is calculated by all the predicted ranges for $j \in \mathcal{N}^{in}(i)$.

We refine the belief before each intervention so that the algorithm selects action based on the belief closer to the real network. We do reasoning on the network, predict the range and check for each belief state where $b_{\mathbf{s}}^{t-1} > 0$ if all mental state values of $\mathbf{s}$ are in respective predicted ranges.

Algorithm 3.2 checks each state where $b_{\mathbf{s}}^{t-1} > 0$ if the students' mental states are in the range predicted by SRR (line 1). We define a vector $\mathbf{d}$ where $d_i = 1$ if student i is observed and $d_i = 0$ otherwise. The

[4] If $\theta_j - \omega/2 < 0$, $lb_j = 0$. If $\theta_j + \omega/2 > \mu$, $ub_j = \mu$.

Algorithm 3.2: `Reasoning`$(b_{t-1}, \mathbf{d})$

1 **for** $\mathbf{s} \in S, b_{\mathbf{s}}^{t-1} > 0$ **do**
2 $\mathbf{lb}, \mathbf{ub}$ = `PredictMentalStateValues`$(\mathbf{s}, \mathbf{d})$;
3 **for** $v_j \in \mathbf{s}$ **do**
4 **if** $v_j \notin [lb_j, ub_j]$ **then** $b_{\mathbf{s}}^{t-1} = 0$;
5 $b_{\mathbf{s}}'^{t-1} = b_{\mathbf{s}}^{t-1} / \sum_{\mathbf{s}' \in S} b_{\mathbf{s}'}^{t-1}, \forall \mathbf{s} \in S$;
6 **return** b'^{t-1};

range for each student's mental state value is predicted according to SRR in `PredictMentalStateValues`(.) which is described in Algorithm 3.3. After that, the algorithm checks if the mental state of all students are in the respective predicted ranges and sets $b_{\mathbf{s}}^{t-1}$ to 0 otherwise (line 4). Finally, non-zero belief values are rescaled to sum up to 1 (line 5).

Predicting Mental State Values by SRR. Algorithm 3.3 returns the predicted ranges for the students. lb_i, ub_i are set as certain mental state of i and $\tilde{w}_{ij}$ is set as certain influence of i and j if i is observed, i.e., $d_i = 1$, and set to $lb_i = 0, ub_i = \mu, \tilde{w}_{ij} = 1$ otherwise (lines 1-7). Given the observed student i, the mental state range for her out-neighbour j is predicted by applying SRR (lines 8-15). Lines 16-22 describe that the mental state range $[lb_i, ub_i]$ of observed student i is calculated using the predicted ranges for her in-neighbours. If $\alpha \in [lb_j, ub_j], \forall j \in \mathcal{N}^{in}(i)$ predicts the range $[lb_i, ub_i]$ and $v_i \notin [lb_i, ub_i]$, it is removed from $[lb_j, ub_j]$ (line 22).

Algorithm 3.3: `PredictMentalStateValues`$(\mathbf{s}, \mathbf{d})$

for $i \in V$ **do**
  if $d_i == 1$ **then**
    $lb_i = ub_i = v_i$;
    for $j \in V$ **do** $\tilde{w}_{ij} = w_{ij}$;
  else
    $lb_i = 0$; $ub_i = \mu$;
    for $j \in V$ **do** $\tilde{w}_{ij} = 1$;
for $i \in V, d_i == 1$ **do**
  for $j \in \mathcal{N}^{out}(i), d_j == 0$ **do**
    $lb_j = \lfloor \frac{\sum_{k \in \mathcal{N}^{in}(j)} lb_k \cdot \tilde{w}_{kj}}{\sum_{k \in \mathcal{N}^{in}(j)} \tilde{w}_{kj}} \rfloor$;
    $ub_j = \lceil \frac{\sum_{k \in \mathcal{N}^{in}(j)} ub_k \cdot \tilde{w}_{kj}}{\sum_{k \in \mathcal{N}^{in}(j)} \tilde{w}_{kj}} \rceil$;
    $\theta_j = (lb_j + ub_j)/2$;
    $lb'_j = \theta_j - \omega/2, ub'_j = \theta_j + \omega/2$;
    if $lb'_j < lb_j$ **then** $lb_j = lb'_j$;
    if $ub'_j > ub_j$ **then** $ub_j = ub'_j$;
for $i \in V, d_i == 1$ **do**
  for $j \in \mathcal{N}^{in}(i), d_j == 0$ **do**
    for $\alpha \in [lb_j, ub_j]$ **do**
      $lb_i = \lfloor \frac{\alpha \cdot \tilde{w}_{ji} + \sum_{k \in \mathcal{N}^{in}(i) \setminus \{j\}} lb_k \cdot \tilde{w}_{ki}}{\sum_{k \in \mathcal{N}^{in}(i)} \tilde{w}_{ki}} \rfloor$;
      $ub_i = \lceil \frac{\alpha \cdot \tilde{w}_{ji} + \sum_{k \in \mathcal{N}^{in}(i) \setminus \{j\}} ub_k \cdot \tilde{w}_{ki}}{\sum_{k \in \mathcal{N}^{in}(i)} \tilde{w}_{ki}} \rceil$;
      if $lb_i > v_i$ *or* $ub_i < v_i$ **then**
        remove α from $[lb_j, ub_j]$;
return lb, ub;

3.3.2 Abstraction of POMDP States

We reduce the huge state space by defining σ such that a student's mental state is one of the values in the abstracted discrete set $\mathcal{M}' = \{0, \sigma, 2\sigma, ..., \mu\}$ and we define the state set S' with $\mathcal{M}'$. Hence, the non-abstracted total number of states, i.e., $(\mu + 1)^N$ is reduced to $\lceil (\mu + 1)/\sigma \rceil^N$.

Every state in S' belongs to S, i.e., $S' \subset S$. Given action $\mathbf{a} \in A$ and state

$\mathbf{s} \in S'$, the successive state $\hat{\mathbf{s}} \in S$ is obtained according to the transition function of original POMDP $\mathcal{P}$. The change in mental states is computed with observed v_i for $a_i = 1$ and abstracted v'_j for $a_j = 0$. If $\hat{\mathbf{s}} \notin S'$, we replace with $\mathbf{s}' \in S'$ such that $0 < v'_i - \hat{v}_i \leq \sigma, \forall i \in V$. The policy evaluated with S' is denoted as π'. We denote the optimal policy of the abstracted POMDP as π'^* and the expected reward as $V^{\pi'^*}$.

Although the abstraction method improves scalability, there is some loss of solution quality due to the approximation, i.e., $V^{\pi^*} - V^{\pi'^*}$. Hence, in Lemma 1, we prove a theoretical bounded error, $V^{\pi^*} - V^{\pi'^*}$, for independent network at the end of round Q.

Lemma 1. *For a certain network with independent students,* $V^{\pi^*} - V^{\pi'^*} \leq Q \cdot (\sigma - 1) \cdot K$.

Proof. The maximum difference between v_i and v'_i is $(\sigma - 1)$ since we have $\mathbf{s}'$ if $0 < v'_i - \hat{v}_i \leq \sigma$. Hence, for Q rounds where K nodes are chosen at each round, the maximum difference between the total expected rewards of the optimal policies π^* and π'^* is $Q \cdot (\sigma - 1) \cdot K$.

In Lemma 2, we prove the bound for certain networks with connections between the students. For general networks, we cannot prove the bounds since the influence propagation greatly depends on the neighbours' mental states and influence. Hence, we consider a complete graph with $w_{ij} = w, \forall i, j \in V$.

Lemma 2. *For a certain network where the students' mental state values and*

influence are known,

$$V^{\pi^*} - V^{\pi'} \leq Q \cdot K \cdot \big(\frac{\delta \cdot (N \cdot \mu - 2)}{\mu + N - 2} - \delta + \sigma - 1\big) \tag{3.12}$$

Proof. Let π^* be the optimal policy of the original POMDP $\mathcal{P}$ with initial state $\mathbf{s}^0$. At round t, given the state $\mathbf{s}^{t-1}$ and action $\mathbf{a}^t$, $\mathbf{s}^{t-1}$ transits to $\mathbf{s}^t$. We refer to the mental states at t as $v_i^t, \forall i \in V$ where $v_i^t \in \mathbf{v}^t, \mathbf{v}^t \in \mathbf{s}^t$. In a complete graph, the influence level on j at round t is defined using Eq. (3.2) as:

$$\Delta_j^t = a_j^t \cdot \delta + \sum_{a_i^t = 1, \forall i \in V \setminus \{j\}} \frac{(\mu - v_i^{t-1}) \cdot \delta}{(\mu - v_i^{t-1}) + \sum_{k \in N \setminus \{i\}} v_k^{t-1}} \tag{3.13}$$

Hence, we calculate R_t as:

$$R_t = K \cdot \delta + (N-1) \cdot \sum\nolimits_{a_i^t = 1, \forall i \in V} \frac{(\mu - v_i^{t-1}) \cdot \delta}{(\mu - v_i^{t-1}) + \sum_{k \in N \setminus \{i\}} v_k^{t-1}} \tag{3.14}$$

$v_i^{t-1} = 1, \forall i \in V$ gives the largest propagation to find the maximum total estimated reward possible for $\mathcal{P}$. Hence,

$$R_t \leq \frac{K \cdot \delta \cdot (N \cdot \mu - 2)}{\mu + N - 2} \tag{3.15}$$

To find the lower bound, we ignore the propagation process of the abstracted POMDP $\mathcal{P}'$. Hence,

$$R_t' \geq K \cdot (\delta - \sigma + 1) \tag{3.16}$$

Hence, we have the bound for $V^{\pi^*} - V^{\pi'}$ as:

$$V^{\pi^*} - V^{\pi'} \leq \hat{R} = Q \cdot K \cdot \big(\frac{\delta \cdot (N \cdot \mu - 2)}{\mu + N - 2} - \delta + \sigma - 1\big) \tag{3.17}$$

Since $V^{\pi'^*} \geq V^{\pi'}$, we proved that $V^{\pi^*} - V^{\pi'^*} \leq \hat{R}$.

3.3.3 Multi-Level Partitions

We improve the scalability further by multi-level partitioning of the graph. The most intuitive way is to partition the graph into K partitions, i.e., the number of selected students. But the POMDPs are still very large to be solved. Hence, we divide each of K partitions into smaller subpartitions.

The first variant is MLP-B which has ηK *balanced* partitions, i.e., each partition contains a similar number of nodes. Algorithm 3.4 starts with initializing $\mathbb{P}$, a map of partitions and their indices, as an empty set (line 1). We use the METIS algorithm [40] to partition the graph G while minimizing cross-edge influence between partitions so as not to lose the network structure of the graph (line 2). It returns a list of partitions, i.e., $pars$. Each partition is indexed (line 4) and ensembled as POMDP to compute the optimal one-node action. Finally, actions with the best K rewards are chosen to get an action of K nodes.

As balanced partitioning may not maintain the network structure, in the second variant, MLP-C, we cluster the graph into K partitions by finding minimum cross-edge cuts without keeping the partitions balanced. Let l be

Algorithm 3.4: $\texttt{MLP} - \texttt{B}(\eta)$

```
1 Initialize ℙ = ∅ and k = 0 ;                    // ℙ is a map of par and k
2 pars = METIS(G, ηK);
3 for par ∈ pars do
4 |  add ⟨par, k⟩ to ℙ; k++;
5 return ℙ;
```

the maximum limit of a partition that can be solved. If a partition is larger than l, we divide the partition into smaller subpartitions.

In Algorithm 3.5, we first cluster the graph G into K partitions by finding minimum cut and store the partitions in $pars$ (line 1). We initialize the map $\mathbb{P}$ as an empty set (line 2). The partitions are indexed and added to $\mathbb{P}$ (line 8). If a partition has more than l nodes, we divide it into $\lceil \texttt{nodecount}/l \rceil$ subpartitions using METIS and store in $subpars$. We use METIS over clustering to avoid having many levels of partitioning to get subpartitions. The subpartitions are kept with same index and added to $\mathbb{P}$ to indicate that only one node from the partitions with same index will be chosen for an action of K nodes (lines 4-7). After obtaining the optimal action from each POMDP, the action with maximum reward from the partition with same index is chosen.

3.4 Experimental Evaluation

In this section, we analyze the performance of MLPRAP-B and MLPRAP-C, i.e., MLPRAP with MLP-B and MLP-C with different settings. All our experiments are run on a 3.2 GHz 4-core Intel machine having 16 GB of RAM.

Algorithm 3.5: $\texttt{MLP} - \texttt{C}(l)$

```
pars = minimumCutClustering(G, K);
Initialize ℙ = ∅ and k = 0;
for par ∈ pars do
    if nodecount(par) > l then
        subpars = METIS(par, ⌈nodecount(par)/l⌉) ;
        for subpar ∈ subpars do
            add ⟨subpar, k⟩ to ℙ
    else add ⟨par, k⟩ to ℙ ;
    k++;
return ℙ;
```

The results are averaged over 30 trials. We use runtime and total reward as metrics to evaluate scalability and solution quality on both synthetic and real networks. The 95% confidence intervals are drawn in all figures which show that all the results are statistically significant. During experiments, we first compute UCC's policy and simulate on the networks without any uncertainties to compute the real reward obtained by the policy during the intervention process.

3.4.1 Experiment Setup

Problem Instance Generation. We run the lab experiments to evaluate the performance of the algorithm. We synthesize the problem instances since real-world experiments on the study of intervention process in a social network is challenging and there is no publicly available data which studies the stress level of the people in a network. However, evaluations of algorithms on synthetic data are widely accepted [102, 105, 106] as they serve as an

important first step towards future applications of the model. We generate networks with realistic mental state values while using reasonable measures according to [15, 31] so that the network reflects the mental state values of each individual in the network. There are two kinds of networks considered:

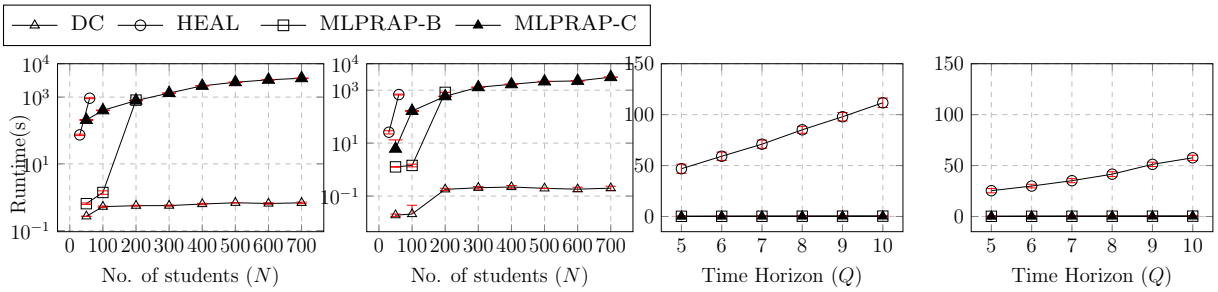
- **synthetic networks**: We generate the synthetic network G by two methods. First, Barabási-Albert scale-free networks, where each new vertex is connected to ξ vertices using a preferential attachment mechanism [4]. Since the students stay in groups of 3 or 4 which is the size of a team for a group project, we let $\xi = 3$. Second, Erdos-Ŕenyi random networks $ER(N, |E|)$, where exactly $|E|$ edges are randomly constructed between each pair of nodes [14]. We set $|E| = 3N$ to let each node have 3 connections on average. Then, we assign $\hat{w}_{ij} \in W_0$ as randomized values from $[0, 1]$.

- **real networks**: The first network is Zachary Karate Club dataset (Karate) with 34 nodes and 78 edges [110] which is the friendship data of the members of a university karate club. This will closely reflect the relationship between students in the network and the effectiveness of interventions. We assign $\hat{w}_{ij} \in W_0$ as randomized values from $[0, 1]$. The second dataset is Mobile-1 dataset (Mobile) which has 107 nodes and 513 edges [93]. It consists of the logs of calls and cell tower IDsx of users for ten months. We assign communication count between users i and j as $\hat{w}_{ij}$.

After we have obtained the network, we set $\mu = 9$, pick a student $i \in V$ and assign the uniformly sampled value from $[0, \mu]$. For all other $j \in V \setminus \{i\}$, we iteratively predict the mental state ranges according to SRR and assign the sampled value from the predicted range. We repeat the process until convergence where all the nodes are assigned with the mental state values.

Baselines. We use two algorithms as baselines: (1) Degree Centrality (DC) which selects the highest degree node first; (2) K-variant HEAL as it is the most relevant algorithm which has been demonstrated to perform much better than earlier algorithms.

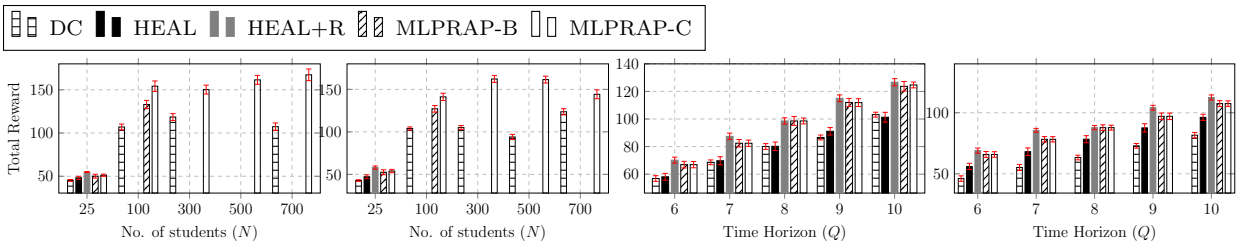
3.4.2 Experiment Results



(a) BA (b) ER (c) BA (d) ER

Figure 3.3: Scalability comparison of MLPRAP variants with DC, HEAL (Figure 3.3a and 3.3b are plotted in log-scale)

Scalability Analysis. We compare the runtimes of our algorithms and baselines by varying network sizes. We set $Q \in \{5, ..., 10\}$, $\delta = 2$, $K = 5$, $\sigma = 3$, $\eta = 2$ and $l = 10$. In Figure 3.3a, we compare the runtime of each algorithm



(a) BA (b) ER (c) BA (d) ER

Figure 3.4: Solution quality comparison of MLPRAP variants with DC, HEAL

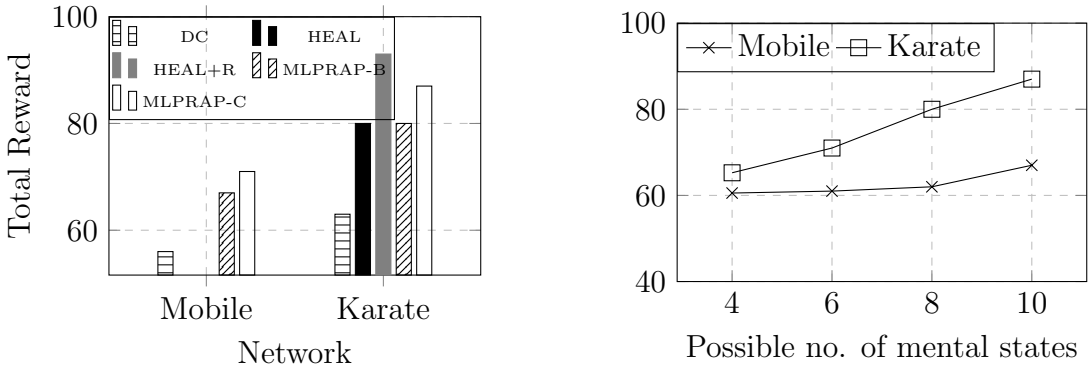
along the y-axis w.r.t varying network sizes along the x-axis. For example, for a problem with a network of 30 students, it takes 0.07 s for DC, 375.383 s for HEAL, 0.149 s for MLPRAP-B and 0.748 s for MLPRAP-C. While DC is the fastest, it does not result in good solution quality as we will discuss in solution quality analysis. HEAL can only solve up to 30 students. It runs out of memory for larger networks. Moreover, HEAL runs very slowly even for small networks. We also run HEAL+R (HEAL with Reasoning) and it has similar runtime with HEAL. While the running time is faster in smaller networks, MLPRAP-B runs out of memory on the network of more than 100 students with $\eta = 2$. MLPRAP-C can solve larger networks than all other algorithms. In this experiment, we show up to network with 700 students which the system can solve within the time limit of 3600s. The system can solve 1000 students network in 6141.92s. We find the same trends for ER networks shown in Figure 3.3b.

The second experiment runs the algorithms on BA and ER networks of 25 students ($N = 25$) with fixed $\eta = 2$ and $l = 10$. In Figures 3.3c and 3.3d, we vary time horizons along the x-axis on the same network to record the

runtime along the y-axis. For a BA network of 25 students, HEAL takes 50s to plan for 5 interventions whereas DC and both variants of MLPRAP take just a fraction of a second, i.e., 0.07s, 0.195s and 0.202s respectively. While all algorithms have linearly increasing runtimes with increasing time horizon, MLPRAP algorithms do not significantly increase and have better scalability.

Although we only scale up to a network with 700 nodes for the scalability analysis since we limit the solving time to 1hr, MLPRAP-C algorithm can solve much larger networks since the multi-level partitioning algorithm keeps partitioning the network until the partitions can be solved by the MLPRAP algorithm. With a more powerful machine and longer time, MLPRAP-C can solve much larger networks.

Solution Quality Analysis. Figures 3.4a and 3.4b show the rewards of the different algorithms along y-axis on the varying sizes of the student networks along the x-axis. We keep N=25, 100, 300, 500, 700 to highlight the limitations of baseline algorithms. E.g., for BA network of 25 nodes, the total reward is 45.2, 48.17, 54.33, 50.1, 51.17 for DC, HEAL, HEAL+R, MLPRAP-B and MLPRAP-C respectively. The trends show that HEAL+R improves solution quality better than simply running HEAL and the approximated solution with abstraction does not suffer a significant loss. As networks become larger, MLPRAP variants have a larger advantage over DC. Therefore, MLPRAP-B and MLPRAP-C are more suitable to solve larger networks.



(a) Real world networks

(b) Effect of estimates

Figure 3.5: Experiments on real networks

In Figure 3.4c and 3.4d, we compare the total reward (y-axis) with increasing time horizon (x-axis) for both types of random networks with 25 nodes. The trends show that MLPRAP variants result in better solution quality than DC and HEAL. While HEAL+R is the best for a network of 25 students, we have shown that it cannot scale up to larger networks. On the other hand, MLPRAP-B and MLPRAP-C do not lose much solution quality compared to HEAL+R.

Effect of uncertainty. We also analyze the effect of uncertainty of the student networks on the solution quality. The uncertainty for initial mental state values is reflected by the initial estimated mental state range. In Figure 3.5b, the value 4 on the x-axis represents that there are 4 possible mental state values with 0.25 probability for each student while the y-axis represents the total reward. We can conclude from the results that the homogeneous distribution, i.e., $|\mathcal{M}| = 10$, gives the best total reward. This is because homogeneous

distribution assigns all the mental states with equal probability lowering the chances of incorrectly estimating the mental states and losing the solution quality in contrast to smaller ranges.

Real Networks. We also evaluate the scalability and solution quality of the algorithms on real networks (Mobile and Karate). Figure 3.5a describes the rewards of each network for 5 interventions with $K = 6$. HEAL and HEAL+R cannot solve Mobile and MLPRAP-C gives the best solution quality. Similar to synthetic networks, HEAL+R gives the best total reward for Karate. While MLPRAP-B only gives a similar solution quality as HEAL, MLPRAP-C gives much higher solution quality than DC and HEAL and comparable to HEAL+R.

3.5 Chapter Summary

In this chapter, we propose a novel model that considers emotion propagation from not only the influencer but also neighbours of the influencee while selecting the students for interventions in uncertain networks. We propose MLPRAP algorithm with reasoning, abstraction of states and multi-level partitioning of the graph into smaller POMDPs to sequentially plan to select the students for each intervention. Finally, experiments with synthetic networks as well as real networks show that MLPRAP approach has significantly better scalability and solution quality comparable to existing algorithms.

Chapter 4

Planning Sequential Interventions to Tackle Depression in Large Uncertain Social Networks Using Deep Reinforcement Learning

In this section, we focus on handling the student counselling problem discussed in Chapter 3 using Deep reinforcement learning (DRL), which adapts the students' changing mental states and connections in a dynamic environment. DRL has shown great success in solving problems with uncertainty and high dimensional state space such as continuous control [55, 103] and Atari Learning Environment (ALE) [5]. In the planning apporach, the student counselling problem is formulated as Partially Observable Markov Decision Process (POMDP) and solved by designing a POMDP solver with abstraction and graph partitioning techniques to break up the social network to

make the problem solvable. This approach breaks up the student network causing information loss. To overcome this, we propose DRL with Particle Swarm Optimization (DRLPSO) to plan the optimal sequential actions in the uncertain dynamic environment with large action space.

We made 5 key contributions in this chapter: 1) We propose Deep Q-learning integrated with Long Short-Term Memory (LSTM) to handle dynamic influence maximization problems with large action space; 2) To handle a very large action space of POMDP formulation in DRL, one straightforward way to solve this problem is by randomly sampling a smaller action set, but this may not lead to the optimal solution as the particle cannot explore the entire action space. To overcome this problem Discrete Particle Swarm Optimization (DPSO) is applied by initializing n particles that search through the entire action space for the optimal action based on a belief state. The Q-network subsequently chooses the action with maximum reward as the predicted action for each round; 3) The main advantage of DPSO is to optimize the search of the maximum Q-value action by generating n particles which effectively capture multiple optima in the multi-modal action space; 4) We demonstrate the efficiency and effectiveness of DRLPSO in comparison with the five existing DRL algorithms using Monte Carlo simulation. We also compare the proposed DRLPSO solution with the previous POMDP solver solution. The results show that the DRLPSO outperforms the POMDP solver solution in terms of effectiveness.

4.1 Deep Reinforcement Learning with Particle Swarm Optimization

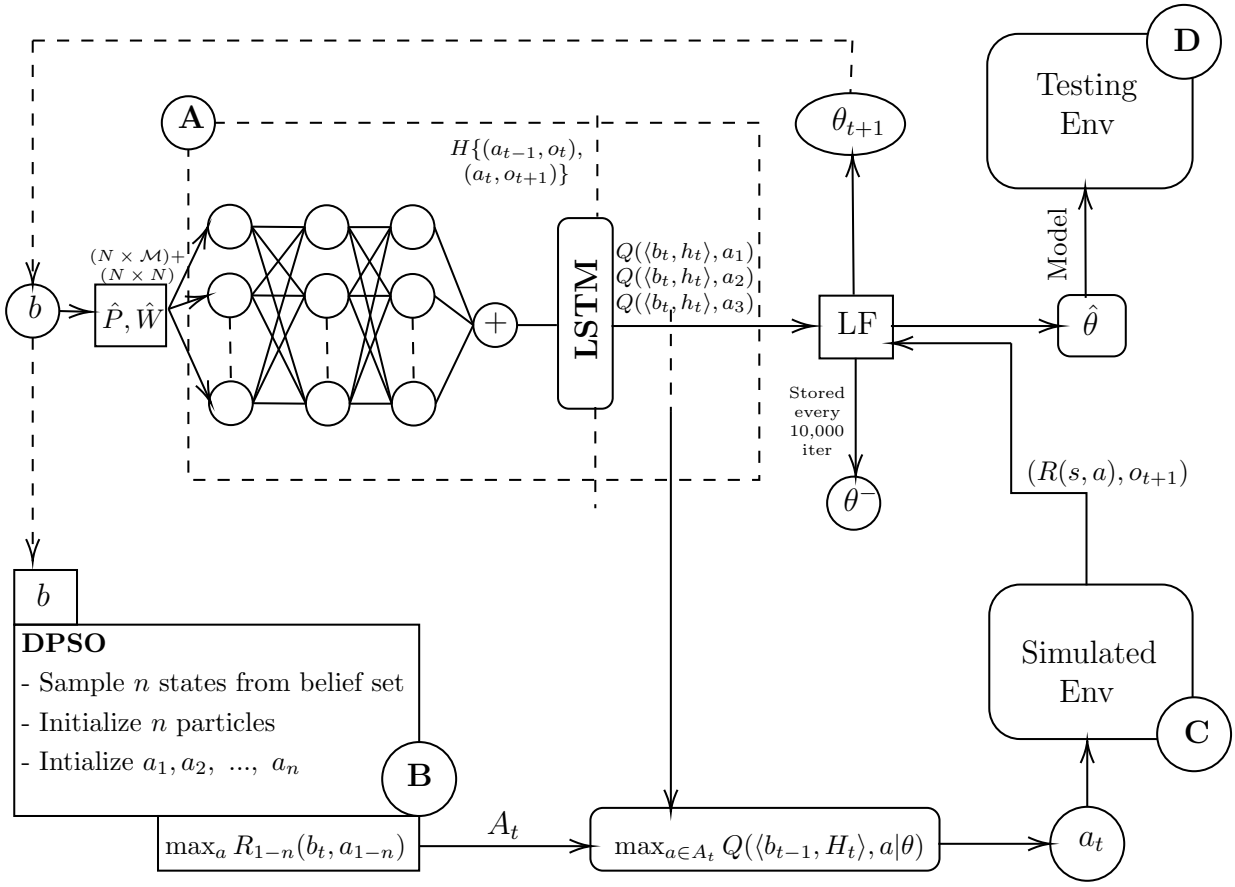


Figure 4.1: DRLPSO Architecture

As discussed in section 2 (Related Works), dynamic influence maximization is NP-Hard which is a challenging task to obtain the optimal solution. Thus, solving by the brute force method is infeasible. Moreover, being formulated as POMDP, the traditional RL method and DQN are infeasible due to Q-function being represented with state values instead of partial observation and no historical information is preserved. Thus, we use deep Q-learning integrated with the LSTM layer (DQ-LSTM). However, the state and action

space of this problem grows exponentially with the number of students in the network i.e., $(\mu + 1)^N$ states and the number of chosen students at each round i.e., $\binom{N}{K}$ actions. For this problem, the output layer of LSTM could be complicated as it cannot handle the large action space and results in a large number of outputs, which could reduce the performance of the deep recurrent Q-network. Hence, in this study, we embed DPSO to pick the optimal action. By doing so, the training phase of DQ-LSTM can converge to an optimal policy.

Instead of randomly selecting an action, DPSO initializes n particles to optimize the reward function and later converges to a set of optimal actions for a given belief. At each step, DQ-LSTM gets the action set optimized by DPSO and also makes the DRL converge to the optimal solution. Therefore, we propose DRLPSO (Deep Reinforcement Learning with Particle Swarm Optimization), where DPSO finds the optimal action for each step and Deep Q-learning integrated with LSTM ensures the optimal policy for the dynamic environment.

4.1.1 DRLPSO Architecture

The overview of DRLPSO is shown in Figure 4.1. We initialize the problem as a discrete environment that takes action as input and outputs observation and reward. The system consists of 4 major components (Blocks A, B, C and D of Figure 4.1).

- Deep Q-learning with LSTM as the top layer (Block A)
- Discrete Particle Swarm Optimization Model to predict the best action (Block B)
- Simulated Environment to train the network (Block C)
- Testing Environment (Block D)

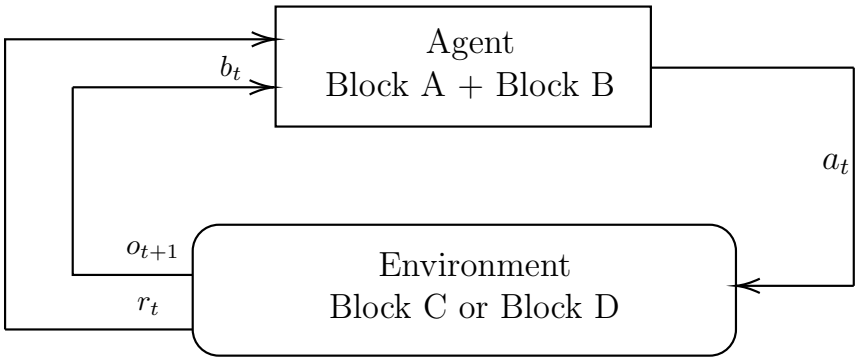


Figure 4.2: DRLPSO components represented as Markov Model commonly used in RL

Block A used for training the parameters to predict the Q-values. The deep Q-learning consists of ReLU as activation function, 3 hidden layers of neurons and learning is performed by backpropagation. LSTM is integrated to retain historical information. The training procedure consists of 1) Choose the action for best Q-value using DPSO (Block B); 2) Execute the action in the simulated environment (Block C), obtain observation and reward; 3) Optimize parameter θ by backpropagation until the loss converges. After the training, we test the model in the testing environment (Block D). The components are represented as a Markov Model in Figure 4.2.

4.1.2 DRLPSO Algorithm

The proposed DRLPSO is described in Algorithm 4.1. We first initialize the experience replay similar to the existing DRL methods, the number of iterations to train, the initial belief according to the environment and the parameters for Q-network and target network (Lines 1-2). For each iteration, we simulate the environment for training, initialize the action, history and assign the initial belief across the state set. Next, we perform $\mathbb{T}$ rounds of intervention. During each round of intervention, we predict the optimal action set with DPSO by randomly initializing n particles with different positions in the discrete action space until they converge to multiple optimal positions (actions) (line 7). DPSO finds the actions with the best reward based on the belief b_{t-1}. From the optimal action set of DPSO, we choose the action where the particle gives the maximum predicted Q-value (line 8). This action is executed in the simulated environment to obtain reward R and observation o_{t+1} and update b_t (lines 9-10). We then store the transition sequence in experience replay (line 12). After that, we randomly sample the transition sequences as a mini-batch and update parameter θ by Eqs. (15-18) (lines 13-16). Finally, we obtain the converged parameter $\hat{\theta}$ (line 18).

Discrete Particle Swarm Optimization. In training the deep recurrent Q-network, existing methods randomly sample an action, without considering the observation received from the environment. Instead, DRLPSO uses

Algorithm 4.1: DRLPSO

Result: Parameter θ

Initialize the experience replay D, # of iterations M;

Initialize Q-network and Target-Network with θ and θ^{-} respectively;

/ train the network shown in Block A in Figure 4.1 */*

for *iteration= 1 to M* **do**

  Simulate environment and b^0 ; *// Block C in Figure 4.1*

  Initialize the first action $a_0 = noaction$, $h_1 = \emptyset, o_1 = \emptyset$;

  for $t = 1...\mathbb{T}$ *rounds* **do**

    A_t = best action set from DPSO module where velocity and position updates using Eqs. (4.1) and (4.2) ; *// Block B in Figure 4.1*

    $a_t = \arg\max_{a \in A_t} Q(\langle b_{t-1}, H_t \rangle, a|\theta)$;

    Execute action a_t to obtain reward $R_t(.)$ and observation o_t;

    Update b_t according to o_t using Eq. (3.9);

    Store transition $\{\langle b_{t-1}, h_t \rangle, a_t, r_t, b_t\}$ in D ;

    Randomly sample a mini batch of transition sequences from D and index as j;

    $$y_j = \begin{cases} r_j, \text{ if } t = \mathbb{T} \\ \text{Eq.(4.4)}, \text{ otherwise} \end{cases}$$

    Compute gradient using the loss function (Eq. (4.6));

    Update θ according to Eq. (4.7);

  end

end

Save model as $\hat{\theta}$;

DPSO (Block B), which takes belief state updated with the observation. We initialize n particles to explore and exploit by parallelly converging to multiple optima. For DQ-LSTM, we adapt the action set from the convergence of the DPSO particles to the output layer of LSTM.

We modified the traditional continuous PSO [42] to the Discrete PSO. We initialize the position boundary B to be $nCr(N, K)$ i.e., the number of different, unordered combinations of K students from the network with N students. We randomize the swarm positions from 0 to $B - 1$ as the initial discrete actions. While optimizing the objective function defined in Eq. (3.7),

we update the velocity $\vec{V_i}$ and position X_i values of each particle by:

$$\vec{V_i}(t+1) \;=\; \vec{V_i}(t) + (c_1 r_1)(pbest_i - X_i(t)) + (c_2 r_2)(gbest_i - X_i(t)) \quad (4.1)$$

$$X_i(t+1) = \begin{cases} 0, & \text{if } \lceil X_i(t) + \vec{V_i}(t+1) \rceil < 0 \\ B, & \text{if } \lceil X_i(t) + \vec{V_i}(t+1) \rceil > B \\ \lceil X_i(t) & +\vec{V_i}(t+1) \rceil, \text{ otherwise} \end{cases} \quad (4.2)$$

where c_1, c_2 are self confidence values, r_1, r_2 are randomized values and $pbest_i$ is the maximum reward position (action) the particle has visited, and the best among all the subswarm particles is stored in $gbest_i$. Generally, $gbest_i$ will converge to a single solution if there exists only one optimum. In this problem, the search space consists of multiple optimal solutions. Hence, the particles will form a subswarm and converge to multiple peaks to map with the output layer of DQ-LSTM.

Deep Recurrent Q-learning. After we obtain the predicted action, we execute the action to the simulated environment (Block C) to obtain the reward $R(s, a)$ and observation o_{t+1}. We then update belief b_t and history h_{t+1} to calculate the predicted loss $\hat{y}$ and the actual loss y. DQN uses experience replay and two networks to train parameter θ: main network to optimize θ value and targeted network to retain θ^- which is updated every 10,000

iterations [68]. Similarly, we adapt experience replay, the two networks and use the stale updated θ^- given by the target network to get the actual Q-values. This technique has been empirically shown to be tractable and stable.

Generally, the state transition problems with MDP considers the Q-update using state and action given by:

$$Q(\mathbf{s}_t, \mathbf{a}_t) \;=\; Q(\mathbf{s}_t, \mathbf{a}_t) \;+\; \alpha(R(\mathbf{s}_t, \mathbf{a}_t) \;+\; \gamma \max_{\mathbf{a}} Q(\mathbf{s}_{t+1}, \mathbf{a}) \;-\; Q(\mathbf{s}_t, \mathbf{a}_t)) \quad (4.3)$$

where α is learning rate and γ is discount factor. This Q-function is well-suited for MDP where the states are fully observed. But for the problems involving POMDP, Q-function is parametrized by belief (b), action (a) and observation (o) since $s \neq o$ due to the partial observability and we cannot observe the current state $\mathbf{s}_t$ in POMDP. We have to use a deep network to better estimate Q-values from belief and observation. We denote weights and biases of the Q-networks as θ and the function can be denoted by $Q(\langle b_{t-1}, H_t \rangle, a_t | \theta)$. We first calculate the Q-values for each belief, history of action and observation and subsequently update the parameters of the deep network by minimizing the loss function at each iteration i of the training phase.

Actual Q-value is calculated by:

$$y = R(\mathbf{s}_t, \mathbf{a}_t) + \gamma \max_{\mathbf{a}} Q(\langle b_t, H_{t+1} \rangle, \mathbf{a} | \theta_i^-) \quad (4.4)$$

Predicted Q-value is calculated by:

$$\hat{y} = Q(\langle b_{t-1}, H_t \rangle, a_t | \theta_i) \tag{4.5}$$

The l_2 loss function (LF) is given by:

$$L(\langle b_t, H_t \rangle, a_t | \theta_i) = (y - \hat{y})^2 \tag{4.6}$$

Finally, we perform backpropagation to train θ which is updated using:

$$\theta_{i+1} = \theta_i + \alpha \nabla_\theta L(\theta_i) \tag{4.7}$$

When the training converges, we save the model $\hat{\theta}$ and execute the testing environment (Block D) for the $\mathbb{T}$ rounds to get the optimal policy.

4.2 Experiment Results

In this section, we evaluate the proposed DRLPSO and compare it with the five existing methods for large sequential decision-making problems. We run the simulated student networks with mental state propagation to evaluate the performance of DRLPSO. We synthesize the problem instances since there is no publicly available data that studies the stress level of the people in a network. However, evaluations of algorithms on simulated networks are widely applied [102, 105, 106] as they serve as an important reference towards the real-world applications of the architecture. Further, we plan to do the exper-

iments and carry out interventions for the university students to know about their stress levels, give them counselling and improve their performance.

4.2.1 Experiment Setup

We simulate a student network G by leveraging Erdos-Ŕenyi random network generation [14], where exactly $|E|$ edges are randomly constructed between each pair of nodes (students). Assuming that the students usually study or stay in groups of 3 or 4 which is also the size of a team for a regular group project, we set $|E| = 3N$ to let each node have at least 3 connections on average. Then, we assign $\hat{w}_{ij} \in W_0$ as randomized values between $[0, 1]$. After that, we set $\mu = 9$ and randomly assign the nodes with mental state values between $[0 - 9]$.

We compare the performance of proposed DRLPSO against the five baseline methods such as Degree Centrality (DC) [38]; HEAL [105]; DBQN [13]; DRQN [18] and ADRQN [111]. Since DRQN and ADRQN use the convolutional layers as input, we convert the observed states as 84x84 grey-scale images through Python Pillow Module and train both networks accordingly. To mitigate the problem of handling large action space, we randomly sample the action space of the problem with 30 actions to train using DBQN, DRQN and ADRQN. To make a par comparison, we assign the LSTM output layer of DRLPSO to have 30 outputs. We assign the randomized initial estimate mental state and influence values for all the methods.

Our experimental settings are as follows: we set $(\mathbb{T}, K, \delta, \mu)$ as $(5, 1, 2, 9)$ for networks of 5 nodes, $(5, 5, 2, 9)$ for networks of 30 nodes and $(10, 5, 2, 9)$ for even larger networks. We implement all the aforementioned programs in Python and run all the experiments on a system of 3.2GHz 4-core Intel CPU and NVIDIA DGX-1 with 32 GB of RAM.

4.2.2 Evaluation

Parameter setting of DPSO. We set the parameter values of c_1 and c_2 with SwarmOps [74]. Since we define the LSTM output as 30 Q-values, we aim to map the size of the converged optimal action set to 30. Figure 4.3 illustrates the number of unique peaks captured by varying the number of particles in the DPSO to search for the best action set for a dynamic environment having a network of 30 students. Here, the minimum and average peak captures are shown after 10 runs. From the results, we can infer that DPSO with at least 90 particles has to be initialized.

Training Rewards. Figure 4.4 shows the training curves of the deep Q-learning algorithms with 100,000 iterations. After every 500 iterations, we predict the actions for 5 rounds on 30 problem instances of the testing environment and record the total reward. From the figure, we observe that DRLPSO converges within the limited number of iterations and achieves much higher reward values in comparison with the other learning methods.

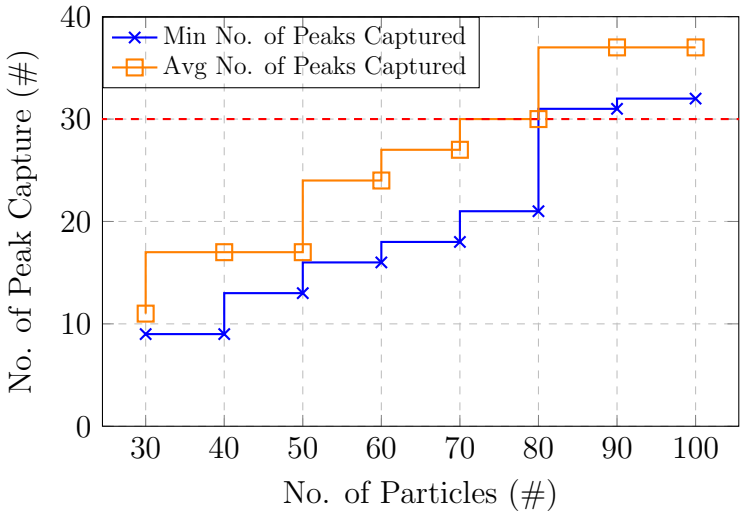


Figure 4.3: Peak convergence by varying the number of particles

Table 4.1: Reward Comparison for MLPRAP

Nodes	DC ($\mu \pm \sigma$)	HEAL ($\mu \pm \sigma$)	DBQN ($\mu \pm \sigma$)
5	9.9 ± 1.668	13.43 ± 1.96	10.56 ± 1.977
30	45.03 ± 2.809	47.2 ± 3.59	31.76 ± 15.36
100	102.6 ± 9	-	155.4 ± 25.53
200	119.53 ± 18.584	-	128.9 ± 16.572
500	118.53 ± 20.91	-	133.93 ± 17.3

Nodes	DRQN ($\mu \pm \sigma$)	ADRQN ($\mu \pm \sigma$)	DQ-LSTM	DRLPSO ($\mu + \sigma$)
5	11.56 ± 1.716	11.53 ± 1.81	13.48 ± 1.743	$\mathbf{13.8} \pm 1.69$
30	38.33 ± 13.514	48.8 ± 11.74	-	$\mathbf{62.3} \pm 3.9$
100	162 ± 21.94	168.3 ± 37.47	-	$\mathbf{225} \pm 12.62$
200	138.33 ± 17.516	171.5 ± 24.81	-	$\mathbf{232.1} \pm 16.04$
500	143.13 ± 19.72	175.17 ± 18.27	-	$\mathbf{233.06} \pm 12.31$

Performance Analysis. We train each algorithm to analyze the total reward after $\mathbb{T}$ rounds of intervention with 30 problem instances of the testing environment for each network size. Table 4.1 shows the reward comparison of different sizes of networks with the aforementioned experimental settings. We use $\sum_{t=1}^{\mathbb{T}} R_t(\mathbf{s}, a)$ as evaluation metric which is the total overall stress level reduced in the process of intervention. From the table, we observe that rewards of the previous dynamic influence maximization methods such as

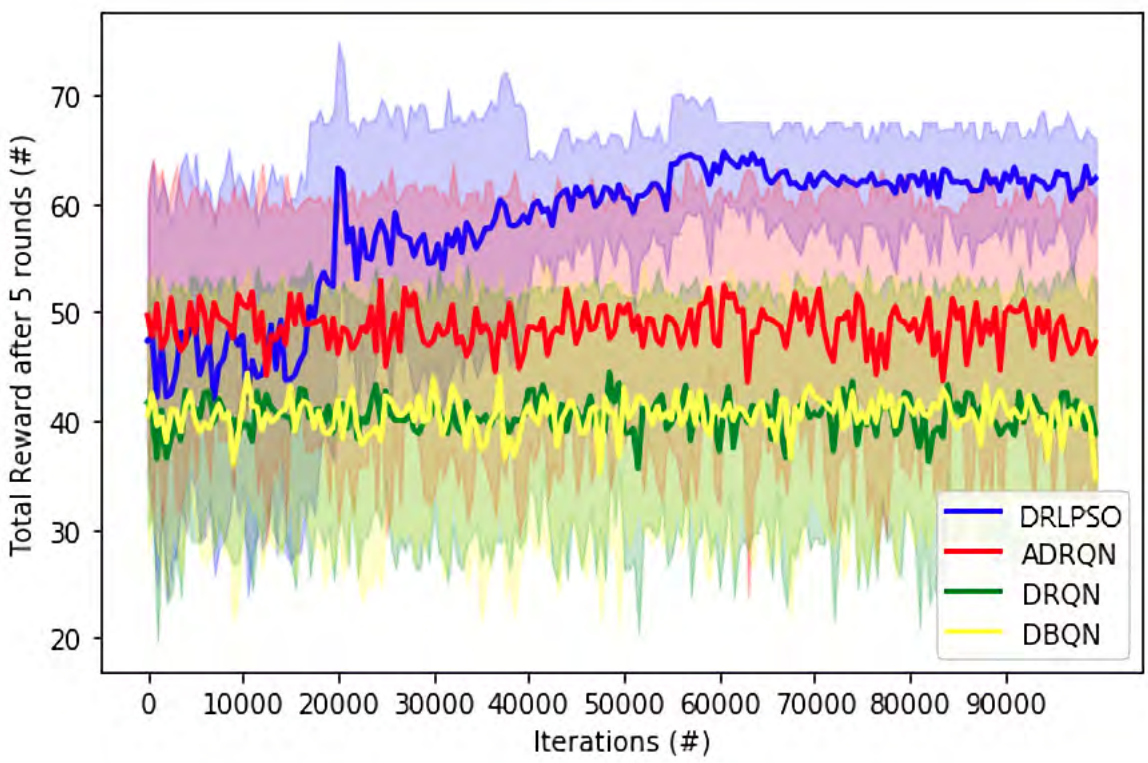


Figure 4.4: Performance of learning algorithms in 100k iterations

DC and HEAL are lower reward values than the proposed DRLPSO. This is because DC does not consider uncertainty and partial observation, while HEAL does not consider the uncertainty of initial mental state values. However, HEAL cannot scale up for networks larger than 30 nodes.

Similarly, when the comparison is made among 5 learning algorithms. Table 4.1 shows DQ-LSTM is only feasible for 5 students but it stops running for 30 students and above as it fails to handle larger state and action space i.e., state space of 10^{30} and action space of $\binom{30}{5} = 142506$. The proposed DRLPSO outperforms DBQN, DRQN and ADRQN in the total reward with a high mean reward value and lower standard deviation over $\mathbb{T}$ rounds with 30 runs. This is because, in DBQN, DRQN and ADRQN, the action sets

are randomly sampled to a set of 30 actions instead of a large action space unlike DRLPSO uses DPSO to optimize the action set before mapping to the Q-values from LSTM.

We also propose the Average Percentage of Reward Increase (APRI), to compare our proposed DRLPSO with baseline methods defined by:

$$APRI = \frac{\mu(R_{DRLPSO}) - \mu(R_{Algo})}{\mu(R_{DRLPSO)}} \times 100\% \tag{4.8}$$

where $\mu(.)$ is the mean reward value from the Monte Carlo simulation of the respective algorithms. We calculate APRI for the network size of 30 nodes where we observe that our proposed DRLPSO significantly outperforms DC, HEAL, DBQN, DRQN, ADRQN by 27.7%, 24.2%, 49%, 38.4% and 21.6% respectively. Thus, the overall stress level in the network will be best reduced after $\mathbb{T}$ rounds of intervention with the counsellors by selecting the students by the DRLPSO method at each round.

Comparison with MLPRAP method. Here, we will compare with our previous method, MLPRAP: Multi-Level Partition algorithm with Reasoning and Abstracted Planning, which uses POMDP solver and multi-level partitioning of the original graph in Chapter 3. We only compare with the technique proposed in this chapter, MLPRAP-C, which hierarchically partitions the original student network in clusters until each partition has under l students which is the maximum limit of the students that can be solved by the afore-

mentioned hardware configuration.

We have also compared using the realistic networks against the MLPRAP method and observe better results. The first network is the Zachary Karate Club dataset (Karate) with 34 nodes and 78 edges [110] which is the friendship data of the members of a university karate club. This will closely reflect the relationship between students in the network and the effectiveness of interventions. We assign $\hat{w}_{ij} \in W_0$ as randomized values from $[0, 1]$. The second dataset is the Mobile-1 dataset (Mobile) which has 107 nodes and 513 edges [93]. It consists of the logs of calls and cell tower IDsx of users for ten months. We assign communication count between users i and j as $\hat{w}_{ij}$.

Table 4.2: Reward Comparison for DRLPSO

Nodes	MLPRAP-B($\mu + \sigma$)	MLPRAP-C ($\mu + \sigma$)	DRLPSO ($\mu + \sigma$)
100	133 ± 4.74	154.17 ± 6.06	$\mathbf{225} \pm 12.62$
200	-	150.23 ± 5.14	$\mathbf{232.1} \pm 16.04$
500	-	161.5 ± 5.17	$\mathbf{233.06} \pm 12.31$
1000	-	160.8 ± 6.88	$\mathbf{210.27} \pm 13.20$
1500	-	161.3 ± 5.43	$\mathbf{217.60} \pm 12.42$
Karate	80 ± 0.7	84.6 ± 1.22	$\mathbf{89.2} \pm 1.51$
Mobile	62.1 ± 1.21	62.7 ± 1.88	$\mathbf{66.4} \pm 1.83$

The comparison is as shown in Table 4.2. Although it takes longer to train to be able to start using to select the students for intention, DRLPSO has better results than MLPRAP-C and MLPRAP-B since there is no loss of information due to the partitioning with the DRL methods.

4.3 Discussion

The overview of evaluation with the actual study is as shown in Figure 4.5. We will collaborate with University Counselling Center (UCC) to conduct a

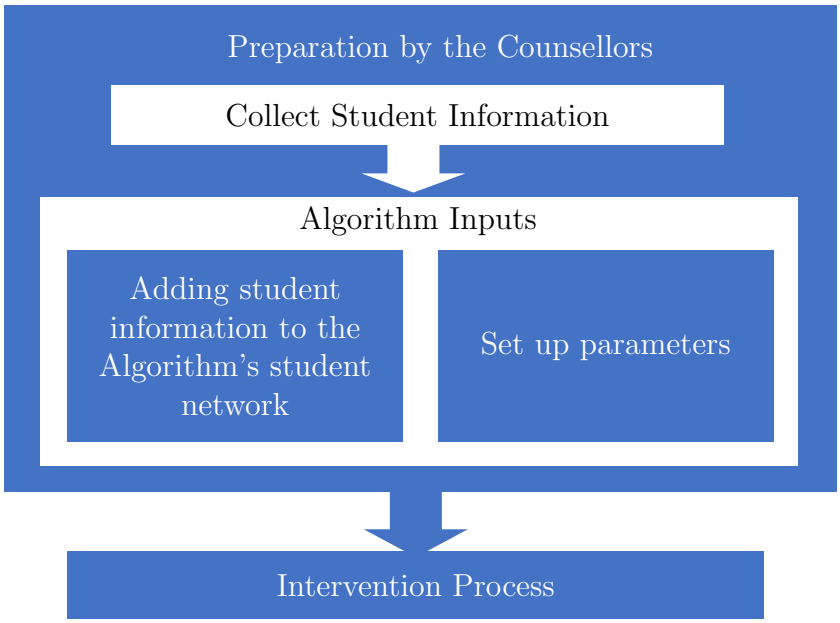


Figure 4.5: Overview of the implementation plan

real scenario with graduate students. Student data and mental state assessments are masked with a pseudonym and direct ID links will be destroyed after the counselling process. The participants are given a consent form and free to withdraw anytime. The detailed plan is as follows.

- We prepare the well-documented implementation plan of the intervention process and design the stress assessment questionnaire according to the questionnaires from the established literature such as (Perceived Stress Scale, 1994).

- We then submit the documents to the review process from Institutional

Review Board for Research Involving Human Subjects to ensure that the research activity follows applicable legislation of Singapore.

- After the approval from the board, we perform the necessary data collection to estimate the students' mental state and influence values. We first get the students' consents with a consent form where the consequence of the study/ interventions is explained in detail, understandable language usage. Afterwards, we collect the following information from the students: 1. College; 2. Supervisor, Co-supervisor; 3. Course Enrollment; 4. Physical Location (Lab/Office location, Home/ Dormitory Address); 5. Participation in student clubs; 6. Publication List; 7. Academic Performance: Grades; 8. Performance: Evaluation by Supervisor.

- Next, we set up the environment according to the collected student information (mental states and influence).

After the preparation step is done, we begin the intervention process. UCC initiates the intervention process in coordination with the algorithm. The intervention process is illustrated in Figure 4.6. At each round, UCC invites the candidate students who are selected by the algorithm. During the intervention round, the counsellors conduct a therapy session with the selected students, evaluate/assess their mental states based on a set of designed questionnaires and learn their social-circles. When the intervention is over the counsellors update the newly collected information to the algorithm.

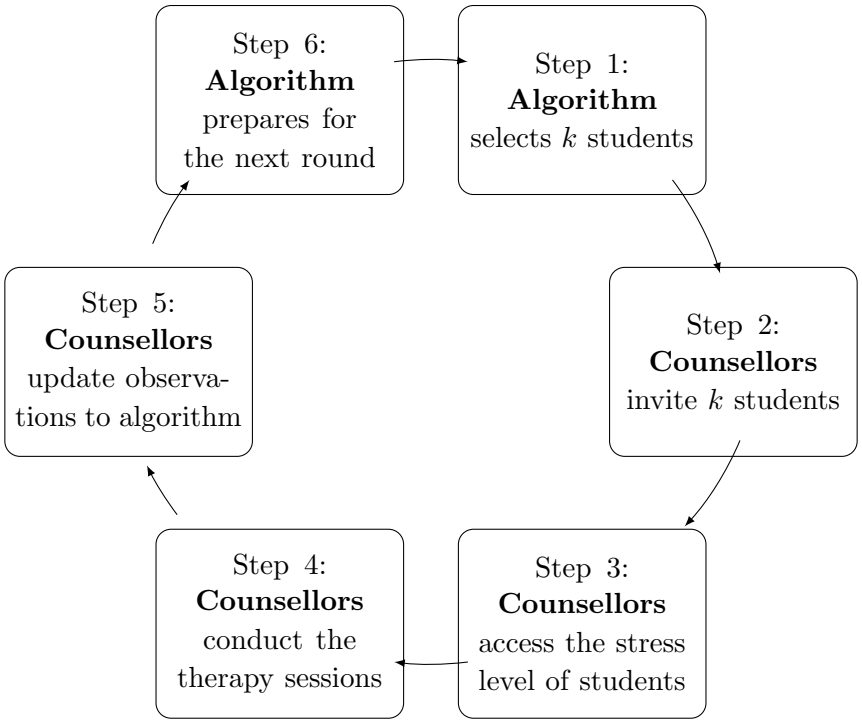


Figure 4.6: Intervention process: Step 1,6 are planned by the algorithm while Step 2-5 are performed by the counsellors

The algorithm then updates the network, estimates the stress levels of other students by the reasoning module and selects the students to be invited for the subsequent rounds.

4.4 Chapter Summary

In this chapter, we propose DRLPSO which uses the Deep Q-learning integrated with LSTM (DQ-LSTM) and DPSO to optimally select the predicted action, obtain Q-values and train the partially observable network. We use belief and history of action and observation as input to DQ-LSTM. Results have shown that in a dynamic environment with a large action space, the proposed DRLPSO achieves a higher total reward compared to the exist-

ing DRL approaches by an average of 32%. Moreover, comparing with the POMDP solver, DRLPSO takes time for training but outperforms in terms of effectiveness. In future, we plan to evaluate the algorithm with the real-world setting and further explore online reinforcement learning methods so that the training time for DRLPSO can be reduced.

Chapter 5

A Double-Oracle Framework for Generative Adversarial Networks (DO-GAN)

Generative Adversarial Networks (GANs) [24] have been applied in various domains such as image and video generation, text-to-image synthesis and equipment condition monitoring [58, 79, 80, 81]. Various architectures are proposed to generate more realistic samples [65, 75, 77] as well as regularization techniques [2, 67]. From the game-theoretic perspective, GANs can be viewed as a two-player game where the generator samples the data and the discriminator classifies the data as real or generated. They are alternately trained to maximize their respective utilities till convergence corresponding to a pure Nash Equilibrium (NE).

However, pure NE cannot be reliably reached by existing algorithms as pure NE may not exist [17, 63]. This also leads to unstable training in

Figure 5.1: Training images with fixed noise for SGAN and DO-SGAN/P (pruning) until termination. Both during training and after convergence, DO-SGAN/P can generate better quality images with FID score of 6.32 against SGAN's FID score of 6.98.

GANs depending on the data and the hyperparameters. Therefore, mixed NE is a more suitable solution concept [33]. Several recent works propose mixture architectures with multiple generators and discriminators that consider mixed NE such as MIX+GAN [3] and MGAN [32] but they cannot guarantee to converge to mixed NE. Mirror-GAN [33] computes the mixed NE by sampling over the infinite-dimensional strategy space and proposes provably convergent proximal methods. However, the sampling approach may not be efficient as mixed NE may only have a few strategies in the support set.

Double Oracle (DO) algorithm [62] is a powerful framework to compute mixed NE in large-scale games. The algorithm starts with a restricted game that is initialized with a small set of actions and solves it to get the NE strategies of the restricted game. The algorithm then computes players' best-

responses using oracles to the NE strategies and add them into the restricted game for the next iteration. DO framework has been applied in various disciplines [7, 37], as well as Multi-agent Reinforcement Learning (MARL) [48].

Inspired by successful applications of DO framework, we, for the first time, propose a Double Oracle Framework for Generative Adversarial Networks (DO-GAN). In this chapter, we present four key contributions. First, we treat the generator and the discriminator as players and obtain the best responses from their oracles and add the utilities to a meta-matrix. Second, we propose a linear program to obtain the probability distributions of the players' pure strategies (meta-strategies) for the respective oracles. The linear program computes an exact mixed NE of the meta-matrix game in polynomial time. Third, since multiple generators and discriminator from the best responses oracles are stored in the memory, the algorithm may be memory-inefficient for problems to train GAN with large-scaled real-world datasets. Thus, we propose two solutions for the scalable double oracle framework: 1) a pruning method for reducing the support set of best response strategies to prevent the oracles from becoming intractable as there is a risk of the meta-matrix growing very large with each iteration of oracle training; 2) applying continual learning to retain the previous knowledge of the networks for the best responses from the generator and discriminator oracles in the multi-task learning setup. We also address the problems in continual learning such as catastrophic forgetting. Finally, we provide comprehensive evaluation on

the performance of DO-GAN with different GAN architectures using both synthetic and real-world datasets. Experiment results show that DO-GAN variants have significant improvements in terms of both subjective qualitative evaluation and quantitative metrics such as inception score and FID score.

5.1 DO-GAN

As discussed in previous sections, computing mixed NE for GANs is challenging as there is an extremely large number of pure strategies, i.e., possible parameter settings of the generator and discriminator networks. Thus, we propose a double oracle framework for GANs (DO-GAN) to compute the mixed NE efficiently. DO-GAN builds a restricted meta-matrix game between the two players and computes the mixed NE of the meta-matrix game, then DO-GAN iteratively adds more generators and discriminators into the meta-matrix game until termination.

5.1.1 General Framework of DO-GAN

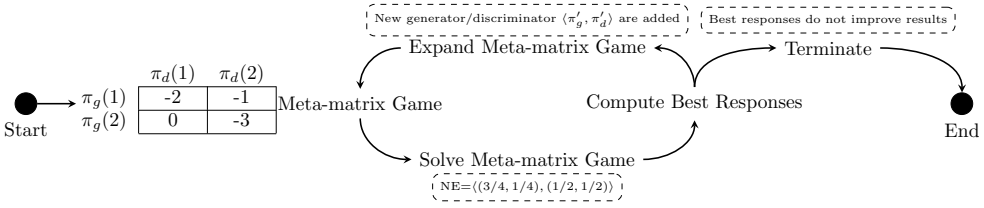


Figure 5.2: An illustration of DO-GAN. Figure adapted from [48].

GAN can be translated as a two-player zero-sum game between the generator player g and the discriminator player d. To compute the mixed NE

of GANs, at iteration t, DO-GAN creates a restricted meta-matrix game U^t with the trained generators and discriminators as strategies of the two players, where the generators and discriminators are parameterized by $\pi_g \in \mathcal{G}$ and $\pi_d \in \mathcal{D}$. We use $U^t(\pi_g, \pi_d)$ to denote the generator player's payoff when playing π_g against π_d, which is defined as L_D. Since GAN is zero-sum, the discriminator player's payoff is $-U^t(\pi_g, \pi_d)$. We define σ_g^t and σ_d^t as the mixed strategies of generator player and discriminator player, respectively. With a slight abuse of notation, we define the generator player's expected utility of mixed strategies $\langle \sigma_g^t, \sigma_d^t \rangle$ as $U^t(\sigma_g^t, \sigma_d^t) = \sum_{\pi_g \in \mathcal{G}} \sum_{\pi_d \in \mathcal{D}} \sigma_g^t(\pi_g) \cdot \sigma_d^t(\pi_d) \cdot U^t(\pi_g, \pi_d)$. We use $\langle \sigma_g^{t*}, \sigma_d^{t*} \rangle$ to denote mixed NE of the restricted meta-matrix game U^t. We solve U^t to obtain the mixed NE, compute best responses and add them into U^t for next iteration. Figure 5.2 presents an illustration of DO-GAN and Algorithm 5.1 describes the overview of the framework.

Algorithm 5.1: `DO − GAN()`

```
Initialize generator and discriminator arrays G = ∅ and D = ∅;
Train generator & discriminator to get the first π_g and π_d;
G ← G ∪ {π_g}; D ← D ∪ {π_d};
Compute the adversarial loss L_D and add it to meta-matrix U^0;
Initialize σ_g^{0*} = [1] and σ_d^{0*} = [1];
for epoch t ∈ {1, 2, ...} do
    π'_g ← generatorOracle(σ_d^{t*}, D);
    G ← G ∪ {π'_g};
    π'_d ← discriminatorOracle(σ_g^{t*}, G);
    D ← D ∪ {π'_d};
    Augment U^{t−1} with π'_g and π'_d to obtain U^t and compute missing entries;
    Compute mixed NE ⟨σ_g^{t*}, σ_d^{t*}⟩ for U^t with linear program;   // Section 5.1.2
    if TerminationCheck(U^t, σ_g^{t*}, σ_d^{t*}) then break;   // Section 5.1.3
```

Our algorithm starts by initializing two arrays $\mathcal{G}$ and $\mathcal{D}$ to store multiple

generators and discriminators (line 1). We train the first π_g and π_d with the canonical training procedure of GANs (line 2). We store the parameters of trained models in $\mathcal{G}$ and $\mathcal{D}$ (line 3), compute the adversarial loss L_D and add it to the meta-matrix U^0 (line 4). We initialize the meta-strategies $\sigma_g^{0*} = [1]$ and $\sigma_d^{0*} = [1]$ since there is only one pair of generator and discriminator available (line 5). For each epoch, we use `generatorOracle`() and `discriminatorOracle`() to obtain best responses π'_g and π'_d to σ_d^{t*} and σ_g^{t*} via Adam Optimizer, respectively, then add them into $\mathcal{G}$ and $\mathcal{D}$ (lines 7-10). We then augment U^{t-1} by adding π'_g and π'_d and calculating $U^t(\pi'_g, \pi'_d)$ to obtain U^t and compute the missing entries (line 11). We compute the missing payoff entries $U^t(\pi'_g, \pi_d), \forall \pi_d \in \mathcal{D}$ and $U^t(\pi_g, \pi'_d), \forall \pi_g \in \mathcal{G}$ by sampling a few batches of training data. After that, we compute the mixed NE $\langle \sigma_g^{t*}, \sigma_d^{t*} \rangle$ of U^t with linear programming (line 12). The algorithm terminates if the criteria described in Algorithm 5.4 is satisfied (line 13).

In `generatorOracle`(), we train π'_g to obtain the best response against σ_d^{t*}, i.e., $U^t(\pi'_g, \sigma_d^{t*}) \geq U^t(\pi_g, \sigma_d^{t*}), \forall \pi_g \in \Pi_g$. Similarly, in `disciminatorOracle`(), we train π'_d to obtain the best response against σ_g^{t*}, i.e., $U^t(\sigma_g^{t*}, \pi'_d) \geq U^t(\sigma_g^{t*}, \pi_d)$, $\forall \pi_d \in \Pi_d$. Full details of generator oracle and discriminator oracle can be found in Algorithms 5.2 and 5.3.

We train the oracles for some iterations which we denote as $k_{0,1,2,\ldots}$. For experiments, we train each oracle for an epoch for the real-world datasets and 50 iterations for the 2D Synthetic Gaussian Dataset. At each iteration t,

Algorithm 5.2: `GeneratorOracle`$(\sigma_d^{t*}, \mathcal{D})$

Initialize a generator G with random parameter setting π'_g;
for *iteration* $k_0 \dots k_n$ **do**
    Sample noise $\mathbf{z}$;
    π_d = Sample a discriminator from $\mathcal{D}$ with σ_d^{t*};
    Initialize a discriminator D with parameter setting π_d;
    Update the generator G's parameters π'_g via Adam optimizer:

$$\bigtriangledown_{\pi'_g} \log\left(1 - D(G(\mathbf{z}))\right)$$

Algorithm 5.3: `DiscriminatorOracle`$(\sigma_g^{t*}, \mathcal{G})$

Initialize a discriminator D with random parameter setting π'_d;
for *iteration* $k_0 \dots k_n$ **do**
    Sample a minibatch of data $\mathbf{x}$;
    for *a minibatch* **do**
        Sample noise $\mathbf{z}$;
        π_g = Sample a generator from $\mathcal{G}$ with σ_g^{t*};
        Initialize a generator G with a parameter setting π_g;
        Generate and add to mixture $G(\mathbf{z})$;
    Update the discriminator D's parameters π'_d via Adam optimizer:

$$\bigtriangledown_{\pi'_d} \log D(\mathbf{x}) + \log\left(1 - D(G(\mathbf{z}))\right)$$

we sample the generators from the support set $\mathcal{G}$ with the meta-strategy σ_g^{t*} to generate the images for evaluation. Similarly, we conduct the performance evaluation with the generators sampled from $\mathcal{G}$ with the final σ_g^* at termination. SGAN consists of a top-down stack of GANs, e.g, for a stack of 2, Generator 1 is the first layer stacked on Generator 0 with each of them connected to Discriminator 1 and 0 respectively. Hence, in DO-SGAN, we store the meta-strategies for the Generator 0 and 1 in σ_g^{t*} and the Discriminator 1 and 0 for σ_d^{t*}. In `GeneratorOracle`(), we first sample Discriminator 1 and 0 from discriminator distribution σ_d^{t*} and train Generator 1 first then followed

by calculating loss with Discriminator 1 and train Generator 0 subsequently, and finally calculate final loss with Discriminator 0 and train the whole model end to end. We perform the same process for `DisciminatorOracle`().

5.1.2 Linear Program for Meta-matrix Game

Since the current restricted meta-matrix game U^t is a zero-sum game, we can use a linear program to compute the mixed NE in polynomial time [86]. Given the generator player g's mixed strategy σ_g^t, the discriminator player d will play strategies that minimize the expected utility of g. Thus, the mixed NE strategy for the generator player σ_g^{t*} is to maximize the worst-case expected utility, which is obtained by solving the following linear program:

$$\sigma_g^{t*} = \arg\max_{\sigma_g^t}\{v : \sigma_g^t \geq 0, \sum\nolimits_{i\in\mathcal{G}} \sigma_g^t(i) = 1, U^t(\sigma_g^t, \pi_d) \geq v, \forall \pi_d \in \mathcal{D}\}. \quad (5.1)$$

Similarly, we can obtain the mixed NE strategy for the discriminator σ_d^{t*} by solving a linear program that maximizes the worst-case expected utility of the discriminator player. Therefore, we obtain the mixed NE $\langle \sigma_g^{t*}, \sigma_d^{t*} \rangle$ of the restricted meta-matrix game U^t.

5.1.3 Termination Check

DO terminates the training by checking whether the best response π_g' (or π_d') is in the support set $\mathcal{G}$ (or $\mathcal{D}$) [37], but we cannot apply this approach to DO-GAN as GAN has infinite-dimensional strategy space [33]. Hence,

Algorithm 5.4: `TerminationCheck`$(U^t, \sigma_g^{t*}, \sigma_d^{t*})$

```
   // U^t is of size m × n
   // |G| = m, |D| = n
1  Compute U^t(σ_g^{t*}, σ_d^{t*});
2  Compute U^t(σ_g^{t*}, D[n]);
3  Compute U^t(G[m], σ_d^{t*});
4  genInc = U^t(G[m], σ_d^{t*}) − U^t(σ_g^{t*}, σ_d^{t*});
5  disInc = −U^t(σ_g^{t*}, D[n]) − (−U^t(σ_g^{t*}, σ_d^{t*}));
6  if genInc < ϵ && − disInc < ϵ then
7      return True
8  else return False ;
```

we terminate the training if the best responses cannot bring a higher utility to the two players than the entries of the current support sets, as discussed in [48, 71]. Specifically, we first compute $U^t(\sigma_g^{t*}, \sigma_d^{t*})$ and the expected utilities for new generator and discriminator $U^t(\mathcal{G}[m], \sigma_d^{t*}), U^t(\sigma_g^{t*}, \mathcal{D}[n])$ (line 1-3). Then, we calculate the utility increment (lines 4-5) and returns **True** if both $U^t(\mathcal{G}[m], \sigma_d^{t*})$ and $U^t(\sigma_g^{t*}, \mathcal{D}[n])$ cannot bring a higher utility than $U^t(\sigma_g^{t*}, \sigma_d^{t*})$ by ϵ (lines 6-8).

5.2 Practical Implementations

As the number of epochs grows during the training of DO, the number of networks and the size of the meta-matrix also grows. Hence, there is a risk that the support strategy set becomes very large and $\mathcal{G}$ and $\mathcal{D}$ become intractable. To make the algorithm practical and scalable, we propose two methods: DO-GAN with meta-matrix pruning (DO-GAN/P) and DO-GAN with continual learning (DO-GAN/C).

5.2.1 Meta-matrix Pruning (DO-GAN/P)

Algorithm 5.5: DO-GAN/P

// I stores indices to be pruned from $\mathcal{G}$ and $\mathcal{D}$
// G stores models to be pruned from $\mathcal{G}$ and $\mathcal{D}$
1 $\texttt{DO} - \texttt{GAN}()$;
2 $I_g = \emptyset; I_d = \emptyset$
3 $K_g = \emptyset; K_d = \emptyset$ **if** $|\mathcal{G}| > s$ **then**
4 **for** $i \in \{0, \ldots, |\mathcal{G}| - 1\}$ **do**
5 **if** $\sigma_g^{t*}(\mathcal{G}[i]) == \min \sigma_g^{t*}$ **then** $I_g \leftarrow I_g \cup \{i\}$; $K_g \leftarrow K_g \cup \{\mathcal{G}[i]\}$;
6 **if** $|\mathcal{D}| > s$ **then**
7 **for** $j \in \{0, \ldots, |\mathcal{D}| - 1\}$ **do**
8 **if** $\sigma_d^{t*}(\mathcal{D}[j]) == \min \sigma_d^{t*}$ **then** $I_d \leftarrow I_d \cup \{j\}$; $K_d \leftarrow K_d \cup \{\mathcal{D}[j]\}$;
9 $\mathcal{G} \leftarrow \mathcal{G} \setminus K_g$; $\mathcal{D} \leftarrow \mathcal{D} \setminus K_d$;
10 $U \leftarrow J_{I_g,m} \cdot U^t \cdot J_{I_d,n}^T$

The first method is to prune the meta-matrix. Here, we adapt the greedy pruning algorithm, as depicted in Algorithm 5.5. When either $|\mathcal{G}|$ or $|\mathcal{D}|$ is greater than the limit of the support set size s, we prune at least one strategy with the least probability, which is the strategy that contributes the least to the player's winning. Specifically, we define $J_{I,b}$ where I is the set of row numbers to be removed, b is the total rows of a matrix. To remove the 2^{nd} row of a matrix having 3 rows, we define $I = \{1\}, b = 3$ and $J_{\{1\},3} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$. If $|\mathcal{G}| > s$, at least one strategy with minimum probability is pruned from $\mathcal{G}$, similarly for $\mathcal{D}$ (lines 3-9). Finally, we prune the meta-matrix using matrix multiplication (line 10).

Algorithm 5.6: DO-GAN/C

Initialize generator and discriminator task arrays $\mathcal{G} = \emptyset$ and $\mathcal{D} = \emptyset$;
Train generator & discriminator to get with the first task to get π_g^0 and π_d^0;
$\mathcal{G} \leftarrow \mathcal{G} \cup \{\pi_g\}$; $\mathcal{D} \leftarrow \mathcal{D} \cup \{\pi_d\}$;
Compute the adversarial loss L_D and add it to meta-matrix U^0;
Initialize $\sigma_g^{0*} = [1]$ and $\sigma_d^{0*} = [1]$;
for *epoch* $t \in \{1, 2, ...\}$ **do**
    Create new tasks π_g^t and π_d^t;
    $\pi_g^t \leftarrow \texttt{generatorOracle}(\sigma_d^{t*}, \mathcal{D})$;
    $\mathcal{G} \leftarrow \mathcal{G} \cup \{\pi_g^t\}$;
    $\pi_d^t \leftarrow \texttt{discriminatorOracle}(\sigma_g^{t*}, \mathcal{G})$;
    $\mathcal{D} \leftarrow \mathcal{D} \cup \{\pi_d^t\}$;
    if $t \geq 2$ **then**
        $\mathcal{G} \setminus \{\pi_g^{t-2}\}$ and $\mathcal{D} \setminus \{\pi_d^{t-2}\}$;
    Create U^t with π_g^{t-1}, π_g^t and π_d^{t-1}, π_d^t;
    Compute mixed NE $\langle \sigma_g^{t*}, \sigma_d^{t*} \rangle$ for U^t with linear program; *// Section 5.1.2*
    if $\texttt{TerminationCheck}(U^t, \sigma_g^{t*}, \sigma_d^{t*})$ **then break**; // Section 5.1.3

5.2.2 Continual Learning (DO-GAN/C)

Our ablation studies show that we still need a support set of at least $s = 10$ for DO-GAN/P to converge and the time complexity grows as s increases. Thus, we further reduce both time and space complexity by making the network retain the knowledge of previous networks so that the algorithm will converge with even smaller support set. Hence, we propose to adapt continual learning to consolidate the knowledge of multiple networks to a single network while setting $s = 2$ to reduce the space complexity as much as possible. We treat each network as a task to train the adaptive continual learning network while having a distribution over the tasks to represent the player's strategies.

To remedy the catastrophic forgetting i.e., all the generator tasks focus only towards fooling the newest discriminator, we adapt Elastic Weight Con-

solidation (EWC) method [54]. We change the generator loss function from non-saturating to saturating and add a penalty function accordingly. Let G be trained on task $t-1$ to have optimal parameters π_g^{t-1}, the Fisher information F is:

$$F = \mathbb{E}_{\mathbf{z}\sim p(\mathbf{z})}\left[\left(\frac{\delta}{\delta \pi_g^{t-1}} \log D(G(\mathbf{z})|\pi_g^{t-1})\right)^2\right] \tag{5.2}$$

After obtaining the Fisher information, we directly use F as a regularization loss to penalize the weight change during the training. Hence, G's loss function is augmented as:

$$L_G = \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}(\mathbf{z})}[-\log D(G(\mathbf{z}) + \lambda \cdot \sigma_g^{t*} \cdot \sum\nolimits_i F_i(\pi_i - \pi_g^{t-1,i})^2 \tag{5.3}$$

where π_g^{t-1} represents the parameters learned for task $t-1$, i is the index of each parameter of the generator model, and λ is the regularization weight.

Algorithm 5.6 describes the changes to $\texttt{DO}-\texttt{GAN}$ algorithm with continual learning. Instead of arrays $\mathcal{G}$ and $\mathcal{D}$ in $\texttt{DO}-\texttt{GAN}$, we initialize tasks arrays for generator and discriminator (line 1). At every epoch, we create new tasks and train the generator and discriminator networks to outperform the previous optimal parameters with the distribution at NE σ_d^{t*} and σ_g^{t*} respectively (lines 7-11). Then, we keep the previously and currently trained optimal parameters (line 13) to create meta-matrix, solve it to compute mixed-NE (lines 14-15). Finally, we perform the termination check (line 16).

Complexity. *Given the same architectures, the space complexity of DO-GAN*

is $\mathcal{O}(t)$ where t is the number of epochs until convergence. In contrast, the space complexity of DO-GAN/P is $\mathcal{O}(s)$ where s is the size of the support set. DO-GAN/C has the minimum storage complexity which is $\mathcal{O}(1)$.

In DO-GAN, we add a pair of generator and discriminator for every epoch of training. Thus, the space complexity of DO-GAN is $\mathcal{O}(t)$, making the algorithm memory-inefficient to train with real-world datasets where it needs a large number of epochs to converge. The space complexity of DO-GAN/P is $\mathcal{O}(s)$ since we prune the meta-matrix and the players' strategies if the $|\mathcal{G}| > s$ or $|\mathcal{D}| > s$ where s is the limit of the support set size. In DO-GAN/C, we train a single adaptive network storing the optimal strategies only for the tasks created at $(t-1)^{th}$ and t^{th} epochs. Thus, the space complexity of DO-GAN/C is kept at $\mathcal{O}(1)$.

5.3 Experiments

We conduct our experiments on a machine with Xeon(R) CPU E5-2683 v3@2.00GHz and 4× Tesla v100-PCIE-16GB running Ubuntu operating system. We evaluate DO-framework for established GAN architectures such as vanilla GAN [24], DCGAN [77], SNGAN [66] and SGAN [35]. We adopt the parameter settings and criterion of the GAN architectures as published. We set $s = 10$ unless mentioned otherwise. We compute the mixed NE of the meta game with Nashpy. According to the ablation studies by [54, 87], we

set λ as 1000 for MNIST, 5000 for CIFAR-10 and 5×10^8 for CelebA.

Table 5.1: Hyperparameters used in DO-GAN for different base architectures

	GAN	DCGAN	SNGAN	SGAN
Generator Learning Rate	0.0002	0.0002	0.0002	0.0001
Discriminator Learning Rate	0.0002	0.0002	0.0002	0.0001
batch size	64	64	64	100
Adam: beta 1	0.5	0.5	0.5	0.5
Adam: beta 2	0.999	0.999	0.999	0.999

We implement our proposed method with Python 3.7, Pytorch=1.4.0 and Torchvision=0.5.0. We set the hyperparameters as the original implementations. We present the hyperparameters set in Table 5.1. We use Nashpy to compute the equilibria of the meta-matrix game.

5.3.1 Value of λ

The experiment in [54] has done ablation studies for FFHQ dataset which are emoji faces and hence we used the λ value for CelebA. Meanwhile, we adopted the results from [87] and set 1000 for MNIST and the maximum value of ablation study for SVHN dataset to train CIFAR-10 as we want to use λ values from the most similar datasets. The experiments in [87] reported that they observed little difference in visual quality regarding with ablation study but high values of λ cause no loss in visual fidelity when beginning training on a new task rather than lower value of λ. Hence, we use the maximum value for λ.

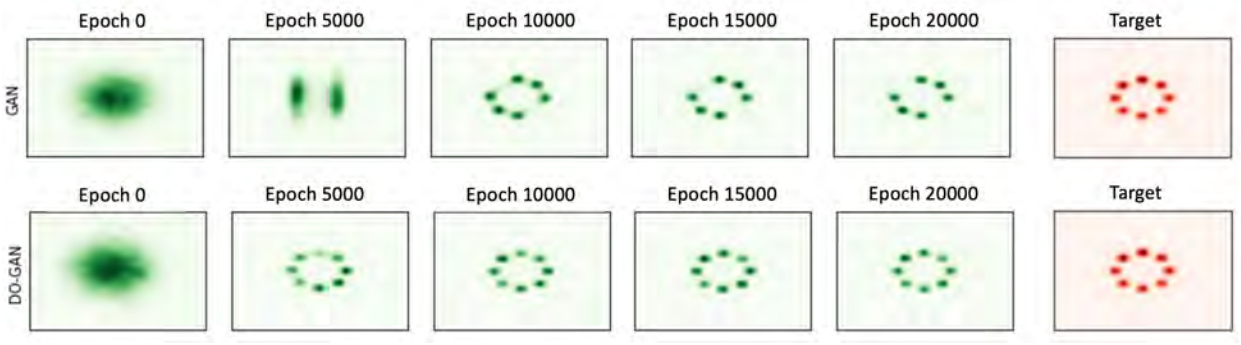
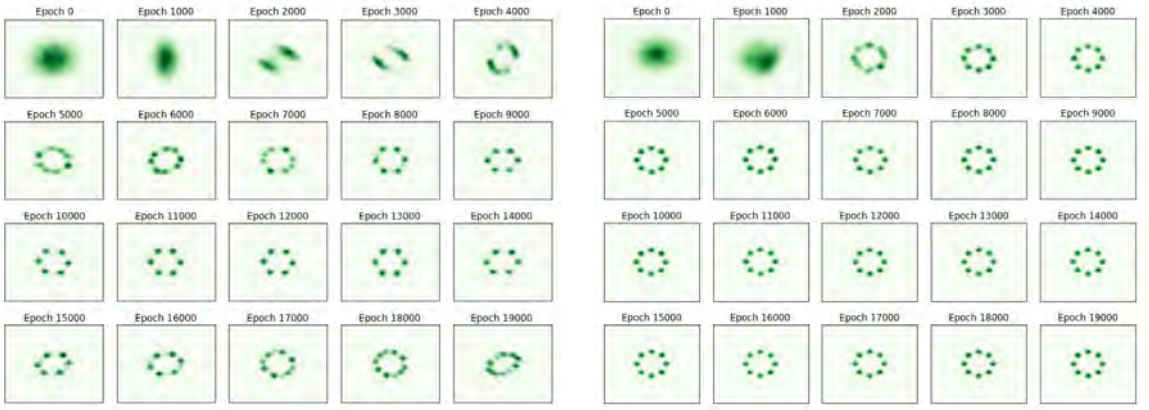


Figure 5.3: Comparison of GAN and DO-GAN/P on 2D synthetic Gaussian Mixture Dataset

5.3.2 Evaluation on 2D Gaussian Mixture Dataset

To illustrate the effectiveness of the architecture, we train a double oracle framework with the simple vanilla GAN architecture on a 2D mixture of 8 Gaussian mixture components with cluster standard deviation 0.1 which follows the experiment by [64]. Figure 5.3 shows the evolution of 512 samples generated by GAN and DO-GAN/P through 20000 epochs. The goal of GAN and DO-GAN/P is to correctly generate samples at 8 modes as shown in the target. The results show that GAN can only identify 6 out of 8 modes of the synthetic Gaussian data distribution, while the DO-GAN/P can obtain all the 8 modes of the distribution. Furthermore, DO-GAN/P takes shorter time (less than 5000 epochs) to identify all 8 modes of the data distribution. We present a more detailed evolution of data samples through the training process on 2D Gaussian Mixtures in Section 5.3.3.



(a) GAN (b) DO-GAN/P

Figure 5.4: Full comparison of GAN and DO-GAN/P on 2D Synthetic Gaussian Dataset

5.3.3 Full Training Process of 2D Gaussian Dataset

Figure 5.4 shows the full training process of DO-GAN/P and GAN on 2D Synthetic Gaussian Dataset. From the results, we find that GAN struggles to generate the samples into 8 modes while DO-GAN/P can generate all the 8 modes of the distribution. Furthermore, DO-GAN/P takes shorter time (less than 5000 iterations) to identify all 8 modes of the data distribution.

Table 5.2: Runtime of DO-GAN/P on 2D Gaussian Dataset with $s = 5, 10, 15$

Support Set Size	Runtime (GPU hours)
$s = 5$	> 1
$s = 10$	0.5627
$s = 15$	0.9989

Ablations. We vary the support set size s to $5, 10, 15$ and record the training evolution and the running time as presented in Table 5.2 and Figure 5.5. We find that if the support size is too small, e.g., $s = 5$, the best responses which are not optimal yet have better utilities than the models in the support set

are added and pruned from the meta-matrix repeatedly making the training not able to converge. However, $s = 15$ takes a significantly longer time as the time for the augmenting of meta-matrix becomes exponentially long with the support set size. Hence, we chose $s = 10$ as our experiment support set size since we observed that there is no significant trade-off and shorter runtime.

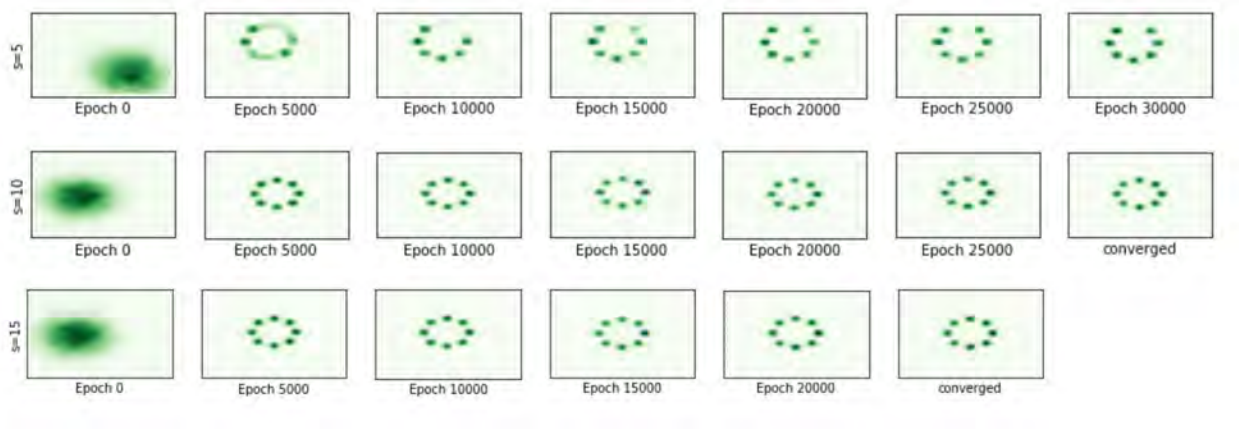


Figure 5.5: Training evolution on 2D Gaussian Dataset with $s = 5, 10, 15$

5.3.4 Choice of GAN Architectures for Experiments

We carried out experiments evaluate the performance of the DO framework by choosing several established GAN architectures from GAN as the backbone. We refer to the taxonomy of GANs [99] and choose each architecture from the groups of GANs focused on Network Architectures i.e., convolutional layers for deep neural networks of GAN: DCGAN, stacked architecture: SGAN as well as Latent Space and Normalization techniques: SNGAN as shown in Figure 5.6. We have also included comparisons with mixture architectures such as MIXGAN and MGAN.

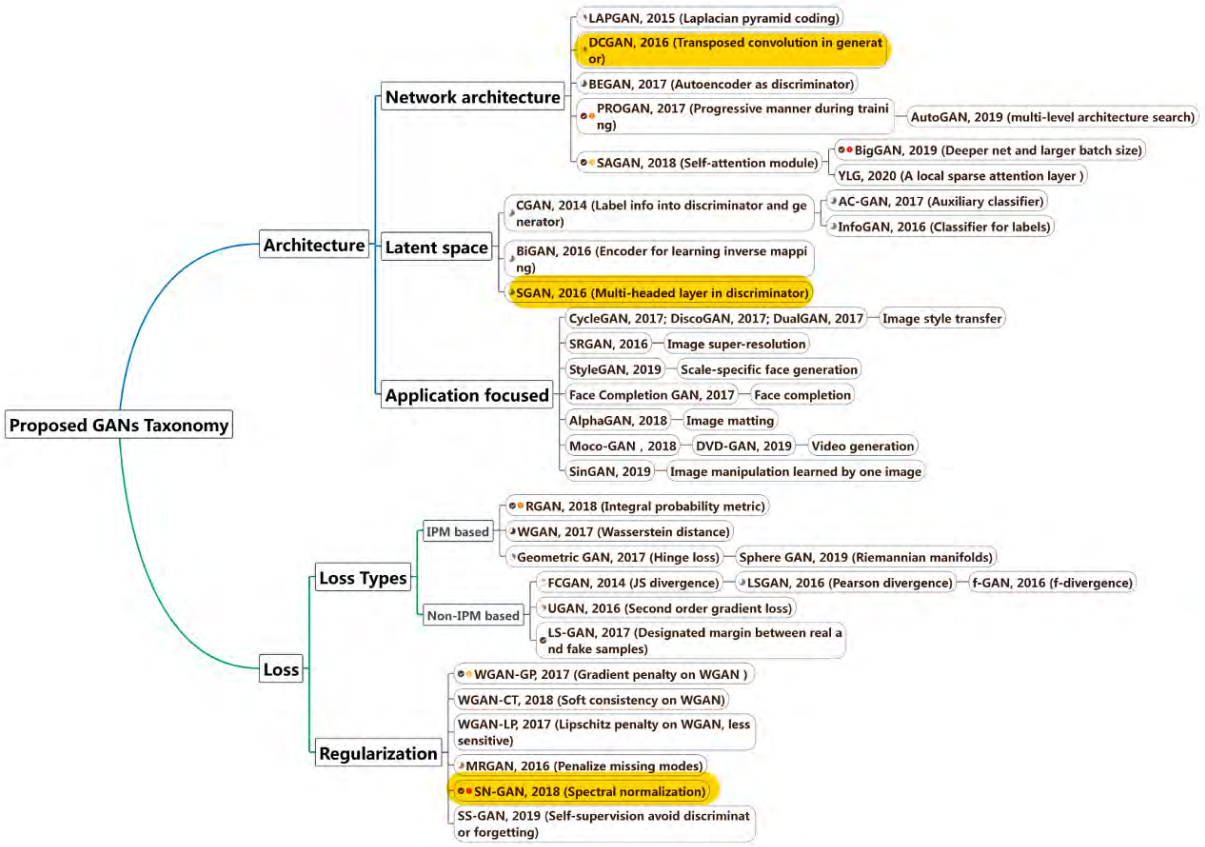


Figure 5.6: Taxonomy of GAN Architectures from [99]

5.3.5 Evaluation on Real-world Datasets

We run experiments on MNIST [50], CIFAR-10 [46] and CelebA [59]. MNIST contains 60,000 samples of handwritten digits with images of 28×28. CIFAR-10 contains $50,000$ training images of 32×32 of 10 classes. CelebA is a large-scaled face dataset with more than 200K images of size 128×128.

Qualitative Evaluation.

We choose the CelebA dataset for the qualitative evaluation since the training images contain noticeable artifacts (aliasing, compression, blur) that make the generator difficult to produce perfect and faithful images. We compare

performances of DO-DCGAN/P, DO-SNGAN/P and DO-SGAN/P with their counterparts. SNGAN which is trained for 40 epochs with termination ϵ of 5×10^{-5} for DO-SNGAN/P where other architectures are trained for 25 epochs with termination ϵ of 5×10^{-5} for DO variants. The generated CelebA images of DCGAN and DO-DCGAN/P are shown in Figure 5.7, where we find that DCGAN suffers mode-collapse, while DO-DCGAN/P does not. We also present the generated images of SNGAN vs DO-SNGAN/P using fixed noise at different training epochs in Figure 5.8. From the results, we can see that SNGAN, SGAN, DO-SNGAN/P and DO-SGAN/P are able to generate various faces, i.e., no mode-collapse. Judging from subjective visual quality, we find that DO-SNGAN/P and DO-SGAN/P are able to generate plausible images faster than SNGAN and SGAN during training, i.e., 17 epochs for DO-SGAN/P and 20 epochs for SGAN.

Figure 5.7: Training images with fixed noise for DCGAN and DO-DCGAN/P until termination.

Quantitative Evaluation.

Inception Score. We first leverage the Inception Score (IS) [85] by using Inception_v3 [91] as the inception model. To compute the inception score, we first compute the Kullback-Leibler (KL) divergence for all generated images and use the equation $\text{IS} = \exp(\mathbb{E}_{\mathbf{x}}[\text{KL}(D(p(y|\mathbf{x}) \parallel p(y)))])$ where $p(y)$ is the conditional label distributions for the images in the split and $p(y|\mathbf{x})$ is that of the image $\mathbf{x}$ estimated by the reference inception model. Inception score evaluates the quality and diversity of all generated images rather than the similarity to the real data from the test set.

FID Score. Fréchet Inception Distance (FID) measures the distance between the feature vectors of real and generated images using Inception_v3 model [30]. Here, we let p and q be the distributions of the representations obtained by projecting real and generated samples to the last hidden layer of Inception model. Assuming that p and q are the multivariate Gaussian distributions, FID measures the 2-Wasserstein distance between the two distributions. Hence, FID Score can capture the similarity of generated images to real ones better than inception score.

Results. The results are shown in Table 5.3. In CIFAR-10 dataset, the pruning method i.e., DO-GAN/P, DO-DCGAN/P and DO-SNGAN/P obtain much better results (7.2 ± 0.16, 7.86 ± 0.14 and 8.55 ± 0.08) than GAN,

Figure 5.8: Training images with fixed noise for SNGAN and DO-SNGAN/P until termination.

DCGAN and SNGAN (3.84 ± 0.09, 6.32 ± 0.05 and 7.58 ± 0.12). However, we do not see a significant improvement in DO-SGAN/P compared to SGAN 8.62 ± 0.12 and 8.69 ± 0.10 since SGAN already can generate diverse images. We did not include IS for CelebA dataset as IS cannot reflect the real image quality for CelebA, as observed in [30]. In CIFAR-10 dataset, DO-GAN/P, DO-DCGAN/P, DO-SNGAN/P and DO-SGAN/P obtain much lower FID scores (31.44, 22.25, 16.56, 18.20) respectively. The trend follows in CelebA obtaining 7.11 for DO-DCGAN/P while 10.92 for DCGAN, 7.62 for SNGAN while 6.92 for DO-SNGAN/P, 6.98 for SGAN and 6.32 for DO-SGAN/P respectively. Although we see a significant improvement in the quality of DO-SGAN/P images, FID score for DO-SGAN/P is affected by distortions.

We observe the competitive results for continual learning method with the pruning method: DO-GAN/C, DO-DCGAN/C, DO-SNGAN/C and DO-

Table 5.3: Inception scores (higher is better) and FID scores (lower is better) of DO-GAN with pruning (DO-GAN/P) and continual learning (DO-GAN/C). The mean and standard deviation are drawn from running 10 splits on 10000 generated images.

	Inception Score		FID Score	
	MNIST	CIFAR-10	CIFAR-10	CelebA
GAN	1.04 ± 0.05	3.84 ± 0.09	71.44	-
DCGAN	1.26 ± 0.05	6.32 ± 0.05	37.66	10.92
SNGAN	1.35 ± 0.11	7.58 ± 0.12	25.50	7.62
SGAN	1.39 ± 0.09	8.62 ± 0.12	24.83	6.98
MIX+DCGAN	-	7.72 ± 0.09	-	-
MGAN	-	8.33 ± 0.10	26.70	-
DCGAN+ EWC	-	7.58 ± 0.07	25.51	-
DO-GAN/P	1.39 ± 0.09 (+0.35)	7.20 ± 0.16 (+3.36)	31.44 (−40.00)	-
DO-DCGAN/P	1.42 ± 0.11 (+0.16)	7.86 ± 0.14 (+1.54)	22.25 (−15.41)	7.11 (−3.81)
DO-SNGAN/P	$\mathbf{1.42 \pm 0.07}$ (+0.07)	8.55 ± 0.08 (+0.97)	18.20 (−7.30)	6.92 (−0.70)
DO-SGAN/P	1.42 ± 0.09 (+0.03)	$\mathbf{8.69 \pm 0.10}$ (+0.07)	**16.56** (−8.27)	**6.32** (−0.66)
DO-GAN/C	1.42 ± 0.10 (+0.38)	7.32 ± 0.30 (+3.48)	26.93 (−44.51)	-
DO-DCGAN/C	1.43 ± 0.13 (+0.17)	8.04 ± 0.22 (+1.72)	21.50 (−16.16)	7.16 (−3.76)
DO-SNGAN/C	1.43 ± 0.08 (+0.08)	8.54 ± 0.16 (+0.96)	19.57 (−5.93)	6.74 (−0.88)
DO-SGAN/C	$\mathbf{1.45 \pm 0.15}$ (+0.06)	$\mathbf{9.78 \pm 0.11}$ (+1.16)	**16.07** (−8.76)	**6.30** (−0.68)

Note: MIX+DCGAN and MGAN results are directly copied from [3, 32]. The magenta values are the improvements from non-DO counterparts.

SGAN/C obtain inception scores of 7.32 ± 0.30, 8.04 ± 0.22, 8.54 ± 0.16 and 9.78 ± 0.11 respectively as well as FID scores of 26.93, 21.50, 19.57 and 16.07 respectively for CIFAR-10 dataset. Moreover, 7.16, 6.74 and 6.30 respectively for CelebA dataset. We also compared with DCGAN+EWC which uses continual learning without the double oracle framework for CIFAR-10 dataset and obtained better results where DCGAN+EWC obtained inception score of 7.58 ± 0.07 and FID score of 25.51.

From the results, we can see that DO framework performs better than each of their original counterpart architectures with both methods: pruning and continual learning. More details can be found in Figure 5.9 where we

computer FID score against training epochs for CIFAR-10 dataset for SGAN and DO-SGAN/P. We can also see that continual learning method that uses least storage space can obtain competitive results with DO-GAN/P without memory storage limitations or pruning the players' strategies.

FID score against training epochs for CIFAR-10 dataset

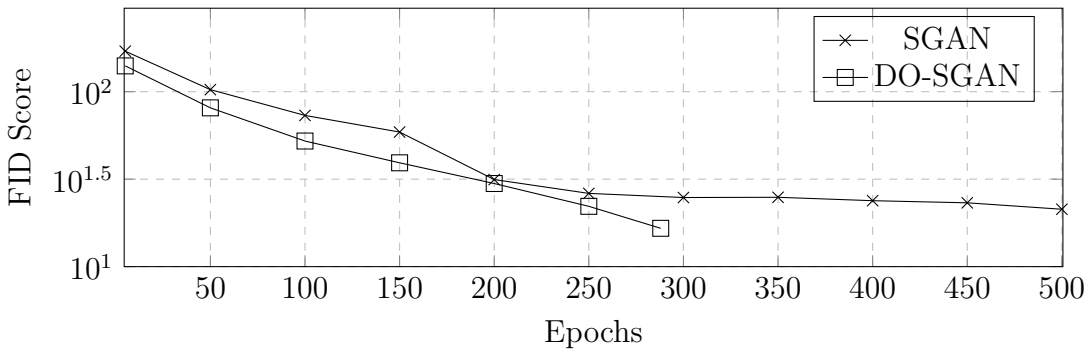


Figure 5.9: FID score vs. Epochs for SGAN and DO-SGAN trained on CIFAR-10

To compute FID score, we use Inception_v3 model with max pool of 192 dimensions and the last layer as coding layer as mentioned in [30]. We resized MNIST, CIFAR-10 generated and test images to 32×32 and CelebA images to 64×64. According to the figure, while both SGAN and DO-SGAN/P perform relatively well in generating plausible images, we can see that DO-SGAN/P terminates early at epoch 288 and has a better FID score of 16.56 compared to 24.83 at 300 epoch until 21.284 at 500 epoch for the training of SGAN.

Wall-clock Runtime

We also compare the wall-clock running times of DO-SGAN/P and DO-SGAN/C with SGAN as shown in Table 5.4. We let SGAN train for 500 epochs on CIFAR-10 dataset. Meanwhile, DO-SGAN/P converged at 288 epochs and DO-SGAN/C converged at 236 epochs. The recorded GPU hours of 143.28 for SGAN, 119.04 for DO-SGAN/P and 97.54 for DO-SGAN/C show that DO variants are more efficient.

Table 5.4: Wall-clock time comparison of SGAN variants

	SGAN	DO-SGAN/P	DO-SGAN/C
Time (GPU-Hrs)	143.28	119.04	**97.54**

Note: SGAN is trained for 500 epochs on CIFAR-10. DO-SGAN/P converged at 288 epochs and DO-SGAN/C at 236 epochs.

5.4 Chapter Summary

In this chapter, we propose DO-GAN which starts with a restricted game and incrementally adds the best responses of the generator and the discriminator oracles as the players' strategies. We then compute the mixed NE to get the players' meta-strategies by using a linear program. With scalable solutions such as pruning the support strategy set and continual learning, we apply DO-GAN approach to established GAN architectures such as vanilla GAN, DCGAN, SNGAN and SGAN. Extensive experiments with the synthetic 2D Gaussian mixture dataset as well as real-world datasets such as

MNIST, CIFAR-10 and CelebA show that DO-GAN variants have significant improvements in comparison to their respective GAN architectures in terms of both subjective image quality and quantitative metrics.

Chapter 6

DONAS: Double-Oracle and Neural Architecture Search for Adversarial Machine Learning

Most machine learning algorithms involve optimizing a single set of parameters to minimize a single cost function. In adversarial machine learning (AML), however, two or more "players" adapt their own parameters to minimize their own cost, in competition with the other players . There are two main schemes of AML: i) generative adversarial networks (GAN) where a generator network generates images, while the discriminator distinguishes the generated images from real images, resulting in the ability of the generator to produce realistic images [9, 24, 39], and ii) adversarial training (AT) where a classifier is trained against an attacker who can manipulate the inputs to decrease the performance of the classifier [25, 61]. In both cases, AML can be viewed as a two-player zero-sum game where each player is alternately

trained to maximize their respective utilities till convergence corresponding to a Nash Equilibrium (NE) [3].

Various approaches have shown promising results to optimize the network architecture in AML. Recent works propose mixed architectures with multiple generators and discriminators and consider mixed-NE, such as MGAN [32], MIX+GAN [3] and MirrorGAN [34]. DO-GAN proposes to train GANs by deploying a double oracle framework using generator and discriminator from the best response oracles. In these techniques, they only update the weights to find the best response networks using predefined architectures. On the other hand, Neural Architecture Search (NAS) approaches have also shown promising results. They have been applied to some computer vision tasks, such as image classification, dense image prediction and object detection. Recently, NAS has been researched in GANs [23, 82]. Particularly, Adversarial-NAS [20] searches both of the generator and discriminator simultaneously in a differentiable manner using gradient-based method NAS and a large architecture search space. Moreover, NAS is also used to search the classifier architectures in Adversarial Training (AT) for robustness. AdvRush [69] considers both white-box attacks and black-box attacks using the input loss landscape of the networks to represent their intrinsic robustness. However, the NAS techniques only derive one final architecture with a criterion such as maximum weight from the search space, ignoring other top performing results from the neural architecture search.

Thus, we propose DONAS, which combines **D**ouble **O**racle from game theory and **N**eural **A**rchitecture **S**earch to leverage the searched networks and the mixed-architectures in adversarial machine learning. We deploy the NAS algorithms: Adversarial-NAS for GANs and AdvRush for AT in the best response oracles of the training process.

In summary, we make four key contributions in this chapter: 1) A general algorithm for DONAS where we obtain the players' best responses from oracles, augment a meta-game and solve it with linear programming; 2) DONAS for GANs with the respective oracles to search multiple discriminators and generators referencing the techniques from Adversarial-NAS and finetune them sequentially with [a]: Harmonic Mean and [b]: Nash; 3) DONAS for AT with the classifier/attacker oracles for the double oracle adversarial training referencing the techniques from AdvRush to search the classifier model; 4) We conduct experiments on both generative adversarial networks and adversarial training, where DONAS can be viewed as an extension of NAS with the mixed strategy or an extension of DO with an augmented action space, i.e., architectures and parameters. Finally, we provide a comprehensive evaluation on the performance of DONAS for both GANs and AT with different Neural Architecture Search algorithms using real-world datasets. Experimental results show that DONAS variants have achieved significant generative results as well as improvement in robustness.

6.1 DONAS: The General Framework

We have discussed the two prominent approaches to optimizing network architecture in adversarial machine learning, mainly, DO as the mixed-architecture approach, and Adversarial-NAS and AdvRush as the NAS approach for GANs and AT, respectively. DO finds the best response architecture by only updating weights for predefined architecture in the oracles, while NAS methods select only one final network from the search space and may miss the information trained by other top searched architectures. To mitigate the issues in both methods, we propose DONAS approach to combine them, allowing searched networks in mixed-architecture framework.

We describe the general framework for our proposed DONAS. We adapt the double oracle framework, in which Neural Architecture Search is used as the best response oracle for the two players: generator-discriminator for GANs and classifier-attacker for AT. We search and sample the best response strategies for the two players with respective oracles. Then, we finetune to improve the quality of the image generation and robustness. To speed up the training, we introduce the terminating condition ϵ.

Algorithm 6.1 initializes the first pair of models and stores them in the two arrays $\mathcal{P}_1$ and $\mathcal{P}_2$. Then, we can randomly sample pure strategies to augment the meta-game and compute the distribution of the player strategies. We initialize the meta-strategies $\sigma_{p1}^{1*} = [1]$ and $\sigma_{p2}^{1*} = [1]$ since only one pair of

Algorithm 6.1: DONAS: General Framework

1 $P_1, P_2 =$ `InitializeModels`();
2 $\mathcal{P}_1 \leftarrow \{P_1\}$, $\mathcal{P}_2 \leftarrow \{P_2\}$, $\sigma_{p1}^{1*} = \sigma_{p2}^{1*} = [1]$;
3 **for** *epoch* $t = 1, 2, ...$ **do**
4 $\quad P_1 =$ `Player1_NASSearch`();
5 $\quad \Pi_1 =$ `SampleFromSupernet`(P_1);
6 $\quad P_2 =$ `Player2_NASSearch`();
7 $\quad \Pi_2 =$ `SampleFromSupernet`(P_2);
8 $\quad \mathcal{P}_1 \leftarrow \mathcal{P}_1 \cup \Pi_1$, $\mathcal{P}_2 \leftarrow \mathcal{P}_2 \cup \Pi_2$;
9 $\quad$ `FineTuning`();
10 $\quad$ Augment U^{t-1} with P_1 and P_2 to obtain U^t
11 $\quad$ Compute mixed-NE $\langle \sigma_{p1}^{t*}, \sigma_{p2}^{t*} \rangle$ for U^t
12 $\quad$ **if** `TerminationCheck`$(U^t, \sigma_{p1}^{t*}, \sigma_{p2}^{t*})$ **then**
13 $\quad\quad$ **break**;

player strategies is available (line 1-2). For each epoch, we search the new strategy for each player (line 4, 6), obtaining the supernets P_1 and P_2. Next, we sample the networks to derive the architecture from supernet (line 5,7). We then finetune the selected models (line 9). In DONAS for AT, we do not have sampling and finetuning for the best response oracle for attacker player since we do not need to search architecture for the attacker player. Next, we generate the meta-game U^t and compute mixed NE with linear programming to obtain σ_{p1}^* and σ_{p2}^* (line 10-11). The algorithm terminates if the terminating condition ϵ is satisfied (line 12).

In Algorithm 6.2, we first compute $U^t(\sigma_{p1}^{t*}, \sigma_{p2}^{t*})$ and the expected utilities for new player strategies $U^t(\mathcal{P}_1[m], \sigma_{p2}^{t*}), U^t(\sigma_{p1}^{t*}, \mathcal{P}_2[n])$ (line 2-5). Then, we calculate the utility increment for the two players (lines 6-7) and return **True** if both $U^t(\mathcal{P}_1[m], \sigma_{p2}^{t*})$ and $U^t(\sigma_{p1}^{t*}, \mathcal{P}_2[n])$ cannot bring a higher utility than $U^t(\sigma_{p1}^{t*}, \sigma_{p2}^{t*})$ by the margin of ϵ, terminating the training process (lines 8-11).

Algorithm 6.2: `TerminationCheck`$(U^t, \sigma_{p1}^{t*}, \sigma_{p2}^{t*})$

```
1  Compute U^t(σ_{p1}^{t*}, σ_{p2}^{t*});
2  // U^t is of size m × n
3  Compute U^t(σ_{p1}^{t*}, P_2[n]);
4  // |P_1| = m, |P_2| = n;
5  Compute U^t(P_1[m], σ_{p2}^{t*});
6  δ_{p_1} = U^t(P_1[m], σ_{p2}^{t*}) − U^t(σ_{p1}^{t*}, σ_{p2}^{t*});
7  δ_{p_2} = U^t(σ_{p1}^{t*}, σ_{p2}^{t*}) − U^t(σ_{p1}^{t*}, P_2[n]);
8  if δ_{p_1} < ϵ && − δ_{p_2} < ϵ then
9      return True
10 else
11     return False
```

Pruning Models for Memory Efficiency

To reduce the memory during the training, we prune the arrays $\mathcal{P}_1$ and $\mathcal{P}_2$ and update the respective parameters when $|\mathcal{P}_1| > K$ or $\mathcal{P}_2 > K$. We propose a memory-efficient solution of DONAS by adapting the greedy pruning algorithm from Chapter 5. Algorithm 6.3 describes `PruneModels`(). I stores indices to be pruned from $\mathcal{P}_1$ and $\mathcal{P}_2$ and G stores models to be pruned from $\mathcal{P}_1$ and $\mathcal{P}_2$. When either $|\mathcal{P}_1|$ or $|\mathcal{P}_2|$ is greater than the limit of the size K, we prune at least one strategy with the least probability, which is the strategy that contributes the least to the player's winning and prunes the meta-game by maxtrix multiplication accordingly. (lines 2-12).

Algorithm 6.3: `PruneModels`$(\mathcal{P}_1, U^t, \mathcal{P}_2, U^t, \sigma_g^*, \sigma_d^*)$

```
1  I_p1 = ∅; I_p2 = ∅;
2  if |P_1| > K then
3      for (i, P) ∈ P_1 do
4          if σ_p1^{t*}(P) == min σ_p1^{t*} then
5              P_1 ← P_1 \ {P};
6              I_p1 ← I_p1 ∪ {i};
7  if |P_2| > K then
8      for (j, P) ∈ P_2 do
9          if σ_p2^{t*}(P) == min σ_p2^{t*} then
10             P_2 ← P_2 \ {P};
11             I_p2 ← I_p2 ∪ {j}
12 U ← J_{I_p1,m} · U^t · J^T_{I_p2,n}
```

6.2 Spices of DONAS

In this section, we present two spices of the general framework DONAS for GAN and AT. We introduce the modular components to make it adaptable to any NAS algorithm for future uses.

6.2.1 DONAS for GAN

We use DONAS for GANs to allow the use of multiple generators and discriminators. We let $\mathcal{P}_1 = \mathcal{G}$ and $\mathcal{P}_2 = \mathcal{D}$ as the two-player zero-sum game between the generator and discriminator with their architectures as α and β respectively. Theoretically, using multiple generators and discriminators with mixed strategies covers multiple modes to produce better generated results.

Algorithm 6.4 describes the specific changes to adapt DONAS for GANs. We initialize the first generator-discriminator pair G, D and finetune the models in `InitializeModels`() (line 1-2). The two-player oracles are shown in

Algorithm 6.4: DONAS for GANs

Procedure: `InitializeModels`()

Initialize the generator G and discriminator D ;

`FineTune`(`G`) ;

Procedure: `GeneratorOracle`()

G = initialize();

for *s steps* **do**

  Sample mini-batch of $2m$ noise samples;

  Update the architecture of generator:

  $\nabla_\alpha \frac{1}{m} \sum_{i=1}^{m} [\sum_{j=1}^{|\mathcal{D}|} \sigma_d^{j*} \cdot \log(1 - D_j(G(z^i)))]$

  Update the weights of generator:

  $\nabla_{W_G} \frac{1}{m} \sum_{i>m}^{2m} [\sum_{j=1}^{|\mathcal{D}|} \sigma_d^{j*} \cdot \log(1 - D_j(G(z^i)))]$;

Procedure: `DiscriminatorOracle`()

D = initialize();

for *s steps* **do**

  Sample $2m$ samples for noise and real data Update the architecture of discriminator:

  $\nabla_\beta \frac{1}{m} \sum_{i=1}^{m} [\log x^i + \sum_{j=1}^{|\mathcal{G}|} \sigma_g^{j*} \cdot \log(1 - D(G_j(z^i)))]$;

  Update the weights of discriminator:

  $\nabla_{W_D} \frac{1}{m} \sum_{i>m}^{2m} [\log x^i + \sum_{j=1}^{|\mathcal{G}|} \sigma_g^{j*} \cdot \log(1 - D(G_j(z^i)))]$;

Procedure: `SampleFromSupernet`(α)

Sample top k subnetworks $\bar{\alpha}$ according to α;

$\Pi = \text{argmin}_{\Pi \in \bar{\alpha}}$ adv_loss();

Procedure: `SequentialFineTune`()

for $r = 1, 2, ...$ **do**

  for $i = 1, \ldots, |\mathcal{G}|$ **do**

    $\max_{\theta_{g_i}} \mathbb{E}_z \left[\sum_{j=1}^{|\mathcal{D}|} \sigma_d^{j*} \cdot D_j(G_i(z)) \cdot \Theta \right]$;

  for $j = 1, \ldots, |\mathcal{D}|$ **do**

    $\max_{\theta_d} \mathbb{E}_x[\log D_j(x)] + \sum_{k=1}^{K} \mathbb{E}_z[\log(1 - D_j(G_k(z)))]$;

  Augment U^t and find mixed-NE (σ_g^*, σ_d^*);

`GeneratorOracle`() and `DiscriminatorOracle`() which describe the best response oracles in the DO-framework of a two-player min-max game with the networks (α and β) where the weight of each network must be the best response to the other player.

Adapting Adversarial-NAS. GAN optimization process in Adversarial-NAS is defined as a two-player min-max game with value function $V(\alpha, \beta)$ where the weight of each network must be the best response [20]. The min-max game is $\min_\alpha \max_\beta V(\alpha, \beta)$ defined as:

$$V(\alpha, \beta) = \mathbb{E}_{x \sim p_{data}}(x)[\log D(x|\beta, W_D^*(\beta)]+$$
$$\mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z|\alpha, W_G^*(\alpha))|\beta, W_D^*(\beta))], \tag{6.1}$$

where the two weights $W_G^*(\alpha), W_D^*(\beta)$ for any architecture pair (α, β) can be obtained through another min-max game between W_G and W_D, i.e., $\min_{W_G(\alpha)} \max_{W_D(\beta)} V(W_{G(\alpha)}, W_{D(\beta)})$ defined as:

$$V(W_{G(\alpha)}, W_{D(\beta)}) = \mathbb{E}_{x \sim p_{data}}(x)[\log D(x|\beta, W_{D(\beta)}]+$$
$$\mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z|\alpha, W_{G(\alpha)})|\beta, W_{D(\beta)}))]. \tag{6.2}$$

The weights of generator and discriminator w.r.t. the architecture pair (α, β), $\{W_{G(\alpha)}^*, W_{D(\beta)}^*\}$, can be obtained by a single step of adversarial training as vanilla GANs [20, 89]. The optimal architectures or weights in each iteration

can be achieved by ascending or descending the corresponding stochastic gradient by updating in the order of architecture followed by the weights.

Generator Oracle. We initialize the generator and sample the noise samples (line 5). Then, we search and update the architecture of the generator by ascending its stochastic gradient using the probability distribution σ_d^* for the pure strategies of the discriminator (line 7). Finally, we update the weight of the discriminator by the stochastic gradient ascent while considering σ_d^* (line 9).

Discriminator Oracle. Similarly, we initialize the discriminator and sample noise samples as well as real-data examples (line 12). Then, we update the architecture (line 13) and weights (line 15) by stochastic gradient ascent with the probability distribution σ_g^* for the pure strategies of the generator.

Sampling Architecture from Supernet. After the search, we derive the architecture from the resulting supernet (line 16- 17). We sample top k networks selecting maximum values of the architecture α and select the networks that give the minimum adversarial loss against the other player.

Finetuning. After the two oracles provide the new strategies for the two players (new generator and discriminator), we sequentially finetune them by Nash distributions σ_g^{0*} and σ_d^{0*} that we obtained from solving the meta-

game with linear programming (line 18-22). To finetune multiple generators altogether, we update the objective function of the generator in training with $\mathbb{E}_{(z\sim p_z(z))}[D(G_i(z,\theta_{gi}))]$ and the sequential generators with harmonic mean [96]. Hence, the generator update is $\max_{\theta g_i} \mathbb{E}_{z\sim p_z(z)}[D(G_i(z,\theta_{gi}))\cdot\Phi]$, where Φ is the harmonic mean of the remaining generators in the array: $\Phi = HM((1-D(G_{i-1}(z,\theta_{g_{i-1}})),\ldots,(1-D(G_1(z,\theta_{g_1}))))]$.

This ensures that the current generator focuses on the data samples from the modes ignored by the previous generators. Arithmetic means and geometric means cannot generalize the ignored data samples well in the case of sampled generators where one of the generators is performing much better. The Harmonic Mean makes sure the currently trained generator focuses on the ignored data samples. The details of sequential finetuning with HM are mentioned in Algorithm 6.5.

Algorithm 6.5: Fintuning K Generators by HM

1 **for** *iteration 1,2, ...* **do**
2 **for** $i = 1,\ldots,K$ **do**
3 Update the generators using Harmonic Mean:
4 $\max_{\theta_{g_i}} \mathbb{E}_{z\sim p_z(z)}[D(G_i(z,\theta_{gi}))\times HM((1-D(G_{i-1}(z,\theta_{g_{i-1}}))),\ldots,(1-D(G_1(z,\theta_{g_1}))))]$;
5 Update the discriminator:
6 $\max_{\theta_d} \mathbb{E}_{x\sim P_{data}}[\log D(x,\theta_d)] + \sum_{k=1}^{K}\mathbb{E}_{z-p_z(z)}[log(1-D(G_k(z,\theta_{g_k}),\theta_d))]$

For every epoch, each generator is updated according to Eq. (23) (line 3) followed by the discriminator update where the fake data from the generator is the combination of all K generators (line 5). After the training has finished, we augment the meta-game and solve it to obtain the distribution of K

generators to be used in the next iteration. However, HM only tells the ignored samples and cannot give a clear picture of which samples are being captured more than the others at each epoch of the finetuning.

Using the meta-game, which only takes linear complexity to solve, gives Nash distribution of the generators, and hence, the update of the objective function can have a better picture to focus on the less-captured data samples. Hence, this method is more effective than HM which focuses heavily on the ignored data samples.

Algorithm 6.6: Fintuning K Generators by Nash

1 **for** *iteration 1,2, ...* **do**
2 **for** $i = 1, \ldots, K$ **do**
3 Update the generators using the distribution σ:
4 $\max_{\theta_{g_i}} \mathbb{E}_{z \sim p_z(z)}[D(G_i(z, \theta_{gi})) \times \sigma_{i-1}(1 - D(G_{i-1}(z, \theta_{g_{i-1}}))) \times \cdots \times \sigma_1(1 - D(G_1(z, \theta_{g_1})))]$;
5 Update the discriminator:
6 $\max_{\theta_d} \mathbb{E}_{x \sim P_{data}}[\log D(x, \theta_d)] + \sum_{k=1}^{K} \mathbb{E}_{z - p_z(z)}[log(1 - D(G_k(z, \theta_{g_k}), \theta_d))]$;
7 Generate the augmented meta-game U^i;
8 Compute mixed-NE σ for U^i;

Algorithm 6.6 describes the finetuning of multiple generators and discriminators with Nash. Using a similar equation as Eq. (23) and $\Theta = \sigma_g^{i-1*}(1 - D_j(G_{i-1}(z, \theta_{g_{i-1}}))) \times \cdots \times \sigma_g^{1*}(1 - D_j(G_1(z, \theta_{g_1})))$, we finetune each generator and discriminator using Nash distribution obtained by generating the augmented meta-game and compute mixed NE by linear programming (line 6-7).

6.2.2 DONAS for AT

We also propose DONAS for AT where we have the two players as the training classifier $\mathcal{P}_1 = \mathcal{C}$ and the adversarial $\mathcal{P}_2 = \mathcal{A}$. We search the classifier in `ClassifierOracle`() against the perturbed datasets from `AttackerOracle`(). Given L_{val} and L_{train} as loss functions on the validation and training sets respectively, the search process of Neural Architecture Search is defined as a bi-level optimization problem:

$$\begin{aligned} &\min_\theta L_{val}(w^*(\theta), \theta) \\ &\text{s.t. } w^*(\theta) = \text{argmin}_w L_{train}(w, \theta). \end{aligned} \tag{6.3}$$

For NAS in the classification task, the expensive inner optimization in Eq. (6.3) is normally approximated by one step training [57] as: $\nabla_\theta L_{val}(w^*(\theta), \theta) \approx \nabla_\theta L_{val}(w - \xi \nabla_w L_{train}(w, \theta), \theta)$ where w denotes the current weights and ξ is the learning rate for a step of inner optimization.

Adapting AdvRush. Algorithm 6.7 describes the specific changes to adapt DONAS for AT. To obtain the first pair of classifier θ and perturbed dataset δ, we perform neural architecture search for θ by AdvRush [69] and set $\delta = 0$ in `InitializeModels`(), then store the parameters in the two arrays $\mathcal{C}$ and $\mathcal{A}$ (line 1-2). For each epoch, we search the new classifier and generate new perturbed dataset with `ClassierOracle`() and `AttackerOracle`() using the

Algorithm 6.7: DONAS for AT

input: Samples X, learning rate γ, the number of hopsteps m;

Procedure: `InitializeModels`()

θ = `AdvRushSearch`();

$\delta = 0$;

Procedure: `AttackerOracle`()

Let $\delta = 0$;

for *step* $s = 1, 2, ..., \frac{N}{m}$ **do**

  for *mini-batch* $B \subset X$ **do**

    for $i = 1, 2, \ldots, m$ **do**

      $g_{adv} = \nabla_{x,\theta\sim(C,\sigma_c^*)} l(x+\delta, y, \theta), x, y \in B$;

      Update δ;

      $\delta \leftarrow$ `Clip`$(\delta + \epsilon \cdot sign(g_{adv}), -\epsilon, \epsilon)$;

Procedure: `ClassifierOracle`()

Initialize architecture and weight (w_0, α_0);

for *iteration* $i = 1, 2, ...$ **do**

  Update weights by descending its stochastic gradient:

  $\nabla_W L_{train}(w_i - 1, \alpha_i - 1)$;

  Update the architecture, i.e., α:

$$\begin{cases} \nabla_\alpha [L_{val}(w_i, \alpha_i - 1)], & \text{if } t < \phi \\ \nabla_\alpha [L_{val}(w_i, \alpha_i - 1) + \gamma L_\lambda], & \text{else} \end{cases}$$

Procedure: `SampleFromSupernet`(θ)

Derive θ through discritization steps from DARTS;

Procedure: `FineTune`()

for *steps* $s = 1, 2, ..., \frac{N}{m}$ **do**

  for *mini-batch* $B \subset X$ **do**

    for $i = 1, 2, \ldots, m$ **do**

      $g_\theta \leftarrow \mathbb{E}_{(x,y)\in B}[\nabla_{\theta,\delta\sim(\dashv,\sigma_a^*)} l(x+\delta, y, \theta)]$;

      $\theta \leftarrow \theta - \gamma \cdot g_\theta$;

meta-strategies σ_c^* and σ_p^*. We refer to the Free adversarial training (Free AT) algorithm [89], with comparable robustness to the traditional 7-step PGD [61] with significantly faster training. We then convert it to the DO-Framework with the two oracles to allow multiple adversarial updates to be made to the same images without multiple backward passes by training the same mini-batch for m times.

Attacker Oracle. For each iteration, we calculate the adversarial gradient using $\theta \in \mathcal{C}$ sampled with probability distribution of the classifier's strategies from the meta-game σ_c^* to update δ (line 7-8).

Classifier Oracle. We adapt AdvRush to search for the architecture and update the weights. According to the ablation study by [69], we set $\gamma = 0.01$ and AdvRush loss term L_λ is introduced at $\phi = 50$ which we set as iteration number for warm-up process. (line 15).

Sampling Architecture from Supernet. After search and update of classifier's architecture and weights, we derive the final architecture (line 16) by running discretization procedure of DARTS [57].

Finetuning. We adversarially train the classifier against the attacker. We update θ with stochastic gradient descent (line 20-21) with δ sampled from $\mathcal{A}$ with σ_a^*. We consecutively train each mini-batch for m times.

Finally, we augment the meta-game by calculating the cross-entropy loss and solve it to obtain the distribution. DONAS-AT ends when the terminating criteria is satisfied meaning both oracles have searched the best response strategies.

6.3 Experiments

6.3.1 Experimental Setup

We run the experiments on 8 Tesla V100-SXM2-32GB service with Ubuntu Operating System. We set the number of generators as $K = 5$ or $K = 10$ for DONAS and multiple generator samplings of Adversarial-NAS as mentioned in the results. Following Adversarial-NAS, we set the hyper-parameters of optimizers for training the weights of both generator and discriminator as $\beta_1 - 0.0, \beta_2 = 0.9$ and learning rate is set to 0.0002. The hyper-parameters of optimizers for both architectures are set to $\beta_1 = 0.5, \beta_2 = 0.9$ and the learning rate is 0.0003 with the weight decay of 0.0001. We set the batch size as 100 for both generator and discriminator, run the search, save them as genotypes (npz) and finally finetune them.

Similarly in AT, we follow AdvRush for the search process. We use SGD to update w with batch size = 32, learning rate = 0.025, $\beta = 0.9$, and weight decay factor = $3e-4$. We use Adam to update α with learning rate = $3e-4$, $\beta 1 = 0.5, \beta_2 = 0.999$, and weight decay factor = $1e-3$. We run the updates

for 60 epochs, setting $\phi = 50$. For AT, we set the clip value to 4.0 and the number of repeat updates $m = 4$. We keep the hyper-parameters of the optimizer in DONAS for GAN and AT the same as the respective settings in searching with $\epsilon = 5e - 3$ as terminating condition.

6.3.2 Experiments on DONAS for GAN

As the first step, we conducted an experiment to evaluate DONAS-GAN on the datasets CIFAR-10 [46], STL-10 [11] and TinyImageNet [49]. We calculate FID score to evaluate the quantitative performance against the baselines AutoGAN [23] and Adversarial NAS [20] to compare with our DONAS-GAN variants where we investigated ablation studies with different approaches varying the number of generators and discriminators.

Variants of DONAS-GAN. Initially, we reproduce Adversarial-NAS search which searches 1 generator followed by sampling multiple top networks from the supernet after the search. We run the experiments for DONAS-GAN Ind., DONAS-GAN HM and DONAS-GAN Nash with K, where K is the number of generators, to select a sample of K generators instead of 1 maximum and finetune the generators. We set the number of discriminators as 1.

DONAS-GAN Ind. uses independent finetuning where all K generators are trained independently whereas DONAS-GAN HM and DONAS-GAN Nash use sequentially finetuning where the sampled K generators are finetuned by

Harmonic Mean (HM) and Nash.

Next, we search K generators iteratively in DONAS-GAN Iter. and sample the architecture with the maximum α instead of a one-time search. To avoid having the searching architecture for the same batches, we shuffle the train data for each iteration. After the iterative process, we finetune the resulting K generators by Nash.

Finally, we perform experiments with full-version of DONAS-GAN with the pruning limit of K for both generators and discriminators. We set K to prune the generator and discriminator arrays $\mathcal{G}$ and $\mathcal{D}$, and the terminating condition ϵ is now set as $5e^{-3}$.

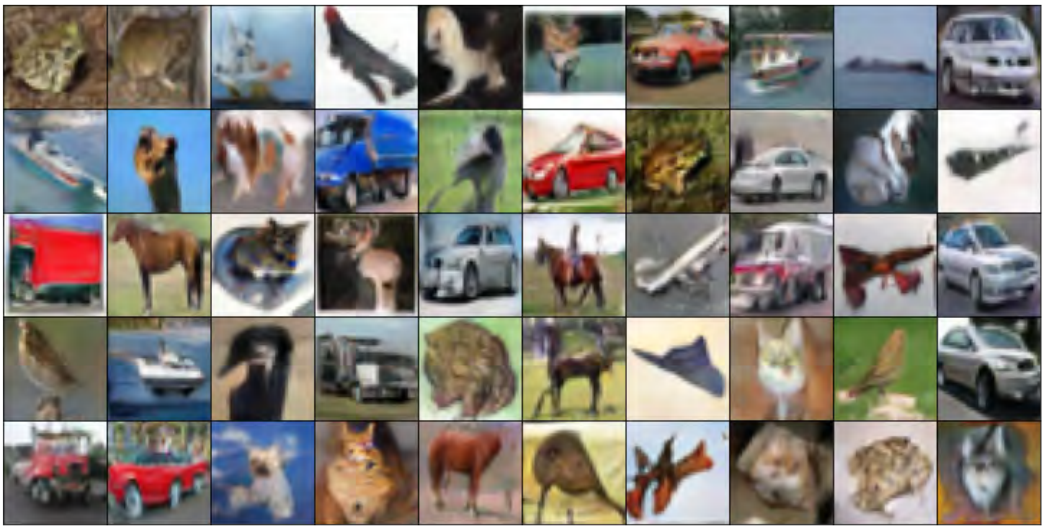
Table 6.1: Generative results for DONAS-GAN.

Methods			AutoGAN	AdvNAS	DONAS-GAN				
					Ind.	HM	Nash.	Iter.	Full
CIFAR-10	Search	5	-	16.35	14.22	14.22	14.22	14.43	-
		10			14.04	14.04	14.04	13.89	-
	Finetune	5	12.42	<u>10.87</u>	10.62	9.27	9.06	9.12	**8.93**
		10			10.33	9.32	9.19	9.10	**8.93**
STL-10	Search	5	-	32.48	29.16	29.16	29.16	30.02	-
		10			28.95	28.95	28.95	28.99	-
	Finetune	5	31.01	<u>26.98</u>	28.88	26.44	26.13	26.15	**25.31**
		10			28.72	26.60	26.17	25.82	**24.75**
Tiny ImageNet	Search	5	-	17.99	17.53	17.53	17.53	-	-
		10			16.91	16.91	16.91	-	-
	Finetune	5	16.21	<u>15.10</u>	15.21	13.85	12.36	-	**11.18**
		10			14.60	13.28	11.94	-	**10.42**

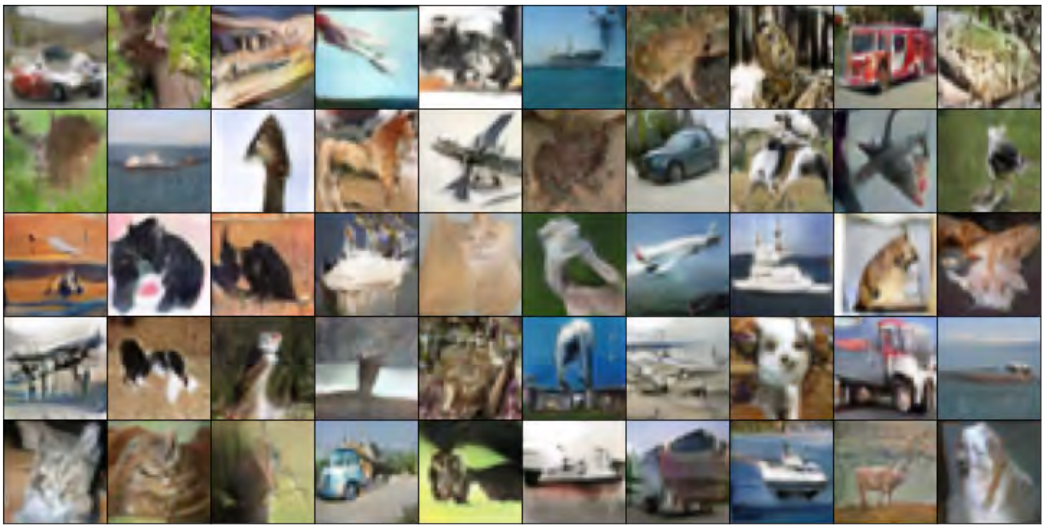
Experiments Results. Table 6.1 shows the experiment results with various methods and datasets investigated on DONAS-GAN. We initially set up the experiment for CIFAR-10 dataset and recorded AutoGAN as 12.42, Adversarial-NAS as 16.35 for the searched architecture and 10.87 after finetuning. For the variants of DONAS-GAN, we recorded $10.62, 9.27, 9.06, 9.12, \mathbf{8.926} (K = 5)$ and $10.33, 9.32, 9.19, 9.10, \mathbf{8.932} (K = 10)$ for DONAS-GAN Ind., HM, Nash, Iter and full-version of DONAS-GAN with K generators/discriminators. The results indicate that our variants bring significant improvements to the results. While DONAS-NAS Iter. suffers from long finetuning time as there is no terminating condition recording 60.88 GPU Hours for $K = 5$ and 109.52 GPU Hours for $K = 10$, DONAS-GAN shows satisfying results terminating within a training time of 54.19 GPU Hours for the best run.

Similar promising trends are observed in STL-10 datasets recording 31.01 and 26.98 for AutoGAN and Adversarial-NAS as baselines. We recorded $28.88, 26.44, 26.13, 26.15, \mathbf{25.31} (K = 5)$ and $28.72, 26.6, 26.17, 25.82, \mathbf{24.75}$ $(K = 10)$ for DONAS-GAN Ind., HM, Nash, Iter and full-version of DONAS-GAN with K generators/discriminators respectively. We also compared DONAS-GAN against baselines for TinyImageNet dataset and recorded 16.21 for AutoGAN, 15.10 for Adversarial-NAS and $11.18 (K = 5), 10.42 (K = 10)$ for DONAS-GAN showing that DONAS-GAN can capture the diversity of data examples better. Qualitative examples in Appendix also indicate the realistic image generation and the ability to capture the diversity of the classes

without mode-collapse. The generated images of the CIFAR-10 dataset using DONAS for GAN using K values as 5 and 10 for the qualitative evaluation are shown in Figure 6.1.

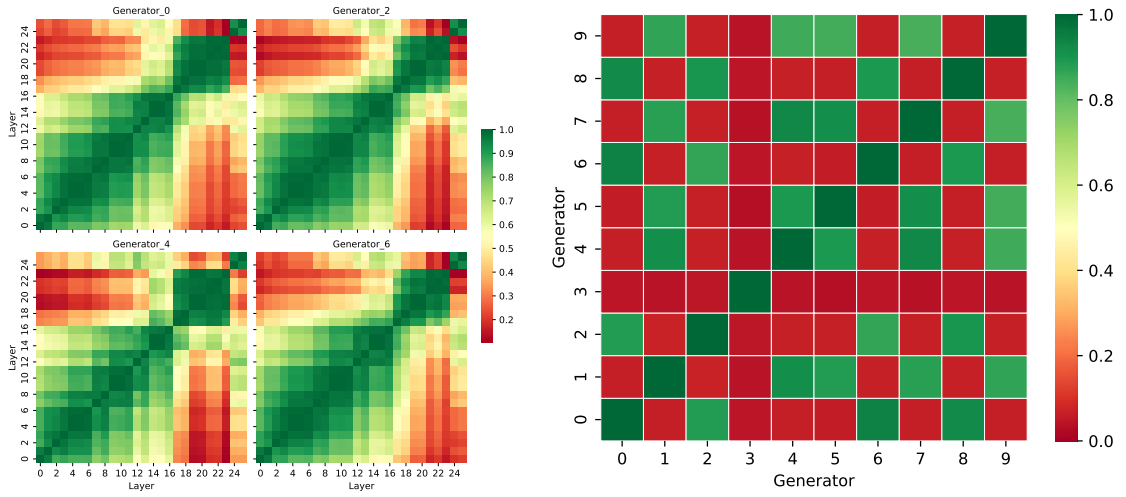


(a) Using $K = 5$



(b)
Using $K = 10$

Figure 6.1: The CIFAR-10 images randomly generated from DONAS for GAN



(a) (b)

Figure 6.2: (a): Linear CKA between layers of the individual searched networks of DONAS-GAN trained on TinyImageNet dataset. The downtrend of similarity for later layers is observed. (b): Linear CKA averaging between the same layers of the searched networks of DONAS-GAN trained on TinyImageNet.

Internal Representation of Networks. To analyze how the representation of the networks evolve and differ from each other, we use the centered kernel alignment (CKA) [44] to measure the similarity of the representation between layers and networks of DONAS-GAN trained on TinyImageNet. CKA is proposed by Kornblith et al to measure the similarity of the representation between layers and networks of DONAS-GAN trained on TinyImageNet. CKA is a commonly used metric for representation similarity, which features in higher accuracy comparing with other similarity indexes. In Figure 6.2a, we visualize the CKA values of layers in the same network of DONAS-GAN as heatmaps. We can see that there is a considerable similarity between the hidden layers as represented by the green area but the similarity goes down at the later layers. The trend indicates that the generation quality progressively improves with depth.

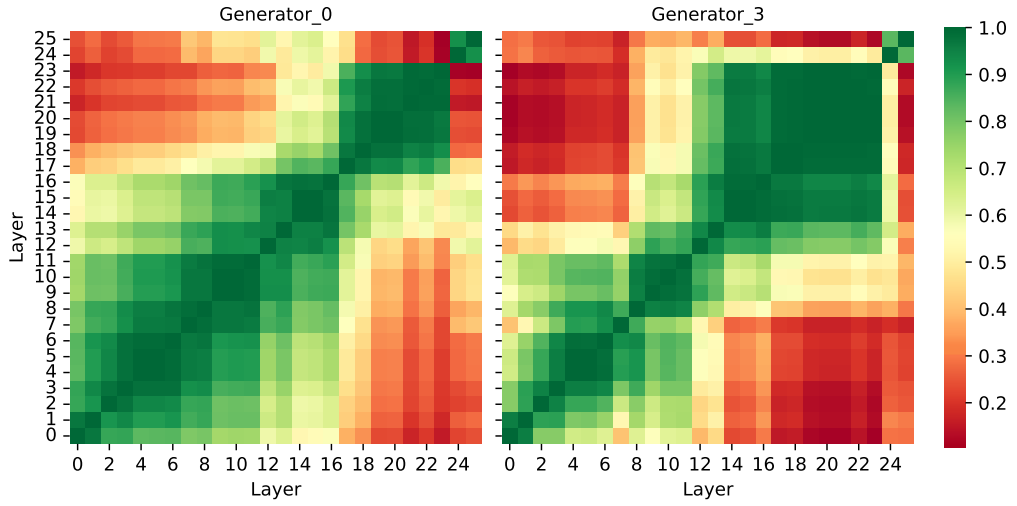


Figure 6.3: The search process of DONAS-GAN obtaining diverse architectures.

In Figure 6.2b, we visualize the similarity across the models. The result indicates that the search process of DONAS-GAN can obtain diverse architectures of generators to be used as mixed-architecture. Figure 6.3 also present the search process obtaining diverse architectures.

6.3.3 Experiments on DONAS for AT

We evaluate the robustness of the architectures searched and adversarially trained by DONAS-AT on CIFAR-10 using FGSM and PGD white-box-attacks [61]. In particular, we evaluate our proposed DONAS-AT with ResNet-18 [28], variations of RobNet [26] and finally, AdvRush [69] on CIFAR-10, TinyImageNet and SVHN datasets. According to [26], the model's robustness is better as we increase the computational budget and thus, we compare with RobNet-Large and RobNet-LargeV2 which have comparable network parameters (number of channels and cells) to WideResNet. Moreover, we also select RobNet-Free which relaxes the cell-based constraint. After searching

the classifier and training in DONAS-AT, we attack the model using PGD attacks where perturbation data is added to input for T iterations (PGD^T). We then use this perturbed input to the model for classification and calculate cross-entropy loss against the target label as the criterion. We test the classifiers using the PGD^{20} and increase the iterations to 100, i.e., PGD^{100}, to introduce stronger attacks.

Table 6.2: Robust accuracy under FGSM and PGD attacks.

Methods		ResNet-18	RobNet			AdvRush	DONAS-AT
			Large	LargeV2	Free		
CIFAR-10	FGSM	49.81%	54.98%	57.18%	59.22%	60.87%	**62.31%**
	PGD-20	45.86%	49.44%	50.53%	52.57%	53.07%	**59.57%**
	PGD-100	45.10%	49.24%	50.26%	51.14%	52.80%	**57.20%**
SVHN	FGSM	88.73%	93.01%	-	93.04%	94.95%	**95.91%**
	PGD-20	69.51%	86.52%	-	88.50%	91.14%	**91.40%**
	PGD-100	46.08%	78.16%	-	78.11%	90.48%	**91.83%**
Tiny ImageNet	FGSM	16.08%	22.16%	-	23.11%	25.20%	**28.11%**
	PGD-20	13.94%	20.40%	-	21.05%	23.58%	**26.30%**
	PGD-100	13.96%	19.90%	-	20.87%	22.93%	**24.10%**

Note: For ResNet-18 and RobNets, we presented the results from the papers [26, 69].

Experiments Results. Evaluation results are shown in Table 6.2. We observe promising results of DONAS-AT with 62.31% accuracy for FGSM attacks on CIFAR-10 dataset surpassing the baseline methods such as ResNet-18, RobNet-Large, RobNet-LargeV2, RobNet-Free and AdvRush that record $49.8\%, 54.98\%, 57.18\%, 59.22\%$, 60.87%. We also evaluated with PGD white-box attacks PGD^{20} and PGD^{100}. We obtain 59.57% accuracy on PGD^{20} white-box attacks and 57.20% for PGD^{100} observing similar trends which

demonstrates the stronger robustness of our approach.

Similar trends are observed in TinyImageNet and SVHN datasets. We record the best results $28.11\%, 26.30\%, 24.10\%$ against various attacks such as FGSM, PGD^{20}, PGD^{100}, respectively, for DONAS-AT on TinyImageNet dataset comparing ResNet-18, RobNet-Large, RobNet-Free and AdvRush. On the other hand, we record 95.91% against FGSM, 91.40% against PGD^{20} and 91.83% against PGD^{100}, respectively, for DONAS-AT on SVHN dataset showing the best results among the baseline methods. From the results, we can see that the results are better for AdvRush even with the cell-based constraints [69] due to the use of architecture search, DONAS-AT which adapts AdvRush to DO framework relaxes the cell-based micro search space using the best response from multiple search architectures. Hence, we can conclude that adding mixed-architecture to adversarial search and training (DONAS-AT) provides far better robustness against the white-box attacks than single model search algorithms both in cases of weaker attack PGD^{20} and very strong attack PGD^{100}.

6.4 Chapter Summary

In this chapter, we propose DONAS to combine the two methods to optimize the network architectures according to mixed-architecture and neural architecture search for searching and training GAN as well as for Adversarial Training (AT). We leverage two NAS algorithms: Adversarial NAS and

AdvRush to DO-framework, and introduce sequential finetuning to allow the finetuning of multi-models. Finally, we conduct extensive experiments to evaluate the performance of DO-NAS for GAN and AT against different NAS algorithms using different CIFAR-10 dataset. We show through our results that our approach achieves significant improvements to both *the generative results of searched and trained architecture for GANs* and *the robustness of the searched architecture against white-box attacks.*

Chapter 7

Conclusion

7.1 Conclusion

In this thesis, we investigate POMDP planning, deep reinforcement learning, game-theory, continual learning methodologies and extend them to handle problems with large discrete state and action spaces. Not only discrete problems, we also extend to the environments which have continuous state and action space.

In the first part of the thesis, we approach a large-discrete problem, i.e., the student counselling problem which has large number of participants. The action space grows exponentially with the number of added participants in the network and thus, makes the problem very difficult to solve. Moreover, we do not have complete information of the initial states of the participants. This uncertainty makes the problem more difficult to reach the Nash Equilibrium (NE) strategy.

Hence, we propose a novel model that considers emotion propagation from not only the influencer but also neighbours of the influencee while selecting the students for interventions in uncertain networks. We propose MLPRAP algorithm with reasoning, abstraction of POMDP states and multi-level partitioning of the graph into smaller POMDPs to sequentially plan to select the students for each intervention. Finally, we experiment with synthetic networks generated as BA and ER networks as well as real network datasets. We show that MLPRAP variants have significantly better scalability and solution quality comparable to the state-of-the-art algorithms.

As an alternative solution on the same problem, we propose a novel architecture DRLPSO to handle the partially observable dynamic environments with large action space. The DRLPSO algorithm uses the Deep Q-learning integrated with LSTM (DQ-LSTM) and DPSO to optimally select the predicted action, obtain Q-values and train the network. We use belief and history of action and observation as input to DQ-LSTM. Results have shown that in a dynamic environment with a large action space, the proposed DRLPSO achieves a higher total reward compared to the existing DRL approaches by an average of 32%. Comparing with the POMDP solver, DRLPSO takes more time for training but outperforms in terms of effectiveness.

In the second part of the thesis, we approach a continuous large-scale problem: Generative Adversarial Networks (GANs). GANs have been ap-

plied in various domains such as image and video generation, image-to-image translation and text-to-image synthesis. Various architectures are proposed to generate more realistic samples as well as regularization techniques. GANs can be seen as a two-player zero-sum game of infinite state and action choices with generator and discriminator as the two players. They are alternately trained to maximize their respective utilities till convergence corresponding to a pure Nash Equilibrium (NE). In this setting, GANs is challenging as a pure Nash equilibrium may not exist and even finding the mixed Nash equilibrium is difficult as GANs have a large strategy (action) space.

First, we propose a novel double oracle framework to GANs, which starts with a restricted game and incrementally adds the best responses of the generator and the discriminator oracles as the players' strategies. We then compute the mixed NE to get the players' meta-strategies by using a linear program. We also propose two approaches to make the solution scalable including pruning the support strategy set and continual learning with an adaptive architecture to store the multiple networks of generators and discriminators. We apply DO-GAN approach to established GAN architectures such as vanilla GAN, DCGAN, SNGAN and SGAN. Extensive experiments with the synthetic 2D Gaussian mixture dataset as well as real-world datasets such as MNIST, CIFAR-10 and CelebA show that DO-GAN variants have significant improvements in comparison to their respective GAN architectures in terms of both subjective image quality and quantitative metrics.

To further our investigation, we propose a novel approach to combine the two methods to optimize the network architectures according to mixed-architecture and neural architecture search. Combining the techniques such as DO from game theory and NAS, we propose a general framework: Double Oracle with NAS (DONAS) for searching and training GAN as well as for Adversarial Training (AT). We leverage two NAS algorithms Adversarial NAS and AdvRush to DO-framework, and introduce sequential finetuning to allow the finetuning of multi-models. Finally, we conduct extensive experiments to evaluate the performance of DO-NAS for GAN and AT against different NAS algorithms using different real-world datasets. We show through our results that our approach achieves significant improvements to both the generative results of searched and trained architecture for GANs and the robustness of the searched architecture against white-box attacks.

7.2 Future Work

In this thesis, we have discussed handling large-scale problems in both exponentially growing discrete action space and continuous action space. For the discrete action space, we have shown effective solutions by partitioning the problem or deep reinforcement learning algorithms. However, there are still many works that need to be done to address and explore solutions for problems with continuous action space.

7.2.1 Reducing Training Time for Neural Architecture Search

The first direction is to reduce the training time for Neural Architecture Search. In DO-GAN and DONAS, we train or search the generators and discriminators in the best response oracles. This step is the bottleneck of our approach as traditionally training GANs from scratch in an unsupervised manner takes a long time. Thus, we seek to investigate if the training time can be reduced with a comparable performance by using the pre-trained models for GANs. One preceding work for Style-GANs [47] proposed an effective selection mechanism from the pre-trained models by probing the linear separability between real and fake samples in pre-trained model embeddings, followed by choosing the most accurate model, and progressively adding to the ensemble. This work has shown impressive results in both limited and large-scale training samples for GANs. We can benefit from adapting selected pre-trained models for DO-GANs. Moreover, Neural Tangent Kernel (NTK) [36] has shown a promising theoretical framework that estimates the performance of a neural architecture at initialization, and Label-Gradient Alignment (LGA) [70] obtains a meaningful level of rank correlation with the post-training test accuracy of the searched architecture for adversarial training such as DARTS. These two metrics can successfully guide the search algorithm to achieve competitive search performances within a few epochs. However, no work has been done to investigate on NAS for GANs as well as a double oracle setting. Thus, we can adapt these methods to initialize the

search step with pre-trained models or use a performance estimation metric in NAS for GANs and evaluate the training time and performance.

7.2.2 DO-GAN/ DONAS for Time Series Datasets

The second direction is to adapt the DO-GAN solutions to not only GAN architecture for image generation but also the GAN architectures for sequential datasets. We can adapt the double oracle framework to established GAN architectures optimized for generating sequential data. Examples include COT-GAN [104] which combines classic optimal transport methods with an additional temporal causality constraint to generate videos and animated images, CurvGAN [53] and GraphGAN [52] which are GAN-based graph representation methods and TimeGAN [109] to generate realistic time-series data. This direction of adapting DO to videos and graphs data would better reflect real-life problems in society and will also improve general time-series problems such as forecasting and extrapolation.

7.2.3 Hybrid (discrete-continuous) Action Space

The third direction is to further investigate hybrid state/action spaces where some variables are discrete while some are continuous. One example is algorithmic trading [73] where the percentage changes in prices is modelled as the continuous action space with the limit price choices a user can make is discrete due to the existence of the tick size. In addition to the reinforcement

learning methods proposed in recent years [51, 73], we can investigate more solutions handling the large-scale problems with hybrid action space such as generative methods for forecasting as well as neural architecture search algorithms.

Bibliography

[1] An, B., Shieh, E., Tambe, M., Yang, R., Baldwin, C., DiRenzo, J., Maule, B., and Meyer, G. Protect–a deployed game theoretic system for strategic security allocation for the united states coast guard. *Ai Magazine 33*, 4 (2012), 96–96. 1

[2] Arjovsky, M., Chintala, S., and Bottou, L. Wasserstein generative adversarial networks. In *ICML* (2017), pp. 214–223. 2, 72

[3] Arora, S., Ge, R., Liang, Y., Ma, T., and Zhang, Y. Generalization and equilibrium in generative adversarial nets (GANs). In *ICML* (2017), pp. 224–232. 7, 15, 73, 93, 98

[4] Barabási, A.-L., and Albert, R. Emergence of scaling in random networks. *Science 286*, 5439 (1999), 509–512. 46

[5] Bellemare, M. G., Naddaf, Y., Veness, J., and Bowling, M. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research 47* (2013), 253–279. 52

[6] Blackmore, E. R., Stansfeld, S. A., Weller, I., Munce, S., Zagorski, B. M., and Stewart, D. E. Major depressive episodes and work stress: Results from a national population survey. *American Journal of Public Health 97*, 11 (2007), 2088–2093. 11

[7] Bošanský, B., Kiekintveld, C., Lisy, V., Cermak, J., and Pechoucek, M. Double-oracle algorithm for computing an exact Nash equilibrium in zero-sum extensive-form games. In *AAMAS* (2013), pp. 335–342. 6, 74

[8] Bosansky, B., Kiekintveld, C., Lisy, V., and Pechoucek, M. Iterative algorithm for solving two-player zero-sum extensive-form games with imperfect information. In *ECAI* (2012), pp. 193–198. 22

[9] Brock, A., Donahue, J., and Simonyan, K. Large scale GAN training for high fidelity natural image synthesis. In *ICLR* (2018). 7, 97

[10] Chen, W., Lin, T., Tan, Z., Zhao, M., and Zhou, X. Robust influence maximization. In *KDD* (2016), pp. 795–804. 13

[11] Coates, A., Ng, A., and Lee, H. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics* (2011), JMLR Workshop and Conference Proceedings, pp. 215–223. 113

[12] Conitzer, V., and Sandholm, T. Computing the optimal strategy to commit to. In *Proceedings of the 7th ACM conference on Electronic commerce* (2006), pp. 82–90. xi, 20, 21

[13] Egorov, M. Deep Reinforcement Learning with POMDPs. Tech. rep., Technical Report, Stanford University, 2015. 62

[14] Erdos, P. On random graphs. *Publicationes Mathematicae 6* (1959), 290–297. 46, 62

[15] Eyre, R. W., House, T., Hill, E. M., and Griffiths, F. E. Spreading of components of mood in adolescent social networks. *Royal Society Open Science 4*, 9 (2017), 170336. 12, 24, 36, 46

[16] Fang, F., Nguyen, T. H., Pickles, R., Lam, W. Y., Clements, G. R., An, B., Singh, A., Schwedock, B. C., Tambe, M., and Lemieux, A. Paws—a deployed game-theoretic application to combat poaching. *AI Magazine 38*, 1 (2017), 23–36. 1

[17] Farnia, F., and Ozdaglar, A. GANs may have no Nash equilibria. *arXiv preprint arXiv:2002.09124* (2020). 3, 5, 72

[18] Foerster, J. N., Assael, Y. M., de Freitas, N., and Whiteson, S. Learning to communicate to solve riddles with deep distributed recurrent Q-networks. *arXiv preprint arXiv:1602.02672* (2016). 19, 62

[19] Fowler, J. H., and Christakis, N. A. Dynamic spread of happiness in a large social network: Longitudinal analysis over 20 years in the framingham heart study. *The BMJ 337* (2008), a2338. 12

[20] Gao, C., Chen, Y., Liu, S., Tan, Z., and Yan, S. AdversarialNAS: Adversarial neural architecture search for gans. In *CVPR* (2020), pp. 5680–5689. 7, 18, 98, 105, 113

[21] Geraeinejad, V., Sinaei, S., Modarressi, M., and Daneshtalab, M. RoCo-NAS: Robust and compact neural architecture search. In *IJCNN* (2021). 17

[22] Goel, S., Watts, D. J., and Goldstein, D. G. The structure of online diffusion networks. In *Proceedings of the 13th ACM Conference on Electronic Commerce* (2012), pp. 623–638. 12

[23] Gong, X., Chang, S., Jiang, Y., and Wang, Z. AutoGAN: Neural architecture search for generative adversarial networks. In *ICCV* (2019), pp. 3224–3234. 98, 113

[24] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. In *NeurIPS* (2014), pp. 2672–2680. 3, 7, 72, 84, 97

[25] Goodfellow, I. J., Shlens, J., and Szegedy, C. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572* (2014). 7, 97

[26] Guo, M., Yang, Y., Xu, R., Liu, Z., and Lin, D. When NAS meets robustness: In search of robust architectures against adversarial attacks. In *CVPR* (2020), pp. 631–640. 17, 118, 119

[27] Hausknecht, M., and Stone, P. Deep recurrent Q-learning for partially observable MDPs. In *2015 AAAI Fall Symposium Series* (2015). 19

[28] He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *CVPR* (2016), pp. 770–778. 118

[29] Helmers, K. F., Danoff, D., Steinert, Y., Leyton, M., and Young, S. N. Stress and depressed mood in medical students, law students, and graduate students at mcgill university. *Academic Medicine 72*, 8 (1997), 708–714. 11

[30] Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., and Hochreiter, S. GANs trained by a two time-scale update rule converge to a local Nash equilibrium. In *NeurIPS* (2017), pp. 6626–6637. 91, 92, 94

[31] Hill, E., Griffiths, F., and House, T. Spreading of healthy mood in adolescent social networks. *Proc. R. Soc. B 282*, 1813 (2015), 20151180. 35, 46

[32] Hoang, Q., Nguyen, T. D., Le, T., and Phung, D. MGAN: Training generative adversarial nets with multiple generators. In *ICLR* (2018). 7, 15, 73, 93, 98

[33] Hsieh, Y.-P., Liu, C., and Cevher, V. Finding mixed nash equilibria of generative adversarial networks. In *ICML* (2019), pp. 2810–2819. 6, 14, 15, 73, 79

[34] Hsieh, Y.-P., Liu, C., and Cevher, V. Finding mixed Nash equilibria of generative adversarial networks. In *ICML* (2019), pp. 2810–2819. 98

[35] Huang, X., Li, Y., Poursaeed, O., Hopcroft, J., and Belongie, S. Stacked generative adversarial networks. In *CVPR* (2017), pp. 5077–5086. 15, 84

[36] Jacot, A., Gabriel, F., and Hongler, C. Neural tangent kernel: Convergence and generalization in neural networks. *Advances in neural information processing systems 31* (2018). 126

[37] Jain, M., Korzhyk, D., Vaněk, O., Conitzer, V., Pěchouček, M., and Tambe, M. A double oracle algorithm for zero-sum security games on graphs. In *AAMAS* (2011), pp. 327–334. 6, 22, 74, 79

[38] Kang, U., Papadimitriou, S., Sun, J., and Tong, H. Centralities in large networks: Algorithms and observations. In *Proceedings of the 2011 SIAM International Conference on Data Mining* (2011), pp. 119–130. 62

[39] Karras, T., Laine, S., and Aila, T. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2019), pp. 4401–4410. 7, 97

[40] Karypis, G., and Kumar, V. Multi level k-way partitioning scheme for irregular graphs. *Journal of Parallel and Distributed Computing 48*, 1 (1998), 96–129. 43

[41] Kempe, D., Kleinberg, J., and Tardos, É. Maximizing the spread of influence through a social network. In *KDD* (2003), pp. 137–146. 13

[42] Kennedy, J. Particle swarm optimization. *Encyclopedia of machine learning* (2010), 760–766. 58

[43] Khan, S., and Khan, R. A. Chronic stress leads to anxiety and depression. *ANN Psychiatry Ment Health 5*, 1 (2017), 1091. 11

[44] Kornblith, S., Norouzi, M., Lee, H., and Hinton, G. Similarity of neural network representations revisited. In *International Conference on Machine Learning* (2019), PMLR, pp. 3519–3529. 117

[45] Kotyan, S., and Vargas, D. V. Towards evolving robust neural architectures to defend from adversarial attacks. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference Companion* (2020), pp. 135–136. 17

[46] Krizhevsky, A., Nair, V., and Hinton, G. CIFAR-10 (Canadian Institute for Advanced Research). https://www.cs.toronto.edu/ kriz/cifar.html, 2009. 89, 113

[47] Kumari, N., Zhang, R., Shechtman, E., and Zhu, J.-Y. Ensembling off-the-shelf models for gan training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2022), pp. 10651–10662. 126

[48] Lanctot, M., Zambaldi, V., Gruslys, A., Lazaridou, A., Tuyls, K., Pérolat, J., Silver, D., and Graepel, T. A unified game-theoretic approach to multiagent reinforcement learning. In *NeurIPS* (2017), pp. 4190–4203. x, 6, 22, 74, 75, 80

[49] Le, Y., and Yang, X. Tiny ImageNet visual recognition challenge. *CS 231N 7*, 7 (2015), 3. 113

[50] LeCun, Y., and Cortes, C. MNIST handwritten digit database. http://yann.lecun.com/exdb/mnist/, 2010. 89

[51] Li, B., Tang, H., Zheng, Y., Hao, J., Li, P., Wang, Z., Meng, Z., and Wang, L. Hyar: Addressing discrete-continuous action reinforcement learning via hybrid action representation. *arXiv preprint arXiv:2109.05490* (2021). 128

[52] Li, H., Zhang, J., Yang, L., Qi, J., and Gao, Z. Graph-gan: A spatial-temporal neural network for short-term passenger flow prediction in urban rail transit systems. *arXiv preprint arXiv:2202.06727* (2022). 127

[53] Li, J., Fu, X., Sun, Q., Ji, C., Tan, J., Wu, J., and Peng, H. Curvature graph generative adversarial networks. In *Proceedings of the ACM Web Conference 2022* (2022), pp. 1528–1537. 127

[54] Li, Y., Zhang, R., Lu, J., and Shechtman, E. Few-shot image generation with elastic weight consolidation. In *NeurIPS* (2020). 83, 84, 85

[55] Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971* (2015). 52

[56] Littman, M. L., Cassandra, A. R., and Kaelbling, L. P. Learning policies for partially observable environments: Scaling up. In *ICML* (1995), pp. 362–370. 18

[57] Liu, H., Simonyan, K., and Yang, Y. DARTS: Differentiable architecture search. In *ICLR* (2018). 17, 109, 111

[58] Liu, M.-Y., Breuel, T., and Kautz, J. Unsupervised image-to-image translation networks. In *NeurIPS* (2017), pp. 700–708. 2, 72

[59] Liu, Z., Luo, P., Wang, X., and Tang, X. Deep learning face attributes in the wild. In *ICCV* (2015), pp. 3730–3738. 89

[60] Lucassen, P. J., Heine, V. M., Muller, M. B., van der Beek, E. M., Wiegant, V. M., Ron De Kloet, E., Joels, M., Fuchs, E., Swaab, D. F., and Czeh, B. Stress, depression and hippocampal apoptosis. *CNS & Neurological Disorders-Drug Targets (Formerly Current Drug Targets-CNS & Neurological Disorders) 5*, 5 (2006), 531–546. 11

[61] Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083* (2017). 7, 16, 97, 111, 118

[62] McMahan, H. B., Gordon, G. J., and Blum, A. Planning in the presence of cost functions controlled by an adversary. In *ICML* (2003), pp. 536–543. 6, 22, 73

[63] Mescheder, L., Nowozin, S., and Geiger, A. The numerics of GANs. In *NeurIPS* (2017), pp. 1825–1835. 3, 5, 15, 72

[64] Metz, L., Poole, B., Pfau, D., and Sohl-Dickstein, J. Unrolled generative adversarial networks. In *ICLR* (2017). 86

[65] Mirza, M., and Osindero, S. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784* (2014). 2, 72

[66] Miyato, T., Kataoka, T., Koyama, M., and Yoshida, Y. Spectral normalization for generative adversarial networks. *ICLR* (2018). 14, 84

[67] Miyato, T., Maeda, S.-i., Koyama, M., and Ishii, S. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence 41*, 8 (2018), 1979–1993. 2, 72

[68] Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. Playing Atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602* (2013). 60

[69] Mok, J., Na, B., Choe, H., and Yoon, S. Advrush: Searching for adversarially robust neural architectures. In *ICCV* (2021), pp. 12322–12332. 7, 17, 98, 109, 111, 118, 119, 120

[70] Mok, J., Na, B., Kim, J.-H., Han, D., and Yoon, S. Demystifying the neural tangent kernel from a practical perspective: Can it be trusted for neural architecture search without training? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2022), pp. 11861–11870. 126

[71] Muller, P., Omidshafiei, S., Rowland, M., Tuyls, K., Perolat, J., Liu, S., Hennes, D., Marris, L., Lanctot, M., Hughes, E., et al. A generalized training approach for multiagent learning. In *ICLR* (2020). 22, 80

[72] Nash, J. Non-cooperative games. *Annals of mathematics* (1951), 286–295. 21

[73] Pan, F., Zhang, T., Luo, L., He, J., and Liu, S. Learn continuously, act discretely: Hybrid action-space reinforcement learning for optimal execution. *arXiv preprint arXiv:2207.11152* (2022). 127, 128

[74] Pedersen, M. E. H. Good parameters for particle swarm optimization. *Hvass Lab., Copenhagen, Denmark, Tech. Rep. HL1001* (2010), 1551–3203. 63

[75] Pu, Y., Gan, Z., Henao, R., Yuan, X., Li, C., Stevens, A., and Carin, L. Variational autoencoder for deep learning of images, labels and captions. In *NeurIPS* (2016), pp. 2352–2360. 2, 72

[76] Puterman, M. L. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, 2014. 31

[77] Radford, A., Metz, L., and Chintala, S. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434* (2015). 2, 14, 72, 84

[78] Rafferty, A. E., and Griffin, M. A. Perceptions of organizational change: A stress and coping perspective. *Journal of Applied Psychology 91*, 5 (2006), 1154. 12, 24, 29

[79] Ragab, M., Chen, Z., Wu, M., Foo, C. S., Kwoh, C. K., Yan, R., and Li, X. Contrastive adversarial domain adaptation for machine remaining useful life prediction. *IEEE Transactions on Industrial Informatics 17*, 8 (2020), 5239–5249. 2, 72

[80] Ragab, M., Chen, Z., Wu, M., Li, H., Kwoh, C.-K., Yan, R., and Li, X. Adversarial multiple-target domain adaptation for fault classification. *IEEE Transactions on Instrumentation and Measurement 70* (2020), 1–11. 2, 72

[81] Reed, S., Akata, Z., Yan, X., Logeswaran, L., Schiele, B., and Lee, H. Generative adversarial text to image synthesis. In *ICML* (2016). 2, 72

[82] Rezaei, S. S. C., Han, F. X., Niu, D., Salameh, M., Mills, K., Lian, S., Lu, W., and Jui, S. Generative adversarial neural architecture search. *arXiv preprint arXiv:2105.09356* (2021). 98

[83] Rice, E., Tulbert, E., Cederbaum, J., Barman Adhikari, A., and Milburn, N. G. Mobilizing homeless youth for HIV prevention: A social network analysis of the acceptability of a face-to-face and online social networking intervention. *Health Education Research 27*, 2 (2012), 226–236. 27, 29

[84] Rong, J., Qin, T., An, B., and Liu, T.-Y. Revenue maximization for finitely repeated ad auctions. In *Thirty-First AAAI Conference on Artificial Intelligence* (2017). 1

[85] Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., and Chen, X. Improved techniques for training GANs. In *NeurIPS* (2016), pp. 2234–2242. 91

[86] Schrijver, A. *Theory of Linear and Integer Programming.* John Wiley & Sons, 1998. 22, 79

[87] Seff, A., Beatson, A., Suo, D., and Liu, H. Continual learning in generative adversarial nets. *arXiv preprint arXiv:1705.08395* (2017). 84, 85

[88] Seshadhri, C., Kolda, T. G., and Pinar, A. Community structure and scale-free collections of Erdős-Rényi graphs. *Physical Review E 85*, 5 (2012), 056109. 28

[89] Shafahi, A., Najibi, M., Ghiasi, A., Xu, Z., Dickerson, J., Studer, C., Davis, L. S., Taylor, G., and Goldstein, T. Adversarial training for free! *arXiv preprint arXiv:1904.12843* (2019). 17, 105, 111

[90] Silver, D., and Veness, J. Monte-carlo planning in large POMDPs. In *NIPS* (2010), pp. 2164–2172. 18, 33

[91] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. Rethinking the inception architecture for computer vision. In *CVPR* (2016), pp. 2818–2826. 91

[92] Tambe, M. *Security and game theory: algorithms, deployed systems, lessons learned*. Cambridge university press, 2011. 1

[93] Tang, J., Lou, T., and Kleinberg, J. Inferring social ties across heterogenous networks. In *Proceedings of the 5th ACM International*

Conference on Web Search and Data Mining (2012), pp. 743–752. 46, 67

[94] Tsai, J., Nguyen, T. H., and Tambe, M. Security games for controlling contagion. In *AAAI* (2012), pp. 1464–1470. 22

[95] UCLA. The STAND program. `https://depression.semel.ucla.edu/stand_home`, 2018. Accessed: 2019-08-30. 12

[96] Varshney, S., Srijith, P., and Balasubramanian, V. N. Stmgan: Sequentially trained multiple generators for mitigating mode collapse. In *International Conference on Neural Information Processing* (2020), Springer, pp. 676–684. 107

[97] Wang, X., An, B., and Chan, H. Who should pay the cost: A game-theoretic model for government subsidized investments to improve national cybersecurity. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI* (2019). 1

[98] Wang, X., An, B., Strobel, M., and Kong, F. Catching captain jack: Efficient time and space dependent patrols to combat oil-siphoning in international waters. In *Proceedings of the AAAI Conference on Artificial Intelligence* (2018). 1

[99] Wang, Z., She, Q., and Ward, T. E. Generative adversarial networks in computer vision: A survey and taxonomy. *arXiv preprint arXiv:1906.01529* (2019). x, 88, 89

[100] Waugh, K., Bard, N., and Bowling, M. Strategy grafting in extensive games. In *NeurIPS* (2009), pp. 2026–2034. 22

[101] WHO. *The ICD-10 Classification of Mental and Behavioural Disorders: Diagnostic Criteria for Research*, vol. 2. World Health Organization, 1993. 28

[102] Wilder, B., Yadav, A., Immorlica, N., Rice, E., and Tambe, M. Uncharted but not uninfluenced: Influence maximization with an uncertain network. In *AAMAS* (2017), pp. 1305–1313. 13, 25, 45, 61

[103] Xu, P., and Karamouzas, I. Particle-based adaptive discretization for continuous control using deep reinforcement learning. *arXiv preprint arXiv:2003.06959* (2020). 52

[104] Xu, T., Wenliang, L. K., Munn, M., and Acciaio, B. Cot-gan: Generating sequential data via causal optimal transport. *Advances in Neural Information Processing Systems 33* (2020), 8798–8809. 127

[105] Yadav, A., Chan, H., Xin Jiang, A., Xu, H., Rice, E., and Tambe, M. Using social networks to aid homeless shelters: Dynamic

influence maximization under uncertainty. In *AAMAS* (2016), pp. 740–748. 1, 13, 25, 45, 61, 62

[106] Yadav, A., Marcolino, L. S., Rice, E., Petering, R., Winetrobe, H., Rhoades, H., Tambe, M., and Carmichael, H. Preventing HIV spread in homeless populations using PSINET. In *AAAI* (2015), pp. 4006–4011. 13, 25, 45, 61

[107] Yadav, A., Noothigattu, R., Rice, E., Onasch-Vera, L., Soriano Marcolino, L., and Tambe, M. Please be an influencer?: Contingency-aware influence maximization. In *AAMAS* (2018), pp. 1423–1431. 25

[108] Ye, N., Somani, A., Hsu, D., and Lee, W. S. DESPOT: Online POMDP planning with regularization. *JAIR 58* (2017), 231–266. 18, 33

[109] Yoon, J., Jarrett, D., and Van der Schaar, M. Time-series generative adversarial networks. *Advances in neural information processing systems 32* (2019). 127

[110] Zachary, W. W. An information flow model for conflict and fission in small groups. *Journal of Anthropological Research 33*, 4 (1977), 452–473. 46, 67

[111] Zhu, P., Li, X., Poupart, P., and Miao, G. On improving deep reinforcement learning for POMDPs. *arXiv preprint arXiv:1704.07978* (2017). 19, 62