Imitation Learning for Combinatorial Optimization and Contact Tracing

Prateek Gupta

Exeter College

University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy

Hillary 2023

Acknowledgements

It is with humility and a sense of awe I write this section, reflecting on the kindness and compassion of many individuals without whom achieving this milestone would not have been possible.

First and foremost, I extend my heartfelt gratitude to my **supervisors**, Andrea, Pawan, and Yoshua[1] for their unwavering support and guidance throughout my doctoral journey. I have had the good fortune of experiencing three distinct styles of research and supervision, all of which have profoundly inspired me to grow as a researcher and collaborator. I thank Pawan for patiently teaching me the art of researching by simplifying intricate problems into manageable chunks, and grounding solutions in simple and workable steps. Thanks to Andrea's guidance, I was able to ground my research in practicalities. Most importantly, I want to express my profound gratitude to Yoshua for numerous invaluable discussions on deep learning, both related to the solutions to my research problems or the societal problems. I am incredibly grateful to Yoshua for giving me the opportunity to work on projects aimed at bringing about social good, which has been a long-standing passion of mine. I also want to thank Julie, Yoshua's assistant, for guiding me on how to make the best use of time with Yoshua.

Kudos to my **colleagues** for making the research journey enjoyable! Oxford crew, you rock! Thanks, Adel, Alban, Alessandro, Alisdair, Florian, Jody, Leo, Pankaj, Puneet, and Rudy. Special thanks to Adel, Leo, and Puneet for their amazing career advice. Andrea's lab was a wilde ride, thanks to Elias, Maxime, Didiér, Antoine, Giulia, Federico, and Claudio. Elias, Maxime, and Didiér – Thanks for all the great discussions and helping with my first publication. Yoshua's lab was a blast with Andrew, David, Karsten, Maksym, Tianyu, Nasim, Pierre-Luc St. Charles, Victor, Tristan, and Yang. Yang – your inspiring words on big thinking and impactful research will always stay with me. Special thanks to Andrew, an amazing collaborator who joined hands with me on our efforts to do good using our skills. Lastly, a big thanks to the entire COVI team for making research meaningful and fun. Andrew, Hannah, Nanor, Soren, Joanna, Martin, Tegan, Nasim, and Yang – thank you for sticking it out until the end. The invaluable experience of teaching

[1]Names are sorted alphabetically.

and creating deep learning courses has helped me to be a better researcher. I want to thank Kieran for helping me with such opportunities.

No words will ever be enough to express my gratitude to my **family** for their unwavering emotional support. Yet, I would like to take this opportunity to acknowledge and thank my Mom, Dad, and Big brother for their constant support and encouragement that has been the rock-solid foundation of my journey. My cousins, Akanksha, Anupam, Apoorva, Deepu, Nandini, Nikita, Noora, Rajat, Rakhi, and Rashmi, have been the cherry on top of the cake, always ready with their witty and lighthearted humor that has made this journey much more enjoyable. Deepu bhai, this is the moment when the tail will also pass - thank you for being there for me. Special thanks to Anupam for being an inspiration to connect with others and make the most of my doctorate time. Lastly, my adorable nieces have been a constant source of joy and cuteness. Thank you, Ahana, Arshiya, Aru, Inaisha, Kanha, Kuhu, and Suhani, for making my journey brighter with your sweet smiles and innocent laughter. A big thanks to Kaushal Uncle, my enthusiastic cheerleader, and Girish Uncle, who planted the seeds of higher studies in me.

Shout out to my amazing **friends**, Adam, Alessandro, Alessio, Andreas, Arunesh, Eugenia, Giulia, Jan, Johann, Julien, Luisa, Nikhita, Nikolas, Pascal, Sofia, Solene, and Solveig. I am incredibly grateful for the unwavering support of Akshit, Ashwin, Nazuk, and Umair. Thank you all for filling my life with so many unforgettable moments and support through this journey. Finally, a big thanks to Prateek Bansal, who has always provided invaluable guidance while also being a great friend.

Finally, I would like to thank the **institutes**, The Alan Turing Institute for funding my doctorate studies, Exeter College for providing an enriching environment for students, and Mila for hosting me as a visiting researcher.

Lastly, I must acknowledge the role of **luck** in my journey, and I choose to do so by thanking Sri Krishna – Hare Krishna, Hare Krishna, Krishna, Krishna, Hare Hare!

Abstract

The field of Imitation Learning (IL) has seen significant progress in recent years as researchers have applied this machine learning technique to various domains such as robotics, self-driving cars, healthcare, and game playing. Each domain has contributed to the advancement of the field by developing and applying new methods to solve the unique problems specific to their domain. In this thesis, we focus on IL in two domains that pose their own unique challenges.

The first application involves learning to imitate a highly accurate heuristic for mixed-integer linear programming (MILP) solvers, which although precise, is not practical due to its computational inefficiency. The second application involves the development of an IL framework to accurately predict the infectiousness of individuals through a smartphone application utilizing the newly developed Proactive Contact Tracing (PCT) framework, which overcomes the limitations of conventional contact tracing methods.

We design our IL frameworks based on the dynamics of a manageable environment (e.g., simulator), with the goal of transferring the learned models to larger, unseen environments. The development of these frameworks requires the consideration and resolution of several challenges. These challenges include incorporating domain-specific inductive biases, ensuring the robustness of models against distribution shifts, and designing models that are lightweight and suitable for deployment. By addressing these challenges, we hope to contribute not only to the advancement of IL, but also to the domains in which it is applied, bringing new and improved solutions to these respective fields.

Specifically, to imitate the expert heuristic of MILP solvers, we identified and addressed two key shortcomings of the existing IL framework. First, the proposed Graph Neural Networks (GNNs) are computationally expensive but highly accurate and their runtime performance degrades in the absence of GPUs. This setting may arise since MILP solvers are CPU–only. To address this, we proposed novel architectures that trade-off the expressivity of GNNs with inexpensive computations of multi-linear perceptrons, along with training protocols that make the models robust to distribution shifts. The models trained using these techniques resulted in up to 26% improvement in runtime. The second issue is the inability to capture the dependence between observations to train GNNs. Our research revealed a

“lookback” phenomenon that occurs frequently in the expert heuristic, where the best decision at the child node is often the second-best at the parent node. To incorporate this phenomenon in the loss function, we proposed a new loss function that imitates this heuristic more accurately, resulting in models with up to 15% improvement in running time.

Finally, during the COVID-19 pandemic, nations around the world faced a dilemma of whether to open up the economy or prioritize saving lives. In response, digital contact tracing applications emerged. However, to avoid violating user privacy, most apps relied on a quarantine-or-not interface with limited intelligence on the level of risk of the notification recipient. This approach led to alert fatigue, making users less likely to follow recommendations. To overcome these issues while maintaining user privacy and sophisticated risk estimation models, we proposed the Proactive Contact Tracing (PCT) framework. Our framework repurposes user communication to carry information about estimated risk in "risk messages". These messages, along with personal information (e.g., medical history or symptoms), are used in a risk estimation model to output risk messages sent to other users. Depending on estimated risk, graded notifications (e.g., exercise caution or avoid unnecessary behavior) are shown to the users. Using an agent-based model (ABM) and a simple interpretable rule-based model, we demonstrated that the rule-based PCT has a better economic-public health trade-off than the existing apps.

In follow-up work, we turned to deep learning to design a risk estimation model. While reinforcement learning would have been ideal, the computationally expensive ABM precludes its use. Instead, we employed an imitation learning framework to train deep learning models, specifically, we proposed several variants of set transformers. We also used domain randomization, collecting observations using several random instantiations of the ABM, to ensure that models were robust to assumptions baked into the ABM. Furthermore, we used iterative training to ensure the models remained robust to auto-induced distribution shifts. Overall, we showed that a deep learning-based PCT outperforms rule-based PCT. To finalize our proposal, we suggest an iterative procedure for app deployment and ABM calibration to bridge the gap from the ABM to real-world deployment.

Contents

List of Figures

1 Introduction

Contents

1.1 Preamble

The ongoing research and advances in the field of Imitation Learning (IL) have been possible through the focused efforts on various real-world applications, such as robotics, self-driving cars, healthcare, and game playing. IL involves learning from expert demonstrations to improve the decision-making process in these applications.

Two popular methods in IL are Behavioral Cloning (BC) and Direct Policy Learning (DPL). BC aims to find a parametrized function trained to minimize a loss function defined on data collected from expert demonstrations, while DPL learns such a function by continually extending the training data to the expert-labeled observations collected using the learned model.

In this thesis, we focus on IL applications in combinatorial optimisation and contact tracing, where the constraints imposed by the problem or the deployment context pose challenges. Our proposed solutions will span across BC and DPL.

First, we investigate the extent to which mixed-integer linear programming (MILP) solvers can benefit from replacing manually designed heuristics with neural networks learned by imitating an expert heuristic. This work will make use of the BC framework to train the neural network.

Second, we develop a novel contact tracing framework, Proactive Contact Tracing (PCT), to generate *early warning signals* of viral infection to achieve a superior trade-off between population mobility restriction and epidemic control. To develop a more accurate neural network based risk estimation model for PCT, we further propose an IL framework using DPL to imitate the ground truth available through a simulator.

1.2 Imitation Learning for Combinatorial Optimization

Combinatorial optimization (CO) involves finding the optimal solution for constrained problems that involve at least one integer variable. Suppose the objective function and the constraints of the problem are linear. In that case, it is also termed as a mixed-integer linear program, which occurs in various real-world applications

such as fleet management in delivery services or airline revenue management, just to mention a few (Papadimitriou and Steiglitz, 1998).

The presence of integer-valued variables makes the problem NP-Hard, i.e., to find the optimal solution, one must search over a vast space of solutions, which can be highly time consuming. However, depending on the problem formulation, the search space can have sufficient structure to lend the problem a polynomial time algorithm. For example, Dijkstra's algorithm (Dijkstra, 1959) to find the shortest path between any two nodes in a graph has a polynomial time complexity. In the absence of such a structure, solving a general MILP requires clever implicit enumeration techniques such as the branch-and-bound (B&B) framework, which involves breaking the problem into subproblems and obtaining bounds on them to facilitate exploration of the space. The bounds on the subproblems are obtained by solving a more manageable problem by relaxing the integral constraints on variables; this is also called the LP-relaxation of MILP because the problem resulting from the relaxation of the integer requirements is a linear program (LP). One can imagine a tree-like structure consisting of subproblems as nodes from a solving procedure using the B&B framework. To enable efficient exploration of the search space, the B&B framework requires numerous decisions (Lodi and Zarpellon, 2017).

MILP solvers use the B&B framework along with several heuristics for such decision-making. These heuristics have been designed by experts using their experience over the decades. One of the most crucial decisions in B&B is the selection of the *branching variable*, which is an integer-constrained variable that takes a non-integral value in the LP relaxation of a subproblem.

Over the last decade, researchers have designed variable selection policies empirically shown to be better than manually-designed heuristics. The majority of the approaches have focused on imitating a gold-standard *strong branching* (SB) heuristic that has been empirically shown to exhibit the best iterative performance. However, the strong branching heuristic is computationally expensive to be of any practical use. Thus, most of the machine learning research for MILP-solving has been to replace the heuristic with computationally inexpensive functions that can

imitate the strong branching decisions sufficiently well. As will be discussed in the chapters 2 and 3, such functions are learned based on the data collected on the smaller instances by running the SB heuristic, but their performance is measured on the scaled-up instances where deploying SB would be infeasible.

Alvarez et al. (2017) learn to imitate the strong branching heuristic using extremely randomized trees, while Khalil et al. (2016) used support vector machines to design a fast discriminative classifier. More recently, Gasse et al. (2019) proposed GNNs to learn neural network-based branching policies. GNNs have been shown to outperform the existing learned policies trained to imitate the *strong branching* heuristic.

This thesis addresses two limitations of the state-of-the-art imitation learning framework to imitate the SB heuristic. First, in Chapter 2, we identify that GNNs are computationally demanding and need expensive GPUs to make the learned branching policies competitive with the existing baselines. However, the MILP solvers are entirely CPU-based, and the GNN-based policies might not be suitable when (a) practitioners may not have GPUs or (b) when multiple MILPs need to be solved in parallel, thereby requiring multiple GPUs to retain the performance of MILP solvers. Thus, to overcome GNN's shortcoming, we design a neural network architecture and the corresponding training protocol to restore the GNN's benefits on CPU-only machines.

Second, in Chapter 3, we note the dependencies between observations collected using the SB heuristic. Specifically, we find that often parent nodes' second best SB choice is the child nodes' best SB choice. We call this a *lookback phenomenon*. Thus, we propose modifications to the loss function to incorporate this inductive bias in the IL framework. In addition to the modifications above, we further propose a formalism to standardize the evaluation of learned policies for MILP solving. The modified imitation framework empirically outperforms the state of the art (SOTA) framework.

1.3 Imitation Learning for Contact Tracing

The emergence of the SARS-CoV-2 virus in 2019 brought a critical dilemma for governments worldwide. For the first time, the public authorities faced the challenge of balancing public health and economic welfare. Nationwide lockdowns were implemented to curb the spread of the virus, saving countless lives. However, this came at a considerable cost to the economy. Governments were thus forced to seek innovative solutions to address this issue.

One of the most apparent solutions was contact tracing. Contact tracing involves identifying and tracing the past contacts of an individual who has tested positive for the virus. This has traditionally been done manually by public health experts (PHE), who would contact individuals involved in past encounters via telephone or other means. However, the long incubation period of the SARS-CoV-2 virus, which can be up to five days, made the manual process of contact tracing slow and inefficient. Therefore, there was a need for a faster and more accurate tracing process.

Digital Contact Tracing (DCT) applications emerged as an innovative solution to address the shortcomings of the manual contact tracing process. DCT apps use Bluetooth Low Energy (BLE) devices installed on smartphones to exchange private keys with nearby contacts. These keys enable sending a notification to past contacts later when a smartphone user inputs a positive test result in the user's own app. However, the process of developing such applications was met with various design constraints. The main concerns were around the privacy implications for the app user and the behavioral implications for the user.

The existing versions of DCT apps were designed to use minimal information about the user in estimating the risk of infecting their past contacts. Commonly deployed worldwide, these apps notify contacts to quarantine when appropriate. However, the binary nature of these recommendations asks more individuals than are infected to quarantine, leading to reduced compliance by users. This phenomenon has been mockingly referred to as the "pingdemic," wherein DCT apps ping or notify many uninfected smartphone users.

To address these issues, we proposed a Proactive Contact Tracing framework in Chapter 4. The PCT framework generates early-warning signals of infection, allowing users to exercise varying degrees of caution depending on their risk estimates. For example, a smartphone user that is predicted to have an intermediate likelihood of infection will be asked to "avoid unnecessary contacts". By using intermediate levels to alert users, the PCT framework provides personalized recommendations, rather than sending the same extreme advice to all users.

Users input their data into the PCT framework, including preexisting conditions, daily symptoms, and COVID-19 test results. A risk estimation model uses this information, along with "risk messages" from past contacts, to predict users' likelihood of infection within a certain timeframe. The estimation model then sends the likelihood of infection on the day of the encounter, as a "risk message", to the user's contacts on that day. The likelihood of infection is then converted into a behavioral recommendation asking the user to exercise caution.

We propose a simple rule-based risk estimation model that improves upon existing Digital Contact Tracing apps, providing a superior trade-off between mobility restrictions and the spread of the viral infection. To demonstrate the effectiveness of the framework, we developed an agent-based model simulating the spread of infection and movement of individuals carrying smartphones in a hypothetical city based on the demographics of the City of Montréal, Québec.

In Chapter 5, we propose an imitation learning framework to train neural network-based risk models to improve the accuracy of infection predictions. These models are trained to imitate the ground-truth viral load based on the information sources available in the PCT-based apps. However, deploying these models in the real world may result in a distribution shift, a well-known challenge in robotics, where models need to adapt to the real world. We discuss the challenges in the sim-to-real transfer of the imitation learning framework for PCT and propose solutions to address them.

1.4 Thesis Outline & Contributions

As mentioned previously, this thesis investigates the two applications of imitation learning. Chapter 2 and 3 of this thesis investigate the gaps in imitating an expert heuristic for MILP solvers, while chapter 4 proposes a novel contact tracing framework, we call Proactive Contact Tracing, and chapter 5 proposes a imitation learning framework for PCT to enable superior risk modeling.

In what follows, we detail the content and contribution of each chapter. We point out that this thesis follows the integrated thesis format, and as such is a collection of papers. Each of the following chapters correspond to a publication.

1.4.1 Hybrid Models for Learning to Branch

Corresponding publication. Gupta et al. (2020a), published at NeurIPS 2020.

Summary. GNNs, proposed by Gasse et al. (2019), to imitate a gold-standard branching heuristic, lose their runtime performance when they are deployed on CPU-only machines. Such scenarios arise when the practitioner does not have a GPU (MILP solvers are entirely CPU-based) or when lots of MILPs are to be solved in parallel, thus making CPU deployment as a cost-effective route. Thus, in order to retain the expressivity of GNNs while making it cheaper to evaluate them, we propose hybrid models for learning to branch. To bring them at par with the GPU performance of GNNs, we further propose training protocols such as depth-based loss function penalty, knowledge distillation to use GNNs output as the target, and auxiliary tasks. We empirically demonstrate that the hybrid models outperform GNNs deployed on CPUs and bring the performance closer to the GNNs deployed on GPUs.

Broader Impact. This publication establishes a bridge between the work done in ML in the last years on learning to branch and the traditional MILP solvers used to routinely solve thousands of optimization applications in energy, telecommunications, logistics, biology, just to mention a few.

The MILP solvers are executed on CPU-only machines and the technical challenge of using GPU-based algorithmic techniques to hybridize them had been neglected thus far. Admittedly, such a challenge was not urgent when the learning to branch literature was in its early stages. That situation has changed drastically with the GNN implementation in Gasse et al. (2019), the first approach to show significant benefit with respect to the default version of a state-of-the-art MILP solver like SCIP. For this reason, this publication comes at the due time for the literature in the field and addresses the challenge, for the first time, in a sophisticated, yet relatively simple way.

Thus, our work provides the first viable way for commercial and noncommercial MILP solver developers to implement and integrate an ML-based "learning to branch" framework and for hundreds of thousands of users and practitioners to use it. In an even broader sense, the fact that we were able to approximate the performance of GPU-based models with a sophisticated integration of CPU-based techniques is consistent with, for example, Hinton et al. (2015), and widens the space of problems to which machine learning (ML) techniques can be successfully applied.

1.4.2 Lookback for Learning to Branch

Corresponding publication. Gupta et al. (2022), published at TMLR 2022.

Summary. The imitation learning framework proposed by Gasse et al. (2019) minimizes a standard cross-entropy loss between the output of GNNs and a one-hot encoded target. In this work, we ask whether the framework incorporates appropriate inductive biases. To investigate this line of questioning, we looked at the dependencies between various nodes in a B&B tree. We discovered a characteristic property of the strong branching heuristic, which we call *lookback* phenomenon. Specifically, we found that often the second-best decision at the parent node is the best decision at the child node. Thus, in this work, we propose modifications to the loss function to account for the lookback phenomenon. We show empirically that the GNNs trained with these inductive biases have a superior performance.

Broader Impact. This paper continues the exploration of the use of machine learning techniques for the most critical step in branch-and-bound methods, i.e., variable selection. Branch and bound is the method of choice for solving a myriad of discrete optimization problems appearing in all sort of applications and it is the basic scheme of all commercial and open-source MILP solvers. Thus, the impact of the research in the area is potentially very high, not only from a methodological perspective but also in terms of day-to-day challenges that we all face, including drug discoveries and climate change.

This paper significantly advances the research in the area by observing for the first time a hidden pattern, the lookback phenomenon, in the statistical evolution of the most successful heuristic for variable selection. The paper proposes several methods to exploit such a phenomenon and makes a significant step forward on the use of ML for discrete optimization.

1.4.3 Proactive Contact Tracing

Corresponding publication. Gupta et al. (2023), published at PLOS Digital Health.

Summary. In late 2019, the outbreak of the SARS-CoV-2 virus presented governments worldwide with the difficult challenge of balancing economic growth with public health. One response to this challenge is contact tracing, which involves informing the contacts of infected individuals to quarantine, thereby limiting the virus' spread while allowing economic activity. However, overwhelmed public health authorities and the virus' characteristics made it necessary to establish a faster and more scalable contact tracing process. This need led to the emergence of Digital Contact Tracing apps. However, the most basic versions of these apps only recommend quarantine to users if their past contacts have tested positive for the infection, ignoring rich information sources that can enable more sophisticated risk modeling to categorize users into different risk levels. To address these limitations, we propose the Proactive Contact Tracing framework, which incorporates rich information sources while preserving user privacy, enabling a variety of behavioral

recommendations based on sophisticated risk modeling. In this work, we devise a simple rule-based risk estimation model and demonstrate its superior ability to balance economic and health considerations through extensive sensitivity analysis.

Broader Impact. This project was born out of the need to address the COVID-19 pandemic. Initially, it was not intended to be a publication exercise while it was being considered by the Québec government. However, as it became clear that the government was not going to adopt our framework, called as COVI, we felt it was in our best interest to formally publish the project. This way, it can serve as inspiration for others to undertake similar initiatives in the future.

During the pandemic, we saw rapid innovation in the field of contact tracing and risk modeling. Our approach has been quite different from the rest of the proposals. Thus, our work brought out a novel perspective on dealing with the problems in designing the contact tracing frameworks.

Through this process, we were also able to establish a multi-disciplinary collaboration among researchers in epidemiology, economists, and computer scientists. This is an invaluable experience as it exposes the contributors to collective strength in addressing complex societal problems. For example, the follow-up works have involved using artificial intelligence (AI) to address critical issues in climate change (see section 1.5).

1.4.4 Predicting Infectiousness for Proactive Contact Tracing

Corresponding publication. Bengio et al. (2021), published at ICLR 2021

Summary. The PCT framework proposed in Gupta et al. (2023) relies on a risk estimation model. While a simple rule-based predictor was found to be more effective than existing methods, we take a step further by proposing a sophisticated neural network-based risk model. To achieve this, we introduce an imitation learning framework to train these models to predict viral load characteristics and their evolution in an infected individual. However, the deployment of such a model is complicated by the sim-to-real transfer problem, which arises when the model

faces observations that were not seen during training upon deployment in the real world. We suggest various training protocols to overcome this issue. Our results demonstrate that deep learning-based PCT models outperform rule-based PCT models, highlighting the potential of this approach.

Broader Impact. Our projects aimed at saving more lives with less restriction than traditional manual and digital contact tracing methods, which can be limited in their effectiveness. By providing graded risk warnings of possible infection, even before a person shows symptoms, we can limit the spread of the virus and ultimately save lives. However, this potential benefit must be carefully balanced against potential costs, including privacy and ethical concerns, which cannot be ignored.

To address these concerns, we contributed to a comprehensive paper (Bengio et al., 2020) on the social impact of our technology, which explored issues related to privacy, dignity, and democracy. Additionally, we established a non-profit organization, COVI, to manage the project independently of governments and corporations. This organization included a diverse board of experts in public health, ethics, and machine learning, as well as representatives of civil society and government.

We recognize that there may be concerns about the potential for our technology to favor certain groups, such as wealthier individuals who are more likely to own smartphones. To address this issue, we propose using census data to monitor and mitigate any disparities in prediction error on a per-neighborhood basis. By reweighing or oversampling examples from disadvantaged communities, we can reduce any potential bias and ensure that our technology benefits everyone equally.

Overall, our projects represent an important step forward in the fight against the pandemic. By leveraging the power of machine learning and mobile technology, we can save lives and limit the spread of the virus while minimizing the impact on people's lives and liberties.

1.5 Other works

Over the course of my doctorate, I have contributed to works that are not part of this thesis. The ideas in my thesis are invariably borrowed or transferred to these works. This section is a list of those works.

Deep Metric Learning: Revisiting Training & Generalization Roth et al. (2020), published at ICML 2020.

Deep Metric Learning (DML) aims to learn the visual similarities. At the time of this research, there were numerous studies without the consensus on evaluation protocols, making them uncomparable. In this work, we ran an extensive experimentation to bring all the DML methods to standard benchmarks. In doing so, we recognized an opportunity to improve the generalization performance of the DML models through a regularization procedure. My contribution to this work was mainly in the design of coreset algorithms for mini-batch selection such that the mini-batch gradients are representative of the unbiased gradients.

AI for Global Climate Cooperation: Modeling Global Climate Negotations, Agreements, and Long-Term Cooperation in RICE-N Zhang et al. (2022b), published at AAAI Climate Change Symposium 2022.

As of March 2023, this project has been named as one of the **top 10 most innovative projects** by Netexplo in 2022. The final results will be announced in April 2023.

Throughout the years, one of the most persistent observations in the annual Conference of Parties (COP) has been the failure of nations to deliver on their promises. This observation raises an important and pressing question: Can artificial intelligence design negotiation protocols that enable nations to uphold their commitments?

In response to this question, we organized a global competition that called for researchers across diverse fields ranging from economics, climate science, AI, and policy design. The competition aimed to provide a solution to the pressing question

of how AI can help facilitate the implementation of sustainable commitments made by nations.

The competition involved providing the participants with an agent-based model (ABM) that features nations as software agents. These nations interact with each other through regulations on trade and tariffs, as they aim to increase their country's welfare while considering the global damage (i.e., rise in global temperature).

The competition required participants to submit a negotiation protocol that exhibits overall GDP growth while minimizing the rise in global temperature. The participants were tasked with creating negotiation protocols that will foster cooperation among the nations and promote adherence to the commitments made. The competition is still in phase-I, and we intend to publish the results by June.

(Private)-Retroactive Carbon Pricing: A market-based approach for climate finance and risk assessment. Bengio et al. (2022), working paper, currently being prepared as a one-pager policy brief.

Carbon emissions have a significant, long-lasting impact that is often not immediately apparent, such as the displacement of populations in coastal areas, incurring immense costs to taxpayers and causing immeasurable harm to human life. Therefore, it is imperative to hold accountable those who engage in activities that produce carbon emissions and contribute to long-term damage. However, accurately estimating the true cost of carbon emissions has proven challenging due to the complex interplay of environmental and socio-political factors.

To address this issue, we propose a novel mechanism called Retrospective Carbon Pricing. This mechanism involves revising the long-term damage caused by carbon emissions every year, with emitters being liable to pay for the yearly adjustments, which may be refunded if they have overpaid. To hedge against future adjustments, emitters can purchase insurance through a market designed specifically for this purpose. Insurance companies that can develop accurate forecasting models stand to profit, thereby incentivizing innovation in the field of climate modeling.

The Retrospective Carbon Pricing mechanism is designed to provide a stable, long-term approach to address carbon emissions. It prevents fluctuations in the ideology of political organizations, households, and businesses that operate on a shorter time horizon than the time scale at which the effects of carbon emissions are evident.

Our proposal has garnered the attention of Mark Carney, former Governor of the Bank of England and the Bank of Canada, who provided valuable feedback that helped position the idea for specific industries. With his advice in mind, we are currently working on preparing a policy draft that policymakers can use to take action on this critical issue. The implementation of the Retrospective Carbon Pricing mechanism would ensure that those who engage in activities that produce carbon emissions are held accountable for the long-term damage they cause, providing a significant step forward in the fight against climate change.

2

Hybrid Models for Learning to Branch

Contents

About this work

Publication. Gupta et al. (2020a), published in NeurIPS 2020

Peer-reviews. Peer-reviews can be accessed at `https://bit.ly/3XU9E8W`.

Code. Code is made publicly-available at `https://github.com/pg2455/Hybrid-learn2branch`.

Contributions. I contributed to this project in the following ways,

- Identification of the gap between practice and existing research
- Brainstorming the solutions with collaborators
- Coding the solutions and running experiments
- Preparing the figures and tables for the publication
- Writing the draft and managing the submission as well as the peer-reviews

Talks. This work has been presented at the following venues,

- NeurIPS 2020
- Huawei Research ML Seminar, Aug' 21
- G-Research ML Seminar, Jan' 23

Abstract

A Graph Neural Network (GNN) approach for learning to branch has been shown to successfully reduce the running time of branch-and-bound (B&B) algorithms for Mixed Integer Linear Programming (MILP). While the GNN relies on a GPU for inference, MILP solvers are purely CPU-based. This severely limits its application as many practitioners may not have access to high-end GPUs. In this work, we ask two key questions. First, in a more realistic setting where only a CPU is available, is the GNN model still competitive? Second, can we devise an alternate

computationally inexpensive model that retains the predictive power of the GNN architecture? We answer the first question in the negative, and address the second question by proposing a new hybrid architecture for efficient branching on CPU machines. The proposed architecture combines the expressive power of GNNs with computationally inexpensive multi-layer perceptrons (MLP) for branching. We evaluate our methods on four classes of MILP problems, and show that they lead to up to 26% reduction in solver running time compared to state-of-the-art methods without a GPU, while extrapolating to harder problems than it was trained on. The code to replicate the results in this chapter is available at `https://github.com/pg2455/Hybrid-learn2branch`.

2.1 Introduction

Mixed-Integer Linear Programs (MILPs) arise naturally in many decision-making problems such as auction design (Abrache et al., 2007), warehouse planning (Elson, 1972), capital budgeting (Finn, 2005) or scheduling (Floudas and Lin, 2005). Apart from a linear objective function and linear constraints, some decision variables of a MILP are required to take integral values, which makes the problem NP-hard (Papadimitriou and Steiglitz, 1982).

Modern mathematical solvers typically employ the B&B algorithm (Land and Doig, 1960) to solve general MILPs to global optimality. While the worst-case time complexity of B&B is exponential in the size of the problem (Wolsey, 1988), it has proven efficient in practice, leading to wide adoption in various industries. At a high level, B&B adopts a divide-and-conquer approach that consists in recursively partitioning the original problem into a tree of smaller sub-problems, and solving linear relaxations of the sub-problems until an integral solution is found and proven optimal.

Despite its apparent simplicity, there are many practical aspects that must be considered for B&B to perform well (Achterberg, 2007); such decisions will affect the search tree, and ultimately the overall running time. These include several decision problems (Lodi and Zarpellon, 2017) that arise during the execution of

the algorithm, such as *node selection*: which sub-problem do we analyze next?; and *variable selection* (a.k.a. branching): which decision variable must be used (branched on) to partition the current sub-problem? While such decisions are typically made using hard-coded expert heuristics which are implemented in modern solvers, more and more attention is given to statistical learning approaches for replacing and improving upon those heuristics (He et al., 2014b; Alvarez et al., 2017; Khalil et al., 2016; Gasse et al., 2019; Zarpellon et al., 2021). An extensive review of different approaches at the intersection of statistical learning and combinatorial optimization is given in Bengio et al. (2018).

Recently, Gasse et al. (2019) proposed to tackle the variable selection problem in B&B using a Graph Neural Network (GNN) model. The GNN exploits the bipartite graph formulation of MILPs together with a shared parametric representation, thus allowing it to model problems of arbitrary size. Using imitation learning, the model is trained to approximate a very good but computationally expensive "expert" heuristic named *strong branching* (Applegate et al., 1995). The resulting branching strategy is shown to improve upon previously proposed approaches for branching on several MILP problem benchmarks, and is competitive with state-of-the-art B&B solvers. We note that one limitation of this approach, with respect to general B&B heuristics, is that the resulting strategy is only tailored to the class of MILP problems it is trained on. This is very reasonable in our view, as practitioners usually only care about solving very specific problem types at any time.

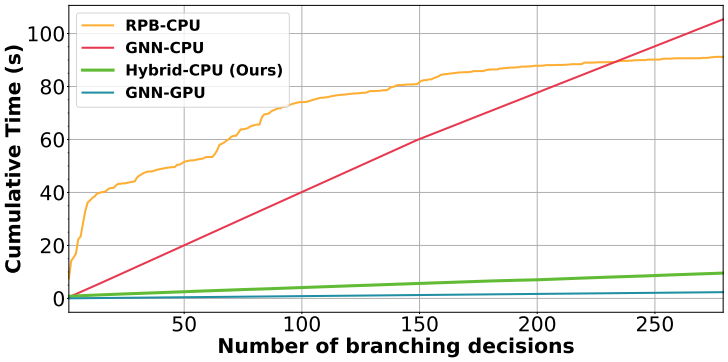

Figure 2.1: Cumulative time cost of different branching policies: (i) the default internal rule RPB of the SCIP solver; (ii) a GNN model (using a GPU or a CPU); and (iii) our hybrid model. Clearly the GNN model requires a GPU for being competitive, while our hybrid model does not. (Measured on a capacitated facility location problem, medium size).

While the GNN model seems particularly suited for learning branching strategies, one drawback is a high computational cost for inference, i.e., choosing the branching variable at each node of the B&B tree. In Gasse et al. (2019), the authors use a high-end GPU card to speed-up the GNN inference time, which is a common practice in deep learning but is somewhat unrealistic for MILP practitioners. Indeed, commercial MILP solvers rely solely on CPUs for computation, and the GNN model from Gasse et al. (2019) is not competitive on CPU-only machines, as illustrated in Figure 2.1. There is indeed a trade-off between the quality of the branching decisions made and the time spent obtaining those decisions. This trade-off is well-known in MILP community (Achterberg, 2007), and has given rise to carefully balanced strategies designed by MILP experts, such as *hybrid branching* (Achterberg and Berthold, 2009), which derive from the computationally expensive *strong branching* heuristic (Applegate et al., 1995).

In this chapter, we study the time-accuracy trade-off in learning to branch with the aim of devising a model that is both computationally inexpensive and accurate for branching. To this end, we propose a hybrid architecture that uses a GNN model only at the root node of the B&B tree and a weak but fast predictor, such as a simple Multi-Layer Perceptron (MLP), at the remaining nodes. In doing so, the weak model is enhanced by high-level structural information extracted at the root node by the GNN model. In addition to this new hybrid architecture, we experiment and evaluate the impact of several variants to our training protocol for learning to branch, including: (i) end-to-end training (Glasmachers, 2017; Bojarski et al., 2016), (ii) knowledge distillation (Hinton et al., 2015), (iii) auxiliary tasks (Liebel and Körner, 2018), and (iv) depth-dependent weighting of the training loss for learning to branch, an idea originally proposed by He et al. (2014b) in the context of node selection.

We evaluate our approach on large-scale MILP instances from four problem families: Capacitated Facility Location, Combinatorial Auctions, Set Covering, and Independent Set. We demonstrate empirically that our combination of hybrid architecture and training protocol results in state-of-the-art performance in the realistic setting of a CPU-restricted machine. While we observe a slight decrease

in the predictive performance of our model with respect to the original GNN from (Gasse et al., 2019), its reduced computational cost still allows for a reduction of up to 26% in overall solving time on all the evaluated benchmarks compared to the default branching strategy of the modern open-source solver SCIP (Gleixner et al., 2018). Also, our hybrid model preserves the ability to extrapolate to harder problems than trained on, as the original GNN model.

2.2 Related Work

Finding a branching strategy that results in the smallest B&B tree for a MILP is at least as hard–and possibly much harder–than solving the MILP with any strategy. Still, small trees can be obtained by using a computationally expensive heuristic named *strong branching* (SB) (Applegate et al., 1995; Linderoth and Savelsbergh, 1999). The majority of the research efforts in variable selection are thus aimed at matching the performance of SB through faster approximations, via cleverly handcrafted heuristics such as *reliability pseudocost branching* (Achterberg et al., 2005), and recently via machine learning[1] (Alvarez et al., 2017; Khalil et al., 2016; Gasse et al., 2019). We refer to Lodi and Zarpellon (2017) for an extensive survey of the topic.

Alvarez et al. (2017) and Khalil et al. (2016) showed that a fast discriminative classifier such as extremely randomized trees (Geurts et al., 2006) or support vector machines (Hearst, 1998) on hand-designed features can be used to mimic SB decisions. Subsequently, Gasse et al. (2019) and Zarpellon et al. (2021) showed the importance of representation learning for branching. Our approach, in some sense, combines the superior representation framework of Gasse et al. (2019) with the computationally cheaper framework of Khalil et al. (2016). Such hybrid architectures have been successfully used in ML problems such as visual reasoning (Perez et al., 2018), style-transfer (Dumoulin et al., 2017), natural language processing (Srivastava et al., 2015; Dhingra et al., 2017), and speech recognition (Kim et al., 2017).

[1]Interestingly, early works on ML methods for branching can be traced back to 2000 (Achterberg, 2007, Acknowledgements).

2.3 Preliminaries

Throughout this chapter, we use boldface for vectors and matrices. A MILP is a mathematical optimization problem that combines a linear objective function, a set of linear constraints, and a mix of continuous and integral decision variables. It can be written as:

$$\underset{\mathbf{x}}{\operatorname{argmin}} \mathbf{c}^\intercal \mathbf{x}, \quad \text{s.t.} \quad \mathbf{A}\mathbf{x} \leq \mathbf{b}, \quad \mathbf{x} \in \mathbb{Z}^p \times \mathbb{R}^{n-p},$$

where $\mathbf{c} \in \mathbb{R}^n$ denotes the cost vector, $\mathbf{A} \in \mathbb{R}^{m \times n}$ the matrix of constraint coefficients, $\mathbf{b} \in \mathbb{R}^m$ is the vector of constant terms of the constraints, and there are p integer variables, $1 \leq p \leq n$.

The B&B algorithm can be described as follows. One first solves the linear program (LP) relaxation of the MILP, obtained by disregarding the integrality constraints on the decision variables. If the LP solution $\mathbf{x}^\star$ satisfies the MILP integrality constraints, or is worse than a known integral solution, then there is no need to proceed further. If not, then one divides the MILP into two sub-MILPs. This is typically done by picking an integral decision variable that has a fractional value, $i \in \mathcal{C} = \{i \mid x_i^\star \notin \mathbb{Z}, i \leq p\}$, and create two sub-MILPs with additional constraints $x_i \leq \lfloor x_i^\star \rfloor$ and $x_i \geq \lceil x_i^\star \rceil$, respectively. The decision variable i that is used to partition the feasible region is called the *branching variable*, while $\mathcal{C}$ denotes the *branching candidates*. The second step is to select one of the leaves of the tree, and repeat the above steps until all leaves have been processed[2].

In this work, we refer to the first node processed by B&B as the *root node*, which contains the original MILP, and all subsequent nodes containing a local MILP as *tree nodes*, whenever the distinction is required. Otherwise we refer to them simply as nodes.

[2]For a more involved description of B&B, the reader is referred to Achterberg (2007).

2.4 Methodology

As mentioned earlier, computationally heavy GNNs can be prohibitively slow when used for branching on CPU-only machines. In this section we describe our hybrid alternative, which combines the superior inductive bias of a GNN at the root node with a computationally inexpensive model at the tree nodes. We also discuss various enhancements to the training protocol, in order to enhance the performance of the learned models.

2.4.1 Hybrid architecture

A variable selection strategy in B&B can be seen as a scoring function f that outputs a score $s_i \in \mathbb{R}$ for every branching candidate. As such, f can be modeled as a parametric function, learned by ML. Branching then simply involves selecting the highest-scoring candidate according to f:

$$i_f^\star = \operatorname*{argmax}_{i \in \mathcal{C}} \mathbf{s}_i$$

We consider two forms of node representations for machine learning models: (i) a graph representation $\mathbf{G} \in \mathcal{G}$, such as the variable-constraint bipartite graph of Gasse et al. (2019), where $\mathbf{G} = (\mathbf{V}, \mathbf{E}, \mathbf{C})$, with $\mathbf{V} \in \mathbb{R}^{n \times d_1}$ variable features, $\mathbf{E} \in \mathbb{R}^{n \times m \times d_2}$ edge features, and $\mathbf{C} \in \mathbb{R}^{m \times d_3}$ constraint features; and (ii) branching candidate features $\mathbf{X} \in \mathbb{R}^{|\mathcal{C}| \times d_4}$, such as those from Khalil et al. (2016), which is cheaper to extract than the first representation. For convenience, we denote by $\mathcal{X}$ the generic space of the branching candidate features, and by $\mathbf{G}^0$ the graph representation of the root node. The various features d_i used in this work are detailed in Appendix A.2.

In B&B, structural information in the tree nodes, $\mathbf{G}$, shares a lot of similarity with that of the root node, $\mathbf{G}^0$. Extracting, but also processing that information at every node is an expensive task, which we will try to circumvent. The main idea of our hybrid approach is then to succinctly extract the relevant structural information only once, at the root node, with a parametric model $\mathrm{GNN}(\mathbf{G}^0; \boldsymbol{\theta})$. We then combine in the tree nodes this preprocessed structural information with the cheap candidate features $\mathbf{X}$, using a hybrid model $f := \mathrm{MLP}_{\mathrm{HYBRID}}(\mathrm{GNN}(\mathbf{G}^0; \boldsymbol{\theta}), \mathbf{X}; \boldsymbol{\phi})$.

By doing so, we hope that the resulting model will approach the performance of an expensive but powerful $f := \text{GNN}(\mathbf{G})$, at almost the same cost as an inexpensive but less powerful $f := \text{MLP}(\mathbf{X})$. Figure 2.2 illustrates the differences between those approaches, in terms of data extraction.

For an exhaustive coverage of the computational spectrum of hybrid models, we consider four ways to enrich the feature space of an MLP via a GNN's output, sumarrized in Table 2.1. In CONCAT, we concatenate the candidate's *root representations* $\mathbf{\Psi}$ with the features $\mathbf{X}$ at a node. In FiLM (Perez et al., 2018), we generate film parameters $\boldsymbol{\gamma}$, $\boldsymbol{\beta}$ from the GNN, for each candidate, which are further used to modulate the hidden layers of the MLP. In details, if $\boldsymbol{h}$ is the intermediate representation of the MLP, it gets linearly modulated as $\boldsymbol{h} \leftarrow \boldsymbol{\beta} \cdot \boldsymbol{h} + \boldsymbol{\gamma}$. While both the above architectures have similar computational complexity, it has been shown that FiLM subsumes the CONCAT architecture (Dumoulin et al., 2018). On the other the end of the spectrum lie the most inexpensive hybrid architectures, HyperSVM and HyperSVM-FiLM. HyperSVM is inspired by HyperNetworks (Ha et al., 2017), and simply consists in a multi-class Support Vector Machine (SVM), whose parameters are predicted by the root GNN. We chose a simple linear disciminator, for a minimal computational cost. Finally, in HyperSVM-FiLM we increase the expressivity of HyperSVM with the help of modulations, similar to that of FiLM.

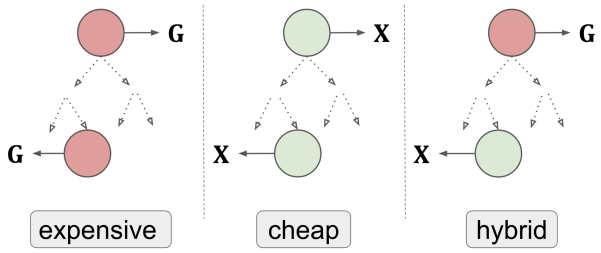

Figure 2.2: Data extraction strategies: bipartite graph representation $\mathbf{G}$ at every node (expensive); candidate variable features $\mathbf{X}$ at every node (cheap); bipartite graph at the root node and variable features at tree node (hybrid).

2.4.2 Training Protocol

We use *strong branching* decisions as ground-truth labels for imitation learning, and collect observations of the form $(\mathbf{G}^0, \mathbf{G}, \mathbf{X}, i^\star_{SB})$. Thus, the data used for training the model is $\mathcal{D} = \{(\mathbf{G}^0_k, \mathbf{G}_k, \mathbf{X}_k, i^\star_{SB,k}), k = 1, 2, ..., N\}$. We treat the problem of

Table 2.1: Various functional forms f considered for variable selection ($\odot$ denotes Hadamard product).

	Data extraction	Computational Cost	Decision Function
GNN Gasse et al. (2019)	expensive	Expensive	$\mathbf{s} = \text{GNN}(\mathbf{G})$
MLP	cheap	Moderate	$\mathbf{s} = \text{MLP}(\mathbf{X})$
CONCAT	hybrid	Moderate	$\mathbf{\Psi} = \text{GNN}(\mathbf{G}^0)$ $\mathbf{s} = \text{MLP}([\mathbf{\Psi}, \mathbf{X}])$
FiLM (Perez et al., 2018)	hybrid	Moderate	$\boldsymbol{\gamma}, \boldsymbol{\beta} = \text{GNN}(\mathbf{G}^0)$ $\mathbf{s} = \text{FiLM}(\boldsymbol{\gamma}, \boldsymbol{\beta}, \text{MLP}(\mathbf{X}))$
HyperSVM	hybrid	Cheapest	$\mathbf{W} = \text{GNN}(\mathbf{G}^0)$ $\mathbf{s} = (\mathbf{W} \odot \mathbf{X})\mathbf{1}$
HyperSVM-FiLM	hybrid	Cheapest	$\boldsymbol{\gamma}, \boldsymbol{\beta_1}, \boldsymbol{\beta_2} = \text{GNN}(\mathbf{G}^0)$ $\mathbf{s} = \boldsymbol{\beta_2}^\intercal \max(0, \boldsymbol{\beta_1} \odot \mathbf{X} + \boldsymbol{\gamma})$

identifying $i^\star_{SB,k}$ as a classification problem, such that $i^* = \text{argmax}_{i \in \mathcal{C}} f(\mathbf{G}^0, \mathbf{X})$ is the target outcome. Considering $\mathcal{D}$ as our ground-truth, our objective (2.1) is to minimize the cross-entropy loss l between $f(\mathbf{G}^0, \mathbf{X}) \in \mathcal{R}^{|\mathcal{C}|}$ and a one-hot vector with one at the target:

$$\mathcal{L}(\mathcal{D}; \boldsymbol{\theta}, \boldsymbol{\phi}) = \frac{1}{N} \sum_{k=1}^{N} l(f(\mathbf{G}^0_k, \mathbf{X}_k; \boldsymbol{\theta}, \boldsymbol{\phi}), i^\star_{SB,k}). \tag{2.1}$$

Performance on unseen instances, or generalization, is of utmost importance when the trained models are used on bigger instances. The choices in training the aforementioned architectures influence this ability. In this section, we discuss four such important choices that lead to better generalization.

End-to-end Training (e2e)

A $\text{GNN}_{\hat{\boldsymbol{\theta}}}$, with pre-trained parameters $\hat{\boldsymbol{\theta}}$, is obtained using the procedure described in Gasse et al. (2019). We use this pre-trained GNN to extract variable representations at the root node and use it as an input to the MLPs at a tree node (pre). This results in a considerable performance boost over plain "salt-of-the-earth" MLP used at all tree nodes. However, going a step further, an end-to-end (e2e) approach involves training the GNN and the MLP together by backpropagating the gradients

from a tree node to the root node. In doing so, e2e training aligns the variable representations at the root node with the prediction task at a tree node. At the same time, it is not obvious that it should result in a stable learning behavior because the parameters for GNN need to adapt to various tree nodes. Our experiments explore both pre-training (pre) and end-to-end training (e2e), namely:

$$\text{(pre)} \qquad \boldsymbol{\phi}^* = \underset{\boldsymbol{\phi}}{\operatorname{argmin}}\, \mathcal{L}(\mathcal{D}; \hat{\boldsymbol{\theta}}, \boldsymbol{\phi}), \qquad \text{(e2e)} \qquad \boldsymbol{\phi}^*, \boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}, \boldsymbol{\phi}}{\operatorname{argmin}}\, \mathcal{L}(\mathcal{D}; \boldsymbol{\theta}, \boldsymbol{\phi}). \quad (2.2)$$

Knowledge Distillation (KD)

Using the outputs of a pre-trained expert model as a soft-target for training a smaller model has been successfully used in model compression (Hinton et al., 2015). In this way, one aims to learn an inexpensive model that has the same generalization power as an expert. Thus, for better generalization, instead of training our hybrid architectures with cross-entropy (Goodfellow et al., 2016) on ground-truth hard-labels, we study the effect of training with KL Divergence (Kullback and Leibler, 1951) between the outputs of a pre-trained GNN and a hybrid model, namely:

$$\text{(KD)} \qquad \mathcal{L}_{KD}(\mathcal{D}; \boldsymbol{\theta}, \boldsymbol{\phi}) = \frac{1}{N} \sum_{k=1}^{N} \mathrm{KL}(f(\mathbf{G}_k^0, \mathbf{X}_k; \boldsymbol{\theta}, \boldsymbol{\phi}), \mathrm{GNN}_{\hat{\boldsymbol{\theta}}}(\mathbf{G_k})). \quad (2.3)$$

Auxiliary Tasks (AT)

An inductive bias, such as GNN, encodes a prior on the way to process raw input. Auxiliary tasks, on the other hand, inject priors in the model through additional learning objectives, which are not directly linked to the main task. These tasks are neither related to the final output nor do they require additional training data. One such auxiliary task is to maximize the diversity in variable representations. The intuition is that very similar representations lead to very close MLP score predictions, which is not useful for branching.

We minimize a pairwise loss function that ensures maximum separation between the variable representations projected on a unit hypersphere. We consider two types of objectives for this: (i) Euclidean Distance (ED), and (ii) Minimum Hyperspherical Energy (MHE) (Liu et al., 2018), inspired from the well-known Thomson

problem (Thomson, 1904) in Physics. While ED separates the representations in the Euclidean space on the hypersphere, MHE ensures uniform distribution over the hypersphere. Denoting $\hat{\boldsymbol{\psi}}_i$ as the variable representation for the variable i projected on a unit hypersphere and $e_{ij} = ||\hat{\boldsymbol{\psi}}_i - \hat{\boldsymbol{\psi}}_j||_2$ as the Euclidean distance between the representations for the variables i and j, our new objective function is given as $\mathcal{L}_{AT}(\mathcal{D};\boldsymbol{\theta},\boldsymbol{\phi}) = \mathcal{L}(\mathcal{D};\boldsymbol{\theta},\boldsymbol{\phi}) + g(\boldsymbol{\Psi};\boldsymbol{\theta})$, where

$$\text{(ED)} \qquad g(\boldsymbol{\Psi};\boldsymbol{\theta}) = \frac{1}{N^2}\sum_{i,j=1}^{N} e_{ij}^2, \qquad \text{(MHE)} \qquad g(\boldsymbol{\Psi};\boldsymbol{\theta}) = \frac{1}{N^2}\sum_{i,j=1}^{N} \frac{1}{e_{ij}}. \tag{2.4}$$

Loss Weighting Scheme

The problem of distribution shift is unavoidable in a sequential process like B&B. A suboptimal branching decision at a node closer to the root node can have worse impact on the size of the B&B tree as compared to when such a decision is made farther from it. In such situations, one can use depth (possibly normalized) as a feature, but the generalization on bigger instances is a bit unpredictable as the distribution of this feature might be very different from that observed in the training set. Thus, we experimented with different depth-dependent formulations for weighting the loss at any node. Denoting z_i as the depth of a tree node i relative to the depth of the tree, we weight the loss at different tree nodes by $w(z_i)$, making our objective function as

$$\mathcal{L}(\mathcal{D};\boldsymbol{\theta},\boldsymbol{\phi}) = \frac{1}{N}\sum_{k=1}^{N} w(z_k) \cdot l(f(\mathbf{G}_k^0, \mathbf{X}_k;\boldsymbol{\theta},\boldsymbol{\phi}), i^{\star}_{SB,k}). \tag{2.5}$$

Specifically, we considered 5 different weighting functions such that all of them have the same end points, i.e., $w(0) = 1.0$ at the root node and $w(1) = e^{-0.5}$ at the deepest node. Different functions were chosen depending on their intermediate behavior in between these two points. We experimented with exponential, linear, quadratic and sigmoidal decay behavior of these functions. Table 2.3 lists various functions and their mathematical forms considered in our experiments.

2.5 Experiments

We follow the experimental setup of Gasse et al. (2019), and evaluate each branching strategy across four different problem classes, namely Capacitated Facility Location, Minimum Set Covering, Combinatorial Auctions, and Maximum Independent Set. Randomly generated instances are solved offline using SCIP (Gleixner et al., 2018) to collect training samples of the form $(\mathbf{G}^0, \mathbf{G}, \mathbf{X}, i^\star_{SB})$. We leave the description of the data collection and training details of each model to Appendix A.5.

Evaluation. As in Gasse et al. (2019), our evaluation instances are labeled as small, medium, and big based on the size of underlying MILP. Small instances have the same size as those used to generate the training datasets, and thus match the training distribution, while instances of increasing size allows us to measure the generalization ability of the trained models. Each scenario uses 20 instances, solved using 3 different seeds to account for solver variability. We report standard metrics used in the MILP community for benchmarking B&B solvers: (i) Time: 1-shifted geometric mean[3] of running times in seconds, including the running times for unsolved instances, (ii) Nodes: hardware-independent 1-shifted geometric mean of B&B node count of the instances solved by all branching strategies, and (iii) Wins: number of times each branching strategy resulted in the fastest solving time, over total number of solved instances. All branching strategies are evaluated using the open-source solver SCIP (Gleixner et al., 2018) with a time limit of 45 minutes, and cutting planes are allowed only at the root node.

Baselines. We compare our hybrid model to several non-ML baselines, including SCIP's default branching heuristic *Reliability Pseudocost Branching* (RPB), the "gold standard" heuristic *Full Strong Branching* (FSB), and a very fast but ineffective heuristic *Pseudocost Branching* (PB). We also include the GNN model from Gasse et al. (2019) run on CPU (GNN), and also several fast but less expressive models such as SVMRank from Khalil et al. (2016), LambdaMART from Burges (2010), and

[3]For complete definition refer to Appendix A.3 in Achterberg (2007)

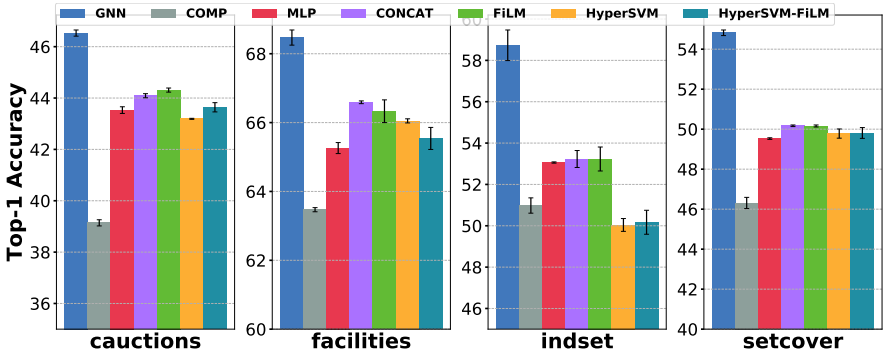

Figure 2.3: Test accuracy of the different models, with a simple e2e training protocol. Note: Part of the axis values have been omitted for clarity.

ExtraTree Classifier from Geurts et al. (2006) as benchmarks. For conciseness, we chose to report those last three competitor models using an optimistic aggregation scheme, by systematically choosing only the best performing method among the three (COMP).

For completeness, we also report the performance of a GNN model run on a high-end GPU, although we do not consider that method as a baseline and therefore do not include it in the Wins indicator. We acknowledge that our comparison to general branching strategy like RPB is not completely fair, however, developing specialized versions of such strategies is a challenge in itself full of modeling choices that we foresee as future work.

Model selection. To investigate the effectiveness of different architectures, we empirically compare the performance of their end-to-end variants. Figure 2.3 compares the Top-1 test accuracy of models across the four problem sets. The performance of GNNs (blue), being the most expressive model, serves as an upper bound to the performance of hybrid models. All of the considered hybrid models outperform MLPs (red) across all problem sets. Additionally, we observe that FiLM (green) and CONCAT (purple) perform significantly better than other architectures. However, there is no clear winner among them. We also note that the cheapest hybrid models, HyperSVM and HyperSVM-FiLM, though better than MLPs, still do not perform as well as FiLM or CONCAT models.

Table 2.2: Test accuracy of FiLM, using different training protocols. The best performing models according to the mean accuracy are in **bold**.

	cauctions	facilities	indset	setcover
Pretrained GNN	44.12 ± 0.09	65.78 ± 0.06	53.16 ± 0.51	50.00 ± 0.09
e2e	44.31 ± 0.08	66.33 ± 0.33	53.23 ± 0.58	50.16 ± 0.05
e2e & KD	44.10 ± 0.09	66.60 ± 0.21	53.08 ± 0.3	50.31 ± 0.19
e2e & KD & AT	**44.56 ± 0.13**	**66.85 ± 0.28**	**53.68 ± 0.23**	**50.37 ± 0.03**

Table 2.3: Effect of different sample weighting schemes on combinatorial auctions (big) instances, with a simple MLP model. $z \in [0, 1]$ is the ratio of the depth of the node and the maximum depth observed in a tree. The best performing scheme according to the size of B&Btree are in **bold**.

Type	Weighting scheme	Nodes	Wins
Constant	1	9678	10/60
Exponential decay	$e^{-0.5z}$	9793	10/60
Linear	$(e^{-0.5} - 1) * z + 1$	9789	12/60
Quadratic decay	$(e^{-0.5} - 1) * z^2 + 1$	9561	14/60
Sigmoidal	$(1 + e^{-0.5})/(1 + e^{z-0.5})$	**9534**	14/60

Training protocols. In Table 2.2, we show the effect of different protocols discussed in Section 2.4 on Top-1 accuracy of the FiLM models. We observe that the presence of these protocols improves the model accuracy by 0.5-0.9%, which translates to a minor yet practically useful improvement in B&B performance of the solver. Except for Combinatorial Auctions, FiLM's performance is improved by knowledge distillation, which suggests that the soft targets of pre-trained GNN yield a better generalization performance. Lastly, we launch a hyperparameter search for auxiliary objective: ED and MHE, on top of the best performing model (Top-1 accuracy), among e2e and e2e & KD models. Auxiliary tasks further help the accuracy of the hybrid models, but for some problem classes it is ED that works well while for others it is MHE. We provide the model performances on test dataset in Appendix A.7 for further reference.

Effect of loss weighting. We empirically investigate the effect of the different loss weighting schemes discussed in Section 2.4.2. We train a simple MLP model on our small Combinatorial Auctions instances, and measure the resulting B&B tree size on big instances. We report aggregated results in Table 2.3, and provide instance-level results in Appendix A.6. We observe that the most commonly used exponential and linear schemes actually seem to degrade the performance of the learned strategy, as we believe those may be too aggressive at disregarding nodes

Table 2.4: Performance of *branching strategies* on evaluation instances. We report geometric mean of solving times, number of times a method won (in solving time) over total finished runs, and geometric mean of number of nodes. Refer to Section 2.5 for more details. The best performing results are in **bold**. *Models were regularized to prevent overfitting on small instances.

	Small			Medium			Big		
Model	Time	Wins	Nodes	Time	Wins	Nodes	Time	Wins	Nodes
FSB	42.53	1 / 60	13	313.33	0 / 59	75	997.23	0 / 51	50
PB	31.35	4 / 60	139	177.69	4 / 60	384	712.45	3 / 56	309
RPB	36.86	1 / 60	**23**	213.99	1 / 60	**152**	794.80	2 / 54	**99**
COMP	30.37	3 / 60	120	172.51	4 / 60	347	633.42	6 / 57	294
GNN	39.18	0 / 60	112	209.84	0 / 60	314	748.85	0 / 54	286
FiLM (ours)	**24.67**	**51** / 60	109	**136.42**	**51** / 60	325	**531.70**	**46** / 57	295
GNN-GPU	28.91	– / 60	112	150.11	– / 60	314	628.12	– / 56	286
Capacitated Facility Location									
FSB	27.16	0 / 60	17	582.18	0 / 45	116	2700.00	0 / 0	n/a
PB	10.19	0 / 60	286	94.12	0 / 60	2451	2208.57	0 / 23	82 624
RPB	14.05	0 / 60	**54**	94.65	0 / 60	1129	1887.70	7 / 27	48 395
COMP	9.83	3 / 60	178	89.24	0 / 60	1474	2166.44	0 / 21	52 326
GNN	17.61	0 / 60	136	242.15	0 / 60	**1013**	2700.17	0 / 0	n/a
FiLM (ours)	**8.73**	**57** / 60	147	**63.75**	**60** / 60	1131	**1843.24**	**20** / 26	**37 777**
GNN-GPU	8.26	– / 60	136	53.56	– / 60	1013	1535.80	– / 36	31 662
Set Covering									
FSB	6.12	0 / 60	6	132.38	0 / 60	71	2127.35	0 / 28	318
PB	2.76	1 / 60	234	25.83	0 / 60	2765	393.60	0 / 59	13 719
RPB	4.01	0 / 60	**11**	26.36	0 / 60	714	**210.95**	**29** / 60	4701
COMP	2.76	0 / 60	82	29.76	0 / 60	930	494.59	0 / 54	5613
GNN	2.73	1 / 60	71	22.26	0 / 60	688	257.99	6 / 60	**3755**
FiLM (ours)	**2.13**	**58** / 60	73	**15.71**	**60** / 60	**686**	217.02	25 / 60	4315
GNN-GPU	1.96	– / 60	71	11.70	– / 60	688	121.18	– / 60	3755
Combinatorial Auctions									
FSB	673.43	0 / 53	47	1689.75	0 / 20	10	2700.00	0 / 0	n/a
PB	172.03	2 / 57	5728	753.95	0 / 45	1570	2685.23	0 / 1	38 215
RPB	59.87	5 / 60	603	173.17	11 / 60	**205**	1946.51	9 / 21	2461
COMP	82.22	1 / 58	847	383.97	1 / 52	267	2393.75	0 / 6	5589
GNN*	**44.07**	15 / 60	**331**	625.23	1 / 50	599	2330.95	0 / 10	**687**
FiLM* (ours)	52.96	**37** / 55	376	**131.45**	**47** / 54	264	**1823.29**	**12** / 15	1201
GNN-GPU*	31.71	– / 60	331	63.96	– / 60	599	1158.59	– / 27	685
Maximum Independent Set									

early on in the tree. On the other hand, both the quadratic and sigmoidal schemes result in an improvement, thus validating the idea that depth-dependent weighting can be beneficial for learning to branch. We therefore opt for a sigmoidal loss weighting scheme in our training protocol.

Complete benchmark. Finally, to evaluate the runtime performance of our hybrid approach, we replace SCIP's default branching strategy with our best performing model, FiLM. We observe in Table 2.4 that FiLM performs substantially better than all other CPU-based branching strategies. The computationally expensive

FSB, our "gold standard", becomes impractical as the size of instances grows, whereas RPB remains competitive. While GNN retains its small number of nodes, it loses in running time performance on CPU. Note that we found that FiLM models for Maximum Independent Set did initially overfit on small instances, such that the performance on larger instances degraded substantially. To overcome this issue we used weight decay Ng (2004) regularization, with a validation set of 2000 observations generated using random medium instances (not used for evaluation). We report the performance of the regularized models in Appendix A.8, and use the best performing model to report evaluation performance on medium and big instances. We also show in Appendix A.9 that the cheapest computation model of HyperSVM/HyperSVM-FiLM do not provide any runtime advantages over FiLM. We achieve up to 26% reduction on medium instances and up to 8% reduction on big instances in overall solver running time compared to the next-best branching strategy, including both learned and classical strategies.

Finally, we note that the majority of "big" problems in Set Covering and Maximum Independent Set are not solved by any of the branching strategies. Therefore, we provide a comparison of optimality gap of unsolved instances at time out in Table 2.5. It is evident that FiLM models are able to close a larger optimality gap than the other branching strategies.

In the absence of significant difference in KD and KD & AT model's Top-1 accuracy (see Table 2.2), a natural question then to ask is: what can be a practically useful choice? To answer this question, we tested the effect of each training protocol on the final B&B performance. Although the detailed discussion is in Appendix A.11 we conclude that in the absence of significant difference in the test accuracy of models between KD and AT & KD, a preference should be made for KD models to attain better generalization.

Table 2.5: Mean optimality gap (lower the better) of commonly unsolved "big" instances (number of such instances in brackets). The best performing model is highlighted in **bold**.

	setcover (33)	indset (39)
FSB	0.1709	0.0755
PB	0.0713	0.0298
RPB	0.0628	0.0252
COMP	0.0740	0.0252
GNN	0.1039	0.0341
FiLM	**0.0597**	**0.0187**

Limitations. We would like to point out some limitations of our work. First, given the NP-Hard nature of MILP solving, it is fairly time consuming to evaluate performance of the trained models on the instances bigger than considered for this work. One can consider the primal-dual bound gap after a time limit as an evaluation metric for the bigger instances, but this is misaligned with the solving time objective. Second, we have used Top-1 accuracy on the test set as a proxy for the number of nodes, but there is an inherent distribution shift because of the sequential nature of B&B that leads to out-of-distribution observations. Third, generalization to larger instances is a central problem in the design of branching strategies. Several techniques discussed in this work form only a part of the solution to this problem. On the Maximum Independent Set problem, we originally noticed a poor generalization capability, which we addressed by cross-validation using a small validation set. In future work, we plan to perform an extensive study of the effect of architecture and training protocols on generalization performance. Another experiment worth conducting would be to train on larger instances than "small" problems used in the work, in order to get a better view of how and to which point the different models are able to generalize. However, not only is this a time-consuming process, but there is also an upper limit on the size of the problems on which it is reasonable to conduct experiments, simply due to hardware constraints. Finally, although we showed the efficacy of our models on a broad class of MILPs, there may be other problem classes for which our models might not result in a substantial runtime improvements.

2.6 Conclusion

As more operations research and integer programming tools start to include ML and deep learning modules, it is necessary to be mindful of the practical bottlenecks faced in that domain. To this end, we combine the expressive power of GNNs with the computational advantages of MLPs to yield novel hybrid models for learning to branch, a central component in MILP solving. We integrate various training protocols that augment the basic MLPs and help bridge the accuracy gap

with more expensive models. This competitive accuracy translates into savings in time and nodes when used in MILP solving, as compared to both default, expert-designed branching strategies and expensive GNN models, thus obtaining the "best of both worlds" in terms of the time-accuracy trade-off. More broadly, our philosophy revolves around understanding the intricacies and practical constraints of MILP solving and carefully adapting deep learning techniques therein. We believe that this integrative approach is crucial to the adoption of statistical learning in exact optimization solvers.

Acknowledgments

This work was funded by CIFAR and IVADO. The compute resources were kindly provided by Compute Canada.

The ideas and experiments in this work were a result of discussions with several people. Many thanks to Felipe Serano and Benjamin Müller for their technical help with SCIP and insightful discussions on branching in MILPs. Further, this work has also been benefited by discussions with Giulia Zarpellon, Didier Chételat, Antoine Prouvost, Karsten Roth, David Yu-Tung Hui, Tristan Deleu, Maksym Korablyov, and Alex Lamb.

3

Lookback for Learning to Branch

Contents

About this work

Publication. Gupta et al. (2022), published in TMLR 2022

Peer-reviews. Peer-reviews can be accessed at `https://openreview.net/forum?id=EQpGkw5rvL`.

Code. Code is made publicly-available at `https://github.com/pg2455/lookback-learn2branch`.

Contributions. I contributed to this project in the following ways,

- Discovery of the lookback phenomenon
- Devising the modified imitation learning framework
- Coding the solutions and running experiments
- Preparing the figures and tables for the publication
- Writing the draft and managing the submission as well as the peer-reviews

Talks. This work has been presented at the following venues,

- Huawei Research ML Seminar, Jan' 2023
- G-Research ML Seminar, Jan' 2023
- SIAM Opt Conference, June' 2023

Abstract

The expressive and computationally efficient bipartite Graph Neural Networks (GNN) have been shown to be an important component of deep learning enhanced Mixed-Integer Linear Programming (MILP) solvers. Recent works have demonstrated the effectiveness of such GNNs in replacing the *branching (variable selection) heuristic* in branch-and-bound (B&B) solvers. These GNNs are trained, offline and on a collection of MILP instances, to imitate a very good but computationally

expensive branching heuristic, *strong branching*. Given that B&B results in a tree of sub-MILPs, we ask (a) whether there are strong dependencies exhibited by the target heuristic among the neighboring nodes of the B&B tree, and (b) if so, whether we can incorporate them in our training procedure. Specifically, we find that with the strong branching heuristic, a child node's best choice was often the parent's second-best choice. We call this the "lookback" phenomenon. Surprisingly, the branching GNN of Gasse et al. (2019) often misses this simple "answer". To imitate the target behavior more closely, we propose two methods that incorporate the lookback phenomenon into GNN training: (a) target smoothing for the standard cross-entropy loss function, and (b) adding a *Parent-as-Target (PAT) Lookback* regularizer term. Finally, we propose a model selection framework that directly considers harder-to-formulate objectives such as solving time. Through extensive experimentation on standard benchmark instances, we show that our proposal leads to decreases of up to 22% in the size of the B&B tree and up to 15% in the solving times.

3.1 Introduction

Many real-world decision making problems are naturally formulated as Mixed-Integer Linear Programs (MILPs), for example, process integration (Kantor et al., 2020), production planning (Pochet and Wolsey, 2006), urban traffic management (Foster and Ryan, 1976; Fayazi and Vahidi, 2018), data center resource management (Nowatzki et al., 2013), and auction design (Abrache et al., 2007), to name a few. In some applications, such as urban traffic management (Fayazi and Vahidi, 2018), these MILPs need to be solved frequently (e.g., every second) with only a slight change in the specifications. Similarly in data center resource management (Nowatzki et al., 2013), the available machines and tasks that must be served evolve over time, prompting repeated assignments through MILP solving. Even with a linear objective function and linear constraints, the requirement that some decision variables must take on integer values makes MILPs NP-Hard (Papadimitriou and Steiglitz, 1982).

As a result, the Branch-and-Bound (B&B) algorithm (Land and Doig, 1960) is used in the modern solvers to effectively prune the search space of the MILPs to

find the global optimum. B&B proceeds by recursively splitting the search space and solving the linear relaxation of the resulting sub-problems, the solution of which serves as an informative bound to prune the search space. The algorithm continues until a solution with integral decision variables is found and proven optimal. Quite naturally, the sub-problems resulting from the branching decisions at each step of the algorithm can be represented as a tree; every node (a sub-MILP) has a parent except the root node.

While seemingly simple, the B&B algorithm must repeatedly make search decisions that are crucial for its efficiency, such as the choice of decision variable over which to branch at each iteration, a problem known as *variable selection*. Even though the worst-case time-complexity of the B&B algorithm is exponential in the size of the problem (Wolsey, 1988), it has been successfully applied in practical settings thanks to the careful design of a number of effective search heuristics.

Modern solvers are configured with expert-designed heuristics and are aimed at solving general MILPs. However, assuming that MILPs come from a specific distribution, there has been a recent surge in research related to statistical approaches to learning such heuristics, e.g., (He et al., 2014b; Alvarez et al., 2017; Khalil et al., 2016; Gasse et al., 2019; Zarpellon et al., 2021; Gupta et al., 2020a; Huang et al., 2022).

Gasse et al. (2019) proposed to use Graph Neural Network (GNN) for the variable selection problem to imitate the branching decisions of a computationally expensive *strong branching* heuristic that yields the smallest B&B trees in practice on a number of benchmarks. The GNN operates on a bipartite representation of a MILP in which variables and constraints are nodes and an edge indicates that a variable has a non-zero coefficient in a constraint. Such a bipartite representation has several advantages. First, it is able to capture the most crucial invariances of MILPs, namely, permutation invariance to the ordering of decision variables/constraints, and, by way of feature engineering, scale invariance to the scaling of constraints or objective coefficients. Second, due to the shared parametric representation, the model can be applied to MILPs of arbitrary size. Thus, GNNs have been extensively used to process MILPs as inputs to a neural network aimed at imitating

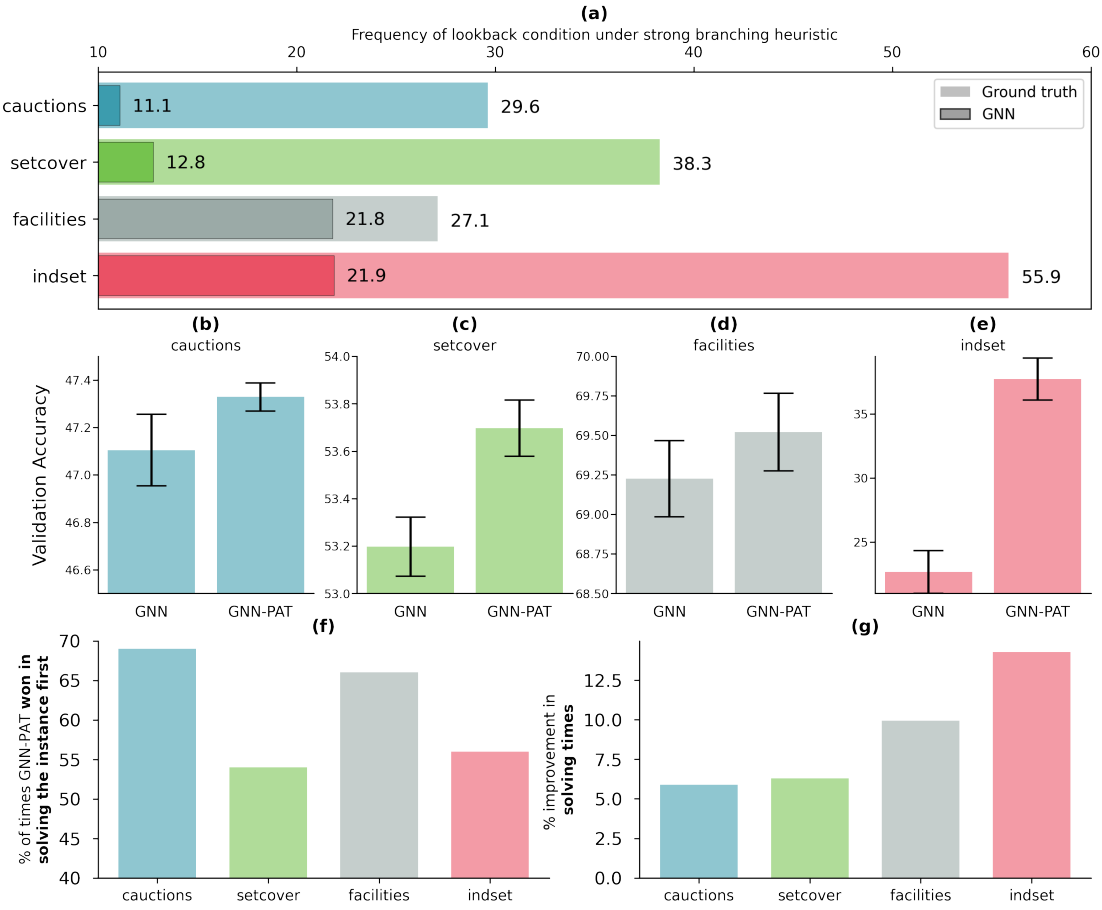

Figure 3.1: **(a)** Frequency of the "lookback" property when (i) instances are solved using the strong branching heuristic (Ground truth), and (ii) corresponding frequency with which GNNs would have respected this property on Ground truth. **(b)-(e)** GNNs trained with the proposed techniques (GNN-PAT) have higher accuracy on the validation dataset. **(f)** GNN-PAT solves test instances faster than other baselines resulting in **(g)** less time as compared to GNNs. Note: Part of the axis values have been omitted for clarity.

various superior heuristic decisions (Huang et al., 2022; Zarpellon et al., 2021; Nair et al., 2020; Khalil et al., 2022), often outperforming off-the-shelf solvers with their default parameters/heuristics.

The training of GNNs consists primarily of two steps: (a) *Dataset collection*: data, i.e., sub-problems and corresponding strong branching decisions, are collected offline by solving reasonably-sized MILPs while invoking, with some exploration probability, either a less efficient but faster heuristic, or the strong branching heuristic, and (b) *Model Training*: assuming independent and identical distribution (i.i.d.) of the collected dataset, GNNs are trained to minimize standard cross-entropy

loss between the predictions of GNNs and the strong branching target.

So far, previous works on learning to branch have tried imitating the strong branching decisions at each node in isolation, thereby ignoring the statistical resemblance in the behavior of the neighboring nodes. However, it often happens that strong branching's top choice at a node was the *second-best* choice at the node's parent, a phenomenon we will refer to onwards as the "lookback" property of strong branching. Figure 3.1(a) shows how frequently this occurs in some standard benchmark instances, labeled for convenience as cauctions (Combinatorial Auction), setcover (Minimum Set Cover), facilities (Capacitated Facility Location), and indset (Maximum Independent Set); see Appendix B.1 for similar statistics on some real-world instance sets, where the lookback property is equally prevalent. Given the importance of decisions closer to the root node, Figure 3.2 further shows that this phenomenon is quite prevalent at the top of the tree. Although it appears quite common empirically, this property has never been explicitly pointed out in the literature.

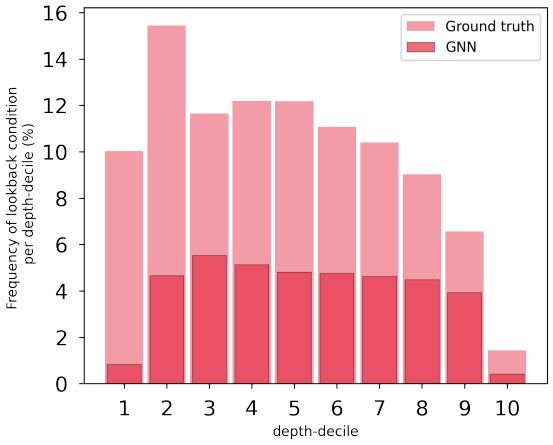

Figure 3.2: (Maximum Independent Set) Frequency of the lookback property per depth-decile, one of the 10 equally divided portions by depth of the B&B tree. Ground truth is obtained by solving small instances using strong branching heuristic. Retrospectively, GNNs do not respect the lookback property well enough. For similar statistics on other benchmark problem families, see Appendix B.2.

Since imitation learning aims to mimic the expert as closely as possible, and the expert exhibits this lookback property, it follows that successful imitation should exhibit the lookback property as well. However, as can be seen in Figures 3.1(a) and 3.2, there is a big gap between how frequently vanilla GNNs respect the lookback condition and the fraction of times it is actually observed. Towards closing this gap, we propose two approaches that encourage GNNs trained by imitation learning to respect the lookback condition, and analyze the effects on solving performance.

A difficulty is that our proposed methods introduce new hyperparameters, which raises the question of how to select them efficiently. Gupta et al. (2020a) used validation accuracy on a few thousand observations collected by solving MILPs of a slightly bigger size than the training instances. We argue that such a selection strategy may not serve the true objectives that a practitioner might have. For instance, they might prefer a learning-enhanced solver with as small a running time as possible, regardless of the validation accuracy relative to the strong branching oracle. Thus, we also design an improved model selection framework to respect such varied requirements. As shown in Figure 3.1(b)-(e), the resulting combined method (GNN-PAT) not only increases the validation accuracy on each benchmark, but also leads to decreases in the size of the final B&B tree by up to 22% and in the running time by up to 15% (Figure 3.1(g)), solving most of the instances fastest among the baselines (Figure 3.1(f)).

To summarize, the contributions of this chapter are as follows.

- First, we demonstrate through numerical experiments that the "lookback" phenomenon often occurs in practice (see Figure 3.1 and 3.2).
- Second, we propose two ways of exploiting this phenomenon, namely through target softening, and through a regularizer term that encourages the GNN to exhibit this phenomenon.
- Third, we propose a model selection framework to incorporate harder-to-formulate practical objectives such as minimum solving times.

The chapter is divided as follows. Sections 3.2 and 3.3 review the literature and preliminary notation and definitions, respectively. Section 3.4 proposes techniques to exploit the lookback phenomenon in imitation learning. Section 3.5 details our proposed hyperparameter selection framework. Then, Section 3.6 presents experimental results that show the benefits of these adjustments. Finally, in Section 3.7, we discuss implications of our proposals, limitations, and potential future directions, concluding in Section 3.8.

3.2 Related Work

The problem of variable selection in B&B based MILP solving has been studied quite extensively. While the gold standard strong branching heuristic yields the smallest B&B trees, due to the high computational cost per iteration it is impractical (see Section 3.3 for details). Thus, in its early years, the focus of research has been on hand-designed heuristic methods (Applegate et al., 1995; Linderoth and Savelsbergh, 1999) that are faster and sufficiently good. As a result, *reliability pseudocost branching* (RPB) (Achterberg et al., 2005) became the preferred branching strategy to solve general MILPs. RPB combines the lookahead strengths of strong branching heuristic with faster estimations offered through computationally inexpensive measures of performance (such as *pseudocosts* (Linderoth and Savelsbergh, 1999)).

In the last decade, researchers have proposed machine learning (ML) methods to imitate the strong branching heuristic, thereby leveraging the computationally inexpensive nature of the learned functions. For example, Alvarez et al. (2017) used extremely randomized trees on the data collected offline, whereas Khalil et al. (2016) used support vector machines to imitate the strong branching ranking from the first few hundred nodes explored in the B&B tree. We refer the reader to Lodi and Zarpellon (2017) for a detailed survey on ML-based branching strategies.

Both Alvarez et al. (2017) and Khalil et al. (2016) learn a classifier on fixed-dimensional, hand-designed features. Recently, however, several works have proposed deep-learning based branching strategies, starting from (Gasse et al., 2019). Their GNN approach has been the basis of several following work. Specifically, Gupta et al. (2020a) explored the relation between the original MILP and subsequent sub-MILPs in a B&B tree to design a CPU-efficient neural network architecture. However, this relationship has more to do with the B&B characteristics than the oracle behavior. Similarly, Zarpellon et al. (2021) explored learning suitable representations based on the evolution of the B&B tree. Peng and Liao (2022) investigated, in the context of combinatorial auctions instances only, the effects of B&B node sampling to collect the training data. Finally, Nair et al. (2020) proposed a GPU-friendly alternating descent method of multipliers approach to solving linear programs that could be

use to run strong branching on larger instances, allowing them to show that the approach could scale to large, heteroneous datasets. In none of these works, however, has the parent-child lookback property been explored in the context of training machine learning based variable selection strategies.

More recently, works have tried to move beyond imitation learning and have investigated reinforcement learning formulations for learning to branch. Sun et al. (2020) used a simple evolutionary strategies approach, but they only obtained improvements on very homogeneous benchmarks, while reporting subpar results on the harder benchmarks of Gasse et al. (2019) used in this work. Zhang et al. (2022a) used imitation learning framework of Gasse et al. (2019) to pre-train GNNs before using a hybrid reinforcement learning and Monte-Carlo tree search framework, reporting more encouraging results. In parallel, Etheve et al. (2020) proposed a Q-learning approach on value functions representing subtree size, and Scavuzzo et al. (2022) reinterpreted their approach as reinforcement learning on a tree-shaped Markov decision processes. Parsonson et al. (2022) proposed a similar mechanism, where the search tree is divided into small diving paths, that are used as imitation targets. These works all seek to address the problem of the long episode lengths using topological information from the branch-and-bound tree.

Although an interesting step forward, these approaches are nonetheless not currently competitive with imitation learning methods, which remain the state of the art. Indeed, they face unusual challenges in this context, including that poor decisions lead to longer, rather than shorter episodes, and also that transitions between states are also particularly computationally slow, since they involve solving linear programs. For these reasons, like most works, we chose to focus on the more successful strategy of imitation learning. Nonetheless, it is plausible that encouraging a lookback property in a reinforcement learning policy could lead to similar benefits to those explored in the context of this work.

3.3 Preliminaries

A MILP is a mathematical optimization problem characterized by a linear objective function and linear constraints in variables $\mathbf{x}$. A generic representation of a MILP is as follows:

$$\min_{\mathbf{x}} \mathbf{c}^\intercal \mathbf{x}, \quad \text{s.t.} \quad \mathbf{A}\mathbf{x} \leq \mathbf{b}, \quad \mathbf{x} \in \mathbb{Z}^p \times \mathbb{R}^{n-p}, \tag{3.1}$$

where $\mathbf{c} \in \mathbb{R}^n$ is the vector of cost coefficients, $\mathbf{A} \in \mathbb{R}^{m \times n}$ is the matrix of constraint coefficients, $\mathbf{b} \in \mathbb{R}^m$ is a vector of constant terms of constraints, and $p > 0$ decision variables are constrained to take integer values.

The B&B algorithm proceeds as follows. The Linear Programming (LP) relaxation of the MILP, obtained by relaxing the integer constraints on the discrete variables, is solved to obtain a lower bound on the global optimum, thereby resulting in $\mathbf{x}^*$ as the optimal solution. If $\mathbf{x}^*$ has integral values for the integer-constrained decision variables, then it is integer-optimal and the algorithm terminates. If not, one of the decision variables with fractional value, $k \in \mathcal{C}$, such that $\mathcal{C} \in \{k \mid \mathbf{x}_k^* \notin \mathbb{Z}, k \leq p\}$, is selected to split the MILP in two sub-MILPs. The resulting sub-MILPs are obtained by adding additional constraints $x_k \leq \lfloor x_k^\star \rfloor$ and $x_k \geq \lceil x_k^\star \rceil$, respectively. We denote as $\mathcal{C}$ the set of *branching candidates*, while the variable x_i is termed as a *branching variable*. The algorithm proceeds recursively in this fashion by selecting the next sub-MILPs to operate on.

Denoting the optimal value of Eq. (3.1) by P, and using the superscripts 0 to denote the parent MILP, $-$ to denote the child MILP obtained by adding the lower bound to the branching variable, and $+$ otherwise. The strong branching heuristic selects the variable x_{sb} that has the maximum potential to improve the bound, namely

$$x_{sb} = \underset{k \in \mathcal{C}}{\operatorname{argmax}} \Big[\max\{P_k^- - P^0, \epsilon_{LP}\} \cdot \max\{P_k^+ - P^0, \epsilon_{LP}\} \Big],$$

where $\epsilon_{LP} > 0$ is a small enough value to prevent the scores to collapse to 0 because of no improvement on either side of the branching (Achterberg, 2007).

3.4 Methodology

In this section, we describe our proposals to incorporate dependencies between successive nodes. A bipartite graph representation of a MILP is denoted by $\mathcal{G} \in (\mathbf{V}, \mathbf{E}, \mathbf{C})$, where $\mathbf{V} \in \mathcal{R}^{n \times d_v}$, $\mathbf{E} \in \mathcal{R}^{k \times d_e}$, and $\mathbf{C} \in \mathcal{R}^{m \times d_c}$. Here, $\mathbf{V}$ is the matrix of features for n decision variables, $\mathbf{C}$ is the matrix of features for m constraints, and $\mathbf{E}$ is the matrix of features for k variable-constraint pairs. The terms d_v, d_c, and d_e are the numbers of input features.

The data is collected by using the strong branching heuristic to solve instances of manageable size, thereby yielding N graphical representations of MILPs, $\{\mathcal{G}_i\}_{i=1}^{N}$, the set of candidate decision variables, $\mathcal{C}_i$, with fractional value at a node i, and their corresponding strong branching scores $\mathbf{s}_i \in \mathbb{R}_{\geq 0}^{|\mathcal{C}_i|}$. We use $\mathbf{s}_{i,j}$ to denote the strong branching score of the j^{th} candidate in $\mathcal{C}_i$. We denote the strong branching target chosen during the solving procedure by y_i, and the set of second-best strong branching variables by $\mathcal{Z}_i = \text{argmax}_{j \neq y_i} \mathbf{s}_{i,j}$. Thus, $\mathcal{Z}_i$ may include variables which have the same strong branching score as y_i if there is a tie, else it includes all the variables with the second-best strong branching score, which can be more than one.

We denote the parent graph by a superscript 0 and the child node by a superscript 1. Thus, the parent bipartite graph of the i-th observation is denoted by $\mathcal{G}_i^0$ and the child graph by $\mathcal{G}_i^1$. Note that we drop the superscripts whenever we do not need a distinction between parent and child nodes. We use $\mathcal{D} = \{(\mathcal{G}_i^0, \mathbf{s}_i^0, \mathcal{G}_i^1, \mathbf{s}_i^1) \mid i \in \{1, 2, 3, ..., N\}\}$ to denote the entire dataset.

Defining a GNN by a function f_θ that is defined over the parameter space Θ, Gasse et al. (2019) proposed to find the optimal parameters by empirical risk minimization of the cross-entropy loss between the predictions and the strong branching target over the dataset $\mathcal{D}$. Thus, if there are n decision variables in $\mathcal{G}_i$, $f_\theta(\mathcal{G}_i) \in \mathbb{R}^n$ represents the scores predicted by the function f_θ. Denoting $\mathbf{y}_i$ as a one-hot encoded vector with the value of 1 at y_i and 0 elsewhere, $\theta_y^\star$ is determined by solving

$$\theta_y^\star = \underset{\theta}{\text{argmin}} \frac{1}{N} \sum_{i=1}^{N} w_i \cdot CE(f_\theta(\mathcal{G}_i), \mathbf{y}_i), \tag{3.2}$$

where CE is the cross-entropy loss function, and w_i is the relative importance given to the observation i, which may depend on the depth of the node in the B&B tree (Gupta et al., 2020a).

3.4.1 Second-best ϵ-smooth loss target

Keeping the cross-entropy loss function as is, we modify the target to a smooth label $\mathbf{z}_i$, i.e., instead of one-hot encoded vector $\mathbf{y}_i$, $\mathbf{z}_i$ carries a value of $1-\epsilon$ at the index of the strong branching target y_i, while the value of ϵ is equally divided among the second-best strong branching decisions in $\mathcal{Z}_i$. Thus, we obtain the optimal parameters as

$$\theta_z^\star = \underset{\theta}{\operatorname{argmin}} \frac{1}{N} \sum_{i=1}^{N} w_i \cdot CE(f_\theta(\mathcal{G}_i), \mathbf{z}_i). \tag{3.3}$$

A modified target such as $\mathbf{z}_i$ in Eq. (3.3) yields parameters that tend to preserve the ranking of the second most important decisions. While intuitive and simple to implement with minimal changes in the existing framework, $\theta_z^\star$ is still not informed from the parent behavior.

3.4.2 Parent-As-Target (PAT) lookback loss term

Here, we are interested in incorporating the relation between parent and child outputs as it happens under the strong branching oracle. In doing so, we expect the learned parameters to more appropriately represent the strong branching behavior. Defining the lookback condition L_i at the node i as

$$L_i = \begin{cases} 1, & y_i^1 \in \mathcal{Z}_i^0 \\ 0, & \text{otherwise,} \end{cases} \tag{3.4}$$

we consider an additional term to Eq. (3.2) or Eq. (3.3) that enforces f_θ to follow the same ordering between parent-child nodes whenever $L_i = 1$. We call this *Parent-As-Target (PAT) lookback* term (Eq. (3.5)), designed to enforce similarity between the logits at the parent node for the candidates $\mathcal{C}_i^1$ of the child node, denoted as $f_\theta(\mathcal{G}_i^0)[\mathcal{C}_i^1]$, and the logits at the child node for the same candidates $\mathcal{C}_i^1$, denoted by $f_\theta(\mathcal{G}_i^1)$.

$$\mathcal{L}_{PAT}(\mathcal{G}_i^1, \mathcal{G}_i^0, \mathcal{C}_i^1) = CE(f_\theta(\mathcal{G}_i^1), f_\theta(\mathcal{G}_i^0)[\mathcal{C}_i^1]), \tag{3.5}$$

Thus, we obtain $\theta^\star_{yPAT}$ with the target $\mathbf{y}_i$ as

$$\theta^\star_{yPAT} = \underset{\theta}{\operatorname{argmin}} \frac{1}{N} \sum_{i=1}^{N} w_i \Big[CE(f_\theta(\mathcal{G}_i^1), \mathbf{y}_i^1) + \frac{NL_i}{\sum_{i=1}^N L_i} \lambda_{PAT} \mathcal{L}_{PAT}(\mathcal{G}_i^1, \mathcal{G}_i^0, \mathcal{C}_i^1) \Big], \tag{3.6}$$

or we obtain $\theta^\star_{zPAT}$ with the target $\mathbf{z}_i$ as

$$\theta^\star_{zPAT} = \underset{\theta}{\operatorname{argmin}} \frac{1}{N} \sum_{i=1}^{N} w_i \Big[CE(f_\theta(\mathcal{G}_i^1), \mathbf{z}_i^1) + \frac{NL_i}{\sum_{i=1}^N L_i} \lambda_{PAT} \mathcal{L}_{PAT}(\mathcal{G}_i^1, \mathcal{G}_i^0, \mathcal{C}_i^1) \Big]. \tag{3.7}$$

The first term in the brackets represents the usual cross-entropy loss which favors getting the top strong branching variable correct. The second term favors aligning the predicted scores of the second-best variable in the parent node whenever it is the best in the child node i. This term is active only when the lookback condition is satisfied. Here, λ_{PAT} is the relative importance given to the lookback condition, whenever it holds.

3.5 Model Selection

The dataset $\mathcal{D}$ is collected on instances of manageable size such that a reasonable number of observations are collected within a certain time budget (e.g., 1 day). We label these instances as *Small*. Given various hyperparameters involved in training machine learning models, a standard practice is to select a model with the best validation accuracy. However, owing to the sequential nature of the branching decisions, the validation accuracy is not necessarily indicative of the models' performance on other metrics such as running time. For example, due to bad branching decisions early on in the search, the models might later face sub-MILPs that are not representative of the training dataset, thereby leading to even worse decisions. Alternatively, a model might make reasonably good guesses of the strong branching variables, but the overall running time may be dominated by the solving time of the intermediate LP relaxations.

In general, a practitioner is interested in using the learned models to solve problems that are potentially bigger than those used for data collection and training.

We label these instances as *Big*. To obtain some estimates of a model's ability to generalize to *Big* instances, we solve K randomly generated *Medium* instances of intermediate size by using each of the learned branching strategies, $b \in \mathcal{B}$, to guide the MILP solver within a time limit of T seconds per instance. The aggregate performance (e.g., arithmetic mean, geometric mean, etc.) of the branching models can be evaluated in terms of (a) solving time, denoted by $t(b)$, across K instances, e.g., 1-shifted geometric mean, (b) node counts, $n(b, \mathcal{B})$, across the instances that are solved by all the strategies in $\mathcal{B}$, or (c) number of solved instances within the time limit, denoted by $w(b)$. Similarly, one can also define a metric based on the optimality gap of unsolved instances.

In exploring generalization metrics, we need to further distinguish between different objectives that a practitioner might have. For example, (a) *Minimum solving time*: the branching strategy that solves the instances as fast as possible; (b) *Maximum number of instances solved*: a branching strategy that solves the most instances to optimality within a certain time budget, irrespective of whether the strategy solved the instances fastest or not; or (c) *Minimum node count*: a branching strategy that yields the smallest trees.

Thus, even though the models are trained to mimic the strong branching oracle, thereby expecting to yield the smallest tree, we incorporate harder-to-formulate objectives by selecting the learned model using a combination of aggregate performance measures. Thus, to incorporate the objective (a), we select the branching strategy as per

$$\begin{aligned} \mathcal{B}^{'} &= \{j \mid j \in \mathcal{B}, \quad t(j) \leq \min_{\mathcal{B}} t(b) + \epsilon_t\}, \\ b^{\star}_{time} &= \underset{b \in \mathcal{B}'}{\operatorname{argmin}}\, n(b, \mathcal{B}^{'}), \end{aligned} \tag{3.8}$$

where we introduce ϵ_t as a tolerance parameter to account for the variability that is inherent to hardware-dependent measurements of solving time. For instance, $\epsilon_t = 1$ considers all the branching strategies $b \in \mathcal{B}$ with aggregate solving time within 1 second of the best such strategy, i.e., $t(b) < \min_{b \in \mathcal{B}} t(b) + 1$.

The objective corresponding to solving the largest number of instances in the minimum amount of time can be formulated as

$$\begin{aligned} \mathcal{B}^{'} &= \{j \mid j \in \mathcal{B}, \quad w(j) = \max_{\mathcal{B}} w(b)\}, \\ \mathcal{B}^{''} &= \{j \mid j \in \mathcal{B}^{'}, \quad t(j) \leq \min_{\mathcal{B}^{'}} t(b) + \epsilon_t\}, \\ b^{\star}_{solved-time} &= \operatorname*{argmin}_{b \in \mathcal{B}^{''}} n(b, \mathcal{B}^{''}), \end{aligned} \tag{3.9}$$

which is slightly different than the objective of selecting the strategy with the minimum time that also solves the most instances,

$$\begin{aligned} \mathcal{B}^{'} &= \{j \mid j \in \mathcal{B}, \quad t(j) \leq \min_{\mathcal{B}} t(b) + \epsilon\}, \\ \mathcal{B}^{''} &= \{j \mid j \in \mathcal{B}^{'}, \quad w(j) = \max_{\mathcal{B}^{'}} w(b)\}, \\ b^{\star}_{time-solved} &= \operatorname*{argmin}_{b \in \mathcal{B}^{''}} n(b, \mathcal{B}^{''}). \end{aligned} \tag{3.10}$$

We discuss other possible formulations in Appendix B.3.

3.6 Experiments

To evaluate the performance of our proposed methods, we consider four benchmark problem families: Combinatorial Auctions, Minimum Set Covering, Capacitated Facility Location, and Maximum Independent Set. These are the same problems that have been used extensively in the "learning to branch" literature (Gupta et al., 2020a; Scavuzzo et al., 2022; Etheve et al., 2020; Sun et al., 2020) since introduced in Gasse et al. (2019). Specifically, we collect a dataset $\mathcal{D}$ for each of the problem families by solving the *Small* instances using SCIP (Gleixner et al., 2018) with the strong branching heuristic. Our models are trained to minimize the objective functions as described in Equations (3.2), (3.3), (3.6), or (3.7). Due to the space constraints, we leave the instance size, dataset collection, and training specifications to Appendices B.4, B.5, and B.6, respectively.

Baselines. To demonstrate the utility of the proposed models, we consider three types of widely used branching strategies: (a) *Reliability Pseudocost Branching* (RPB): Given the online statistical learning aspect of this heuristic, it has been

shown to be the most promising among all. The commercial solvers use this as a default branching strategy; (b) *TunedRPB*: Given that we are focused on learning a branching strategy suitable for problem sets coming from a fixed distribution, we search through the parameters of RPB to select the ones suited best for the problem family. Specifically, we run a grid search on two RPB parameters representing a trade-off between running time and the iterative performance (see Appendix B.7); We select the best performing parameters using the model selection framework from Section 3.5, making this tuned heuristic directly comparable to our method; (c) Graph Neural Networks (GNN): As proposed by Gasse et al. (2019), and widely used in the community, we use GNNs trained on the same dataset as our proposed models. These models have been shown to be the best among all the other machine learning based models. Finally, we also show the performance of our gold-standard strong branching heuristic that is used to collect the training dataset (FSB).

Evaluation. We replace the variable selection heuristic in SCIP Gleixner et al. (2018) with the strategy to be evaluated. For each of the four problem families, we solve 100 randomly generated instances across three scales: *Small*, *Medium*, and *Big* (see Appendix B.4). Since Combinatorial Auctions' big instances are solved fairly quickly, we extend the evaluation to slightly bigger instances. Increasing scale is expected to increase the running time of the B&B algorithm. All *Small* and *Medium* instances used for evaluation are different from those used for training and model selection. Given the NP-Hard nature of the problems, we used the time limit of 45 minutes per instance to solve these instances using SCIP (Gleixner et al., 2018). See Appendix B.8 for the specifications of SCIP and the hardware used for evaluation.

Evaluation Metrics. As per the standard practices in the MILP community, the performance of B&B solvers is benchmarked across the following metrics: (a) Time: 1-shifted geometric mean[1] of solving time of all the instances, irrespective of whether the instance was solved to optimality or not; (b) Nodes: 1-shifted geometric mean of the number of nodes of the *commonly solved instances* (denoted

[1]For a complete definition, refer to Appendix A.3 in Achterberg (2007)

by c in parenthesis for clarity) across all branching strategies; note that this is a hardware-independent measure of performance; (c) Wins: number of instances that were solved (to optimality) the fastest by the branching strategy; (d) Solved: Total number of instances solved within the time limit; and (e) Time: 1-shifted geometric mean of solving time of the *commonly solved instances*. The commonly solved instances are a subset of instances that have been solved to optimality by all the branching strategies.

Model Hyperparameters. We consider a grid search over the following hyperparameters: (a) loss-target $\in \{y, z\}$, where z refers to the modified loss function proposed in Section 3.4.1 and y to the typical loss function that focuses only on the top strong branching variable; (b) $\lambda_{l_2} \in \{0.0, 0.01, 0.1, 1.0\}$, the l_2-regularization penalty; and (c) $\lambda_{PAT} \in \{0.01, 0.1, 0.2, 0.3\}$ to define the strength of the PAT lookback loss term proposed in Section 3.4.2. While we consider λ_{l_2} for both θ_y and θ_z, for θ_{PAT} we consider the best performing model among all the hyperparameters $\{\text{loss-target}, \lambda_{l_2}, \lambda_{PAT}\}$. For each hyperparameter configuration, we train five randomly seeded models as described in Appendix B.6. Finally, θ_y represents the baseline GNN from Gasse et al. (2019) without any of our proposed modifications.

3.6.1 Results

Hyperparameter Selection. Figure 3.3 shows validation accuracy of the best performing hyperparameters for θ_y, θ_z, and θ_{PAT} according to the validation accuracy on collected dataset $\mathcal{D}$. However, to select the models based on the generalization performance, we solve 100 randomly generated medium instances and gather the metrics as described above. Table 3.1 shows the selection of hyperparameters for θ_{PAT} as per various criteria (see Appendix B.9 for the best parameters for θ_y and θ_z). We observe that the models selected by validation accuracy are not always preferred by the selection criteria defined on the evaluation of medium instances. Second, we observe that Eq. (3.9) and (3.10) may or may not have consensus among them; a strategy might have the fastest solving times for all instances except one,

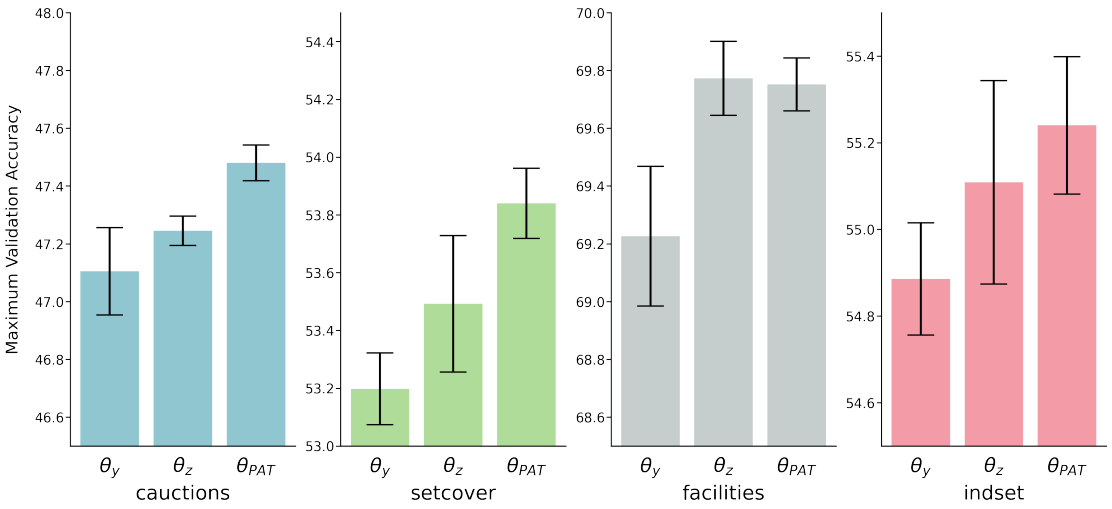

Figure 3.3: Maximum (across the hyperparameters) mean validation accuracy (1-standard deviation) of the proposed models is better than the baseline GNNs (θ_y). We see that the models trained with smoothed target (θ_z) and those with PAT lookback loss term (θ_{PAT}) result in better validation accuracy. Note: Part of the axis values have been omitted for clarity.

but other strategies might solve all the instances in just slightly more time. Third, we observe that the time limit per instance, T, does play a role in model selection; both facilities and indset have a different preference. Finally, we observe that even though the modified target z is not preferred by all the problem families, there is a consensus for the use of the PAT lookback loss term.

Table 3.1: Best performing hyperparameters $\{\text{loss-target}, \lambda_{l_2}, \lambda_{PAT}\}$ for θ_{PAT}. We see that loss-target $= z$ is preferred by some problem families, while the PAT lookback term is preferred by all. In addition, the model selection criteria does impact the chosen hyperparameters.

Problem family	Validation Accuracy	Eq. (3.9)($T = 30$mins)	Eq. (3.10)($T = 30$mins)	Eq. (3.9)($T = 15$mins)
cauctions	$\{y, 0.0, 0.01\}$	$\{z, 0.0, 0.1\}$	$\{z, 0.0, 0.1\}$	$\{z, 0.0, 0.1\}$
setcover	$\{y, 0.0, 0.1\}$	$\{y, 0.0, 0.1\}$	$\{y, 0.0, 0.1\}$	$\{y, 0.0, 0.1\}$
facilities	$\{z, 0.0, 0.0\}$	$\{z, 0.0, 0.1\}$	$\{z, 0.0, 0.1\}$	$\{z, 0.0, 0.3\}$
indset	$\{y, 0.0, 0.1\}$	$\{z, 0.01, 0.2\}$	$\{y, 0.1, 0.3\}$	$\{y, 0.1, 0.3\}$

Validation performance on Medium instances. To assess the interplay between our proposals in Section 3.4 and the model selection framework proposed in Section 3.5, we compare the performance of the selected models using Eq. (3.9) ($T = 30$ minutes) for each of θ_y, θ_z, θ_{yPAT}, and θ_{zPAT}. Figure 3.4 compares their performance on

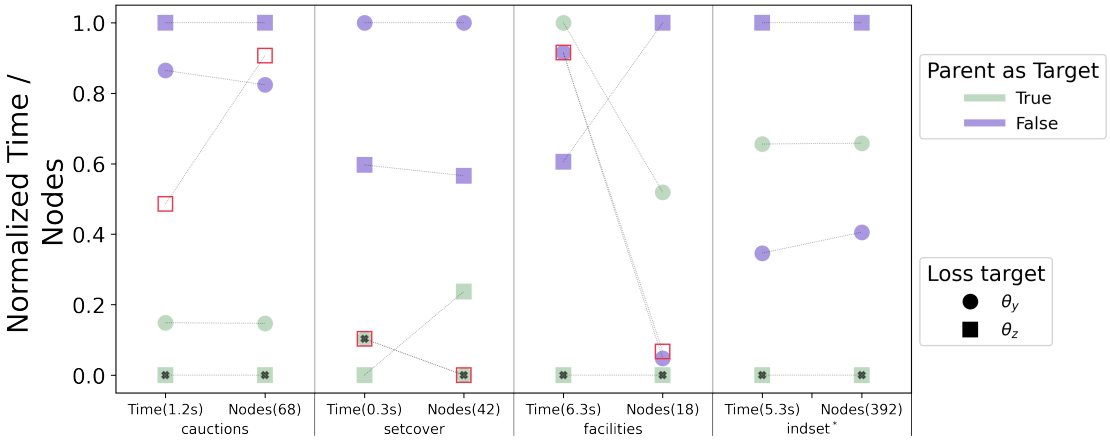

Figure 3.4: We plot the range-normalized (range is specified in parenthesis) Time and Node performance of the selected models as per Eq. (3.9). The centered "X" black mark shows the final models that were selected to be used for evaluating the performance on bigger instances. The points with a red outline show the performance of the models selected according to the best validation accuracy (Note that we omit such models for indset as it distorts the scale of the plot; see Appendix B.10 for complete data.)

medium instances with respect to Time and Nodes. To accommodate different scales of time and nodes, we plot the range-normalized values of these measures and show the range of these measures in parentheses. The centered black marks show the final models selected as per Eq. (3.9). The models chosen as per validation accuracy are shown with transparent marks with red border.

We make the following observations. First, the PAT lookback term is beneficial for the generalization performance most of the time. We observe that for the facilities set of problems, the current GNNs already respect the lookback condition sufficiently well that the proposed modifications do not yield significant improvements compared to θ_y. Second, we see that the cauctions and setcover models perform equally well with respect to time, thereby making the node count an important criterion for identifying better branching models. This is especially important because time measurements are hardware-dependent and thus not as reliable. Third, central to the motivation of our model selection framework, the models chosen as per validation accuracy do not fare well on practical objectives such as Time and Nodes.

Evaluation on Big instances. Given that sets of *Small* and *Medium* instances have already been used in training and selection of the final models, we leave the evaluation

on additional sets of unseen instances from these families to Appendix B.12. Here, we evaluate the performance of the selected models (as per Eq. (3.9)($T = 30$ minutes)) on bigger instances. We use the same instance scaling scheme as proposed by Gasse et al. (2019) (see Appendix B.4).

Table 3.2 shows various evaluation metrics as computed from the evaluation of 100 randomly generated *Big* instances. Since *Big* instances of Combinatorial Auctions are solvable by all the strategies, we extend the scale of these instances to *Bigger* instances.

Specifically, we observe that θ_{PAT} (GNN-PAT) outperforms the baseline model (GNN) in all the problem families on all fronts – Time, Wins, Solved, and Nodes. As an example, for Maximum Independent Set problems, we observe a 15% decrease in Time and a 22% decrease in Nodes. Notably, GNN-PAT increases the number of "Solved" instances by 4 to 5 for all but one problem family. Solving additional instances to optimality is a testament to the improved branching decisions brought about by GNN-PAT.

Table 3.2: Performance of *branching strategies* on *Big* evaluation instances. The best performing numbers are in **bold**. Refer to Section 2.5 for metrics. Since all of the *Big* Combinatorial Auctions instances were solved to optimality, we extend the evaluation to *Bigger* instances (*since only 10 instances were solved using FSB, we omit it here; See Appendix B.11 for the full results including those on *Big* instances.)

Model	Time	Time (c)	Wins	Solved	Nodes (c)
FSB*	n/a	n/a	n/a	n/a	n/a
RPB	626.81	434.92	1	80	17 979
TUNEDRPB	644.20	450.06	0	80	18 104
GNN	507.06	333.59	14	80	17 145
GNN-PAT (ours)	**477.26**	**310.22**	**69**	**84**	**16 388**
Combinatorial Auction (Bigger)					
FSB	2700	n/a	0	0	n/a
RPB	1883.32	1213.57	1	47	58 766
TUNEDRPB	1851.83	1168.58	0	48	58 155
GNN	1708.99	991.07	5	54	39 535
GNN-PAT (ours)	**1601.28**	**892.85**	**54**	**59**	**38 385**
Set Covering					
FSB	917.39	758.19	8	85	50
RPB	737.66	607.19	3	92	104
TUNEDRPB	751.27	619.27	2	91	**97**
GNN	646.03	525.80	16	92	293
GNN-PAT (ours)	**581.91**	**471.20**	**66**	**95**	304
Capacitated Facility Location					
FSB	2700	n/a	0	0	n/a
RPB	1984.91	888.04	0	32	12 407
TUNEDRPB	2016.85	952.25	0	33	12 940
GNN	1207.62	279.71	14	65	7934
GNN-PAT (ours)	**1035.32**	**233.97**	**56**	**70**	**6122**
Maximum Independent Set					

Finally, as noticed above, the learned models for Maximum Independent Set might result in a different hyperparameter configuration based on the selection

criterion. Therefore, we compare the performance of the GNN-PAT models that are selected by each of the Eqs. (3.9) and (3.10) against GNN in Table 3.3. We observe that as per the selection criterion of Eq. (3.9), the branching strategy solved the most number of instances. However, as Eq. (3.10) prefers the strategy with overall lower running time, we observe superior performance of the branching strategy on Time of commonly solved instances. These observations confirm that the proposed model selection approach yields the expected outcomes on unseen test instances.

Similarly, we look at the effect of the time limit T on the selection criterion. The models selected as per Eq. (3.9) for facilities differ depending on the specified time limit T (see Table 3.1). We look at the how the time limit might affect generalization performance in Table 3.3. Specifically, we observe that insufficient time to evaluate *Medium* instances may lead to suboptimal hyperparameters. To conclude, we've shown that the model selection criterion will impact generalization performance significantly.

3.7 Discussion

An objective in this chapter was to imitate the strong branching behavior more closely by taking advantage of its lookback property. The proposals in Section 3.4 are aimed at doing so. A post-hoc analysis shows up to 16% improvement in the GNNs' ability to follow the lookback property (see Appendix B.13). Such improvements are evident at various depths from the root node (see Appendix B.14).

We can argue that the proposals in Section 3.4 are regularizers or inductive biases. Although the modified target and the PAT lookback term were inspired to induce the required oracle behavior, owing to the lack of interpretability of GNNs, we cannot attribute the GNN-PATs' performance improvements to the ability to follow the lookback property with 100% certainty. However, if we consider that the proposals did cause the observed improvements, we might consider it as an inductive bias. Inductive biases are defined as any pre-coded knowledge about the target behavior, such as neural network architecture, choice of features, or loss function.

Table 3.3: Performance measures on branching strategies selected as per different criteria specified in Eqs. (3.9) and (3.10) and the specified time limit per instance T. We observe that each criterion supports the respective measure on scaled-up instances. The best performing models are highlighted in **bold**.

Model	Time	Time (c)	Wins	Solved	Nodes
GNN	1207.62	753.81	1	65	29 573
GNN-PAT (Eq. 3.9)	**1035.32**	621.42	**38**	**70**	**23 574**
GNN-PAT (Eq. 3.10)	1063.12	**613.96**	31	66	**21 162**
		Maximum Independent Set			
GNN	646.03	562.20	13	92	314
GNN-PAT (Eq. (3.9)(T=30mins))	**581.91**	**503.85**	**58**	**95**	**326**
GNN-PAT (Eq. (3.9)(T=15mins))	635.04	551.53	24	94	388
		Capacitated Facility Location			

Further, the PAT lookback loss term is similar to the objective function in knowledge distillation (Hinton et al., 2015), i.e., we are minimizing the cross-entropy between the logits of one model (child node) and the other model (parent node). Whereas in the original distillation framework, there are two separate models (teacher and student) that act on the same inputs, the PAT lookback term can be understood as a form of *cross-distillation* that acts through the same model but on different observations which share some characteristics (e.g., output logits). Thus, we can understand that the PAT lookback term distills knowledge from the parent node to the child node, as was intended.

Finally, our model selection framework enables us to incorporate complex metrics like Time and Nodes into hyperparameter selection criterion. We hope that this framework will help aligning the research in machine learning methods for MILP solvers with practitioners' varied objectives.

Limitations. As illustrated in Figure B.2, the GNNs for facilities instances are capable of capturing the lookback condition 80% of the time. Since it is not possible to consider all possible problem families and their varied formulations, we cannot make a definitive claim on whether our proposed modifications will be useful all the time. Therefore, we recommend checking for the prevalence of the lookback condition to get some idea of expected improvement.

Further, due to time and resource constraints, we restricted the evaluation for model selection to just 100 *Medium* instances. However, for a more robust selection,

we suggest using a larger set of instances. One can also consider setting the size of *Medium* instances such that majority of them are solved, thereby resulting in robust selection of better performing hyperparameters; see Table 3.1 on how the best hyperparameters vary according to time limit per instance T and Table 3.3 for the effect of T on generalization performance.

Finally, we emphasize that the model selection criteria are very much dependent on how these models will be deployed. For example, a practitioner might only be concerned with solving the maximum number of instances (to optimality) while ensuring the smallest optimality gaps in the unsolved instances. This objective can be formulated within our proposed framework, but considering all such formulations is beyond the scope of this work.

Future work. Although we did not specify the minimum optimality gap as the objective of the branching strategies, we ran a post-hoc analysis to compare 1-shifted geometric mean of optimality gaps of the commonly unsolved instances (lower is better). Figure 3.5 shows that, except for setcover instances, the proposed branching strategies are able to close larger gaps than the rest. We acknowledge that, depending on the use, the optimality gap might be of primary importance to the practitioner. We think that the exploration of optimality gap as a secondary objective could be a subject of future work.

As evident from the gaps in Figure B.2 (Appendix B.13) and Figure B.3 (Appendix B.14), we plan to design more ways to incorporate the lookback condition explored in this work. While we studied how the parent and child nodes in a B&B tree are related with respect to a simple PAT lookback condition, there still exists several ways in which such nodes can be related (see Appendix B.15 for another example). Thus, the design of machine learning algorithms to *discover* and *exploit* such dependencies could be an important direction for future research. Moreover, such machine learning-aided discoveries can be equally important for the MILP community to inspire the design of novel heuristics or improve the existing manually-designed heuristics applicable to general instances.

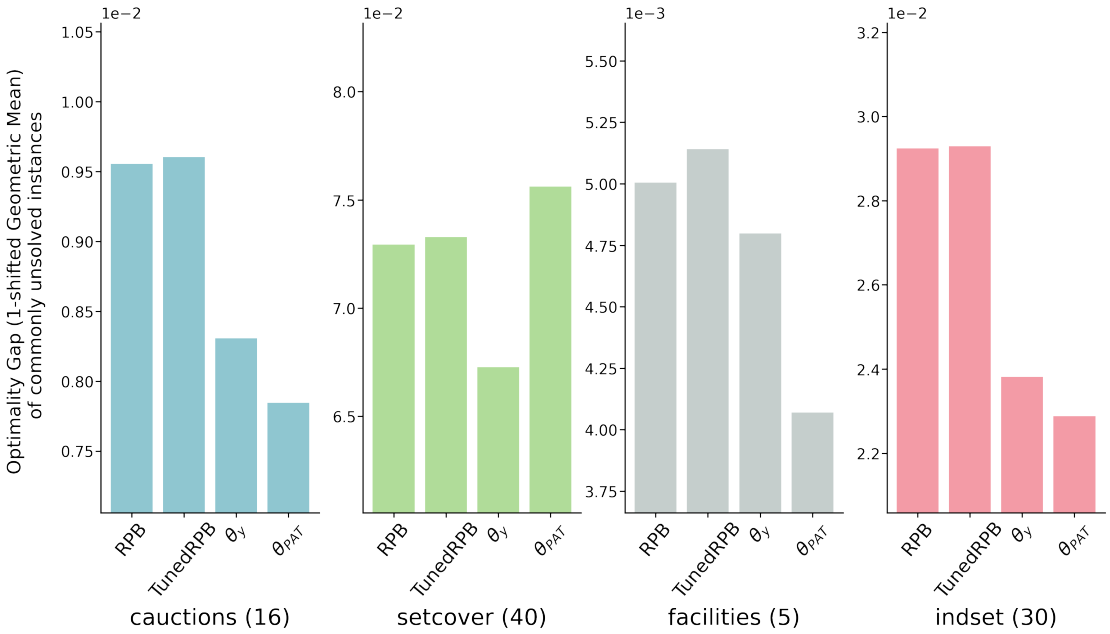

Figure 3.5: Post-hoc analysis of optimality gap of commonly unsolved instances (number is shown in parentheses next to the problem family label) shows that θ_{PAT} is able to achieve the best optimality gap (except for setcover) even though it is not the primary objective specified in the model training. Note: Part of the axis values have been omitted for clarity.

3.8 Conclusion

With the huge gap between the performance of deep learning based heuristics and the oracle heuristics, we expect that the research efforts might require more in-depth investigation of how to imbue these models with the same "reasoning" as the oracles themselves. In this line of thought, we investigated how the parent-child nodes of a B&B tree are related to each other under the oracle heuristic. We found that quite often, the parent's second-best choice is the child's best choice. To incorporate this lookback condition into model training, we designed two methods to align the models more closely with the strong branching oracle's behavior. We believe that this investigative approach to imitating oracle behavior could be a useful way to close the gap between machine learning and the oracle heuristics.

4

Proactive Contact Tracing

Contents

About this work

Publication. Gupta et al. (2023), published in PLOS Digital Health 2023

Peer-reviews. Peer-reviews can be accessed at `https://journals.plos.org/digitalhealth/article/peerReview?id=10.1371/journal.pdig.0000199`.

Code. Code is made publicly-available at `https://github.com/mila-iqia/COVI-AgentSim`.

Contributions. My contributions to this project include:

- Collaborating with privacy and user behavior experts to gain a comprehensive understanding of the constraints in developing an effective contact tracing framework. This effort was carried out as part of a larger team led by me.
- Participating in a team-wide brainstorming session to formulate the idea of the proposed contact tracing framework, which is presented in this chapter.
- Leading a team of AI and epidemiology researchers to develop a simulator for the contact tracing framework, borrowing the standard practices in epidemiology literature
- Writing over 100,000 lines of Python code to create the simulator, incorporating both AI and epidemiology expertise to reflect COVID-19 characteristics
- Setting up the experimental design and running the experiments to test the sensitivity of these methods across a wide range of parameters
- Facilitating communication between AI and epidemiology specialists, and finding ways to convey the complex relationship between these fields to diverse audiences
- Collaborating with economists and policy experts to share project findings with relevant stakeholders
- Drafting, submitting, and managing the peer-review process for the project.

The rules of Rule-based PCT were developed through a collaborative effort with epidemiologists and extensive experimentation with the simulator. The determination of key symptoms and assigned risk values were based on empirical findings gathered through experiments run by my collaborators. The creation of the simulator and the design of the rules were intertwined and continuously evolving, as the results of each aspect fed into the other to drive further development.

Talks. This work has been presented at the following venues,

- Probabilistic Risk Awareness for COVID-19, Bank of Canada Sept '20
- Early-Warning Signals of COVID-19, CIFAR/ELLIS Workshop, Oct '20
- ICLR 2021 (Spotlight talk)
- 10th UK–wide Doctoral Researcher Award, Sept '21
- AI UK Lightning Talk, March '23

Abstract

The COVID-19 pandemic spurred an unprecedented demand for interventions that can reduce disease spread without excessively restricting daily activity, given negative impacts on mental health and economic outcomes. Digital contact tracing (DCT) apps have emerged as a component of the epidemic management toolkit. Existing DCT apps typically recommend quarantine to all digitally-recorded contacts of test-confirmed cases. Over-reliance on testing may, however, impede the effectiveness of such apps, since by the time cases are confirmed through testing, onward transmissions are likely to have occurred. Furthermore, most cases are infectious over a short period; only a subset of their contacts are likely to become infected. These apps do not fully utilize data sources to base their predictions of transmission risk during an encounter, leading to recommendations of quarantine to many uninfected people and associated slowdowns in economic activity. This

phenomenon, commonly termed as "pingdemic", may additionally contribute to reduced compliance to public health measures.

We propose a novel DCT framework, Proactive Contact Tracing (PCT), which uses multiple sources of information (e.g. self-reported symptoms, received messages from contacts) to estimate app users' infectiousness histories and provide behavioral recommendations. PCT methods are by design *proactive*, predicting spread before it occurs. We present an interpretable instance of this framework, the *Rule-based PCT* algorithm, designed via a multi-disciplinary collaboration among epidemiologists, computer scientists, and behavior experts. Finally, we develop an agent-based model that allows us to compare different DCT methods and evaluate their performance in negotiating the trade-off between epidemic control and restricting population mobility. Performing extensive sensitivity analysis across user behavior, public health policy, and virological parameters, we compare *Rule-based PCT* to i) binary contact tracing (BCT), which exclusively relies on test results and recommends a fixed-duration quarantine, and ii) household quarantine (HQ). Our results suggest that both BCT and *Rule-based PCT* improve upon HQ, however, *Rule-based PCT* is more efficient at controlling spread of disease than BCT across a range of scenarios. In terms of cost-effectiveness, we show that *Rule-based PCT* pareto-dominates BCT, as demonstrated by a decrease in Disability Adjusted Life Years, as well as Temporary Productivity Loss.

4.1 Introduction

Governments worldwide have implemented a variety of non-pharmaceutical interventions (NPIs) to contain the spread of COVID-19. Although the strictest of these measures effectively contained case surges in many settings (Vinceti et al., 2020), they have had dramatic consequences including long-term impacts on individuals' mental health (Rossi et al., 2020), financial security (Bonaccorsi et al., 2020), and on local and global economies (Mandel and Veetil, 2020).

At the time of writing this chapter, global immunization was still a distant possibility. Moreover, due to the emergence of highly infectious variants capable of

immune escape (Cele et al., 2021), public health departments aggressively pursued test-and-trace initiatives to curb viral transmission while relaxing or removing restrictions on social and economic activities. Traditionally, the tracing process begins each time a COVID-19 case (an 'index case') is reported. For example, in the U.K, public health authorities directly interview index cases to identify any contacts, defined as persons who were recently within two meters of the index case for fifteen minutes or more (Keeling et al., 2020). This approach, *Manual Contact Tracing*, is time-consuming given the need for follow-up with each individual and challenges in accurately recalling contacts (Kretzschmar et al., 2020; Coughlin, 1990). The scale of the pandemic has consequently stretched the capacity of many health departments undertaking manual tracing (Watson et al., 2020; Czypionka et al., 2022).

Digital contact tracing (DCT) applications, in contrast, employ electronic devices (e.g., phones using Bluetooth) to record encounters between users, reducing the burden of recall and facilitating notification of contacts. DCT applications can additionally capture characteristics of encounters (e.g., proximity, duration) and anonymously exchange information between users.

Most deployed DCT apps adopt a simple model wherein all digitally-recorded contacts of confirmed index cases receive an app notification recommending quarantine (Martin et al., 2020). We refer to such methods as Binary Contact Tracing (BCT) because the inputs consist of binary information about index cases' test results (positive or negative), and the outputs are binary notifications to contacts (quarantine or not). BCT approaches may be sub-optimal for COVID-19 control for the following reasons: First, BCT treats all recorded contacts of an index case as equally infectious; i.e., it does not use contacts' own information (e.g., symptoms, proximity and duration of encounters) to evaluate their likelihood of being infected or infecting others, thereby recommending quarantine to many more people than are infectious (Ivers and Weitzner, 2020; Kleinman and Merkel, 2020). Such overly restrictive notifications can contribute to "pandemic fatigue", leading to reduced compliance (Briscese et al., 2020). Second, people who never develop symptoms or experience mild symptoms may nevertheless be infectious. A systematic review

based on previous SARS-CoV-2 variants suggested that 40% of infections were asymptomatic (He et al., 2020; Luo et al., 2020; Buitrago-Garcia et al., 2020), and recent evidence (Murray, 2022) suggests as high as 80-90% of infections with the Omicron variant are asymptomatic. In the absence of population-wide screening and limited testing capacity, such individuals may not qualify for testing. Finally, infectiousness of those who eventually develop symptoms appears to peak around the time of symptom onset (To et al., 2020). Symptomatic cases are therefore likely to infect others *prior* to being tested, further limiting effectiveness of BCT. Altogether, BCT's exclusive reliance on test results may lead to absent or delayed notification of infectious contacts, undermining epidemic control.

Finally, with the emergence of highly transmissible variants of concern such as Omicron, public health departments have become overwhelmed with test-and-trace approaches. Some nations have even dropped them altogether, shifting responsibility to undertake regular rapid testing and adopt preventive behaviors (including notifying contacts) onto individuals and businesses. Alternative proactive and accurate DCT methods are needed, more than ever, to bridge the gap between BCT and manual tracing methods.

4.1.1 Proactive Contact Tracing

In this chapter, we propose *proactive contact tracing* (PCT), a novel contact tracing framework that uses carefully-designed predictors of infectiousness informed by a rich set of features (e.g., pre-existing conditions, daily symptoms, "risk message") available locally on a given user's smartphone. These predictors privately estimate current and past d-day infectiousness for that user, enabling *proactive* early warning signals of disease. For our purposes, we assume $d = 14$, the total infectiousness period reported in the early days of the pandemic. Discretized values of infectiousness estimates, which we call *risk messages*, are propagated through the network of app users at regular intervals, using a peer-to-peer Bluetooth communication protocol, e.g., COVI (Alsdurf et al., 2020), or alternatively, a centralized protocol (see Appendix C.1 for a discussion on possible design of such a system). Based on

the estimate of the user's current infectiousness, the app recommends one of four sets of increasingly restrictive behaviors, ranging from complete self-isolation to precautions such as avoiding public transport or working from home. A behavior study (Ayres et al., 2020) conducted to understand the efficacy of such an app-based notification system concluded that "less risk-taking by small portions of the population may produce large benefits."

Received risk messages are an input to the local predictor of infectiousness. Therefore, predictions for a given user are indirectly influenced by all other app users. PCT dynamically responds to information as it becomes available (e.g. when informative symptoms are logged), resulting in propagation of an updated risk message at the next interval. Further, PCT improves upon BCT as it allows for less restrictive recommendations in the presence of uncertainty (e.g. in the absence of test results). By suggesting alternative behavioral modifications rather than a fixed-duration quarantine, PCT may impose fewer restrictions on non-infectious individuals, potentially alleviating the economic impact of DCT methods. Hence, PCT not only has the ability to quickly and accurately flag potential infection to a user, but it also proactively generates early-warning signals to recommend cautionary behaviors to likely-to-be-infected users (see Figure 4.1), thus preventing onward transmission.

PCT relies on designing a predictor that can accurately estimate a user's infectiousness from the available features. This is a non-trivial task because the predictions (and corresponding uncertainties) for one user are fed back into the same predictor for other users through risk messages. Research is underway to develop sophisticated deep-learning-based predictors for this purpose, with preliminary findings suggesting unique design and deployment challenges (we discuss this in Chapter 5). For this reason, analyses presented here focus on a simpler, rule-based predictor (described in Section 4.2.2) which is interpretable to clinicians and public health stakeholders. *Rule-based PCT* is also computationally faster to run, including on legacy smartphones.

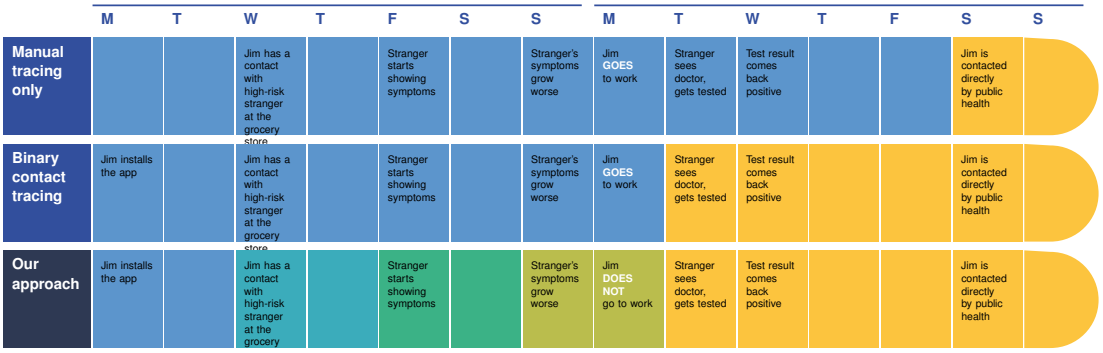

Figure 4.1: Motivating example comparing manual, binary, and proactive contact tracing: This example shows the potential effectiveness of early warning signals in controlling the spread of the infection. We see that manual tracing suffers from a delay between laboratory-confirmed diagnosis and informing all contacts. Further, both manual and digital contact tracing send late signals because they only make use of the strongest possible signal (a laboratory-confirmed diagnosis). The proposed PCT approach takes advantage of reported symptoms and the propagation of risk signals between phones to obtain much earlier signals.

4.1.2 Objectives

Contact tracing methods must negotiate a trade-off between controlling epidemic spread and restricting mobility in a population such that social and economic activities can occur. This chapter assesses, via a detailed agent-based model (COVI-AgentSim), how well PCT and BCT negotiate this trade-off in comparison to *Household Quarantine (HQ)*, which mandates a 14-day quarantine for each confirmed case's entire household. For each method, the app is deployed among a proportion of agents (app users) in a synthetic population. Scenarios estimate the ratio of *cumulative cases* (an indicator of epidemic spread) to *mean number of contacts allowed between agents* (an indicator of population mobility). We further conduct cost-effectiveness analyses across varying degrees of governmental pandemic response. To demonstrate and validate our simulator with real statistics, we focus our analysis on Montréal, Canada. However, the simulator can be configured to any region for which appropriate statistics are available.

4.2 Methods

To evaluate the performance of contact tracing apps, we proceed as follows: (i) implement an agent-based model (ABM) to simulate COVID-19 spread in a population of interest, (ii) develop an app-based tracing mechanism which suggests user-behavior modifications, and (iii) compare different contact tracing methods.

4.2.1 Agent-based model

To simulate the spread of COVID-19 virus in a population and generate naturally-occurring phenomena (e.g. emergence of symptoms) as well as app-based events (e.g. communication of risk messages), we implement COVI-AgentSim[1], a discrete event agent-based model that emulates information propagation between two agents as designed in COVI (Alsdurf et al., 2020), a peer-to-peer communication protocol. Here we present a short description of the simulator; a more detailed description is presented in Appendix C.2.

We sample agent demographics corresponding to the region of Montréal, Québec, Canada using census data[2] and additional epidemiological data sources. The simulator advances the state of the agents by moving them from one location to another, where locations are either home, workplace/school, or of miscellaneous types. At a specific location l, potential encounters are sampled for each agent such that the final aggregated contact pattern resembles that of empirically derived age-stratified contact matrices (Prem et al., 2017) for the Montréal region. We use the procedure typical of demographic standardization (Arregui et al., 2018) to project the Canada–wide contact matrices to the Montréal region, and further refine this derivation using a scaling factor to accommodate fine-grained information available for the Montréal region (Brisson et al., 2020) (e.g., number of contacts at house or school).

We consider four levels, $\{1, 2, 3, 4\}$, of in-app behavior recommendations, with increasing levels restricting the number of close contacts. Whereas an agent in

[1]Code available at `https://github.com/mila-iqia/COVI-AgentSim`

[2]Census profile, 2016 census - Montréal, Québec and Canada. Accessed: 2020-06-02

behavior level 0 (not recommended through app) samples the pre-pandemic mean number of contacts per location, behavior level 4 corresponds to quarantine, in which the agent is recommended to stay at its residence, where they do not sample any contacts. Level 3 represents the post-confinement behavior and exhibits a reduced number of contacts compared to the level 0, where the reduction factor follows from empirical surveys (Brisson et al., 2020). Finally, we introduce two intermediate behavior levels $\{1, 2\}$ each exhibiting half the reduction factor compared to the next higher level (see Table 1 in Appendix C.2 for further details). Our choice of intermediate reduction factors is motivated by simplicity, however, we acknowledge that such reduction factors will likely be influenced by behavior recommendations in these levels, thereby making it crucial to have inputs from PHEs and user behavior experts. As in real life, agents do not always adhere to recommended behavioral restrictions. On a particular day, an agent drops out of their current behavior level to level 1 with a dropout probability (Ferretti et al., 2020). In addition, the simulator assumes imperfect reporting of symptoms or test results by the agents.

Virus transmission, as modeled in the simulator, takes place with a probability (Ferretti et al., 2020) anytime an infectious and a susceptible individual are within 2 meters of each other for at least 15 minutes. See COVID-19 Transmission Model in Appendix C.2 for more details on how this probability is computed. When an agent is infected, we sample a proxy measure for viral load, termed Effective Viral Load (EVL), and symptoms to be experienced by the agent for each day of the infection until recovery. EVL for an agent is modeled with a piece-wise linear function varying between 0 and 1, based on parameters such as incubation period, infectiousness onset, and recovery period, sampled according to the published literature (To et al., 2020; Lauer et al., 2020). See Figure C.7 for a piecewise template of effective viral load in our ABM and Figure C.8 for a mean sampled effective viral load.

We use age-stratified smartphone/app adoption rates for the population as in Ferretti et al. (2020). Thus, given a proportion of the population with a smartphone in different age groups, we vary the global uptake parameter to attain a specific adoption rate. The simulator implements a digital communication

protocol between agents with smartphones to enable contact tracing methods as discussed next.

4.2.2 Tracing Methods

Household Quarantine (HQ). We consider a baseline scenario in which agents who receive a confirmed positive RT-PCR test are quarantined for 14 days. Other residents of agents' households are also quarantined for the same period.

Binary Contact Tracing (BCT). Under BCT, any agent with an app who reports a positive RT-PCR test result broadcasts this information to alert all of their digitally stored contacts from the past 14 days. Those contacts and their household members are put in level 4, i.e., quarantined, occasionally dropping out as per the dropout probabilities.

Rule-based Proactive Contact Tracing (Rule-based PCT). In PCT, the decision about which recommendation level to show to an agent is executed via a predictor. This predictor is part of the contact tracing application on each participating agent's smartphone. The predictor uses the following "features": symptoms, individual characteristics (e.g. age, sex), RT-PCR test results, and anonymized risk messages received from other users (described next) to estimate the agent's mean infectiousness for each of the past 14 days. At regular intervals, each app user's smartphone sends an update to their contacts with a discretized estimate of their infectiousness when the encounter took place; we refer to these as "risk messages". In this chapter, we focus on a rule-based predictor for the ease of interpretability and computational costs. However, we note that the data generated by Covi-AgentSim could be used to train machine learning algorithms (see Chapter 5).

Rule-based PCT uses a set of heuristics to predict an agent's 14-day risk history. This risk history is an estimate of how likely the agent was infectious over the past 14 days, and can be used to provide behavioral recommendations to the agent and their past contacts. These rules are guided by empirical data and the combined knowledge of experts in relevant domains such as epidemiology, virology and behavior research.

Rule-based PCT processes each input feature independently to estimate the agent's infectiousness. For example, the predictor will use the agent's history of symptoms to produce a symptom risk history (Gaythorpe et al., 2020). After constructing a risk history based on each input feature, the predictor constructs the final risk history by taking the maximum of (a) the highest risk estimated for that day derived from the input features, and (b) the previously estimated risk for that day. As illustration, this method would not recommend to quarantine an agent and their past contacts given only a mild symptom such as a cough, but instead make a more moderate recommendation. Figure 4.2 shows a simplified flow chart of the algorithm, and Appendix C.3 provides the complete algorithm.

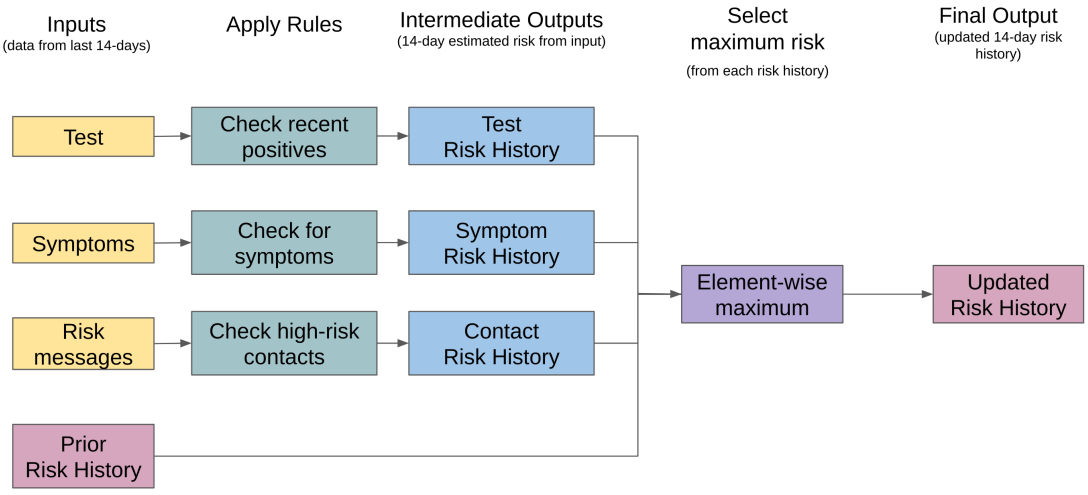

Figure 4.2: Rule-Based PCT Overview. This simplified diagram shows how information flows through the rule-based PCT algorithm. This algorithm is run **each** day on the agent's 14-day history of features to estimate its risk history. Each day the algorithm is run, it takes as input their current RT-PCR test history, user-input symptom history, and anonymized risk message history. Next, a ruleset is applied independently to each input type yielding estimates of the user's risk history over the past 14 days. Finally, to construct a single conservative estimate of the user's risk history, RB-PCT takes the maximum risk across all estimates including the previously generated estimates. There are some exceptions to these rules (e.g. negative test results reset some of the risk history to low values); a full description is provided in Appendix C.2.

4.2.3 Evaluation

We compare the performance of rule-based PCT, BCT, and HQ scenarios across a range of model parameters that influence user behavior, public health policy, and virological assumptions. For each scenario and fixed value of parameters (see Table 4.1), we aggregate results across 50 randomly seeded, 75-day simulations, each of 10K agents with 40 initial infections (chosen to fit our computing budget). To evaluate the performance of these scenarios across varying degrees of outbreak severity, we run 50 simulations each with a randomly sampled global mobility parameter, β, defined as the likelihood of foregoing any given contact. For example, under $\beta = 0.6$, *any* sampled contact will be considered a simulated contact with a probability of 0.4. Thus, β aims to mimic the strength of government-imposed mobility restrictions such as lockdown and stay-two-meters-apart public campaigns. This parameter allows us to control virus containment under any given scenario by varying the number of daily contacts without changing the virus transmission model.

To compare different scenarios across varying parameter values, we compute the ratio of the cumulative incidence (denoted by $\hat{H}$) and mean daily contacts per agent (denoted by $\hat{E}$), both computed at the end of the simulation. This ratio helps us understand how well a tracing method performs in reducing infection spread while allowing for encounters. We compute $\hat{E}$ by summing up encounters across all the agents throughout the period of the simulation and, finally, dividing by the number of days and the population size to yield mean daily contacts per agent. Thus, $\hat{E}$, roughly, serves as a proxy for economic welfare throughout the simulation. The results presented in the subsequent sections use the mean value of $\frac{\hat{H}}{\hat{E}}$ and 1 standard error to compare the different tracing methods; a lower ratio indicates superior performance.

Finally, to account for aleatoric uncertainty, we bootstrap our simulations to compute model means and standard errors. For each scenario, we draw, with replacement, 1,000 random subsets of size 60 from 60 simulations and compute the mean of the performances for each subset. Thus, we obtain a distribution

Table 4.1: Parameter ranges across which the performance of contact tracing methods is evaluated.

Parameter	Description	Minimum value	Maximum value	Default value
User-behavior parameters				
Adoption Rate†	The percentage of population using a contact tracing enabled app	20%	60%	60%, 30%*
Recommendation Adherence	The percentage of app-users that are likely to follow recommendations on any given day	36%	98%	98%
Reporting of symptoms	The likelihood of reporting symptoms for any app-user	20%	80%	80%
Public health policy parameters				
Testing Capacity	Maximum percentage of population that can receive a test on any day	0.05%	0.5%	0.1%
Virological parameters				
Asymptomatic population	The percentage of population that doesn't show any symptoms on getting infected	20%	40%	30%
Asymptomatic infectiousness ratio	Infectiousness ratio of asymptomatic cases relative to symptomatic cases	29%	45%	29%

† The parameter is also influenced by public health policy.
*We present all the results across two adoption rates.

over the performance metric under each scenario. The plots show means and 1-standard errors of these distributions.

Our simulations are run on a population of 10,000 agents for 75 days. To enhance the efficacy of the PCT algorithm, we anticipate refining the Rule-based PCT algorithm according to the dynamic conditions of the pandemic. Specifically, our recalibration will consider the evolving proportions of the infected population and the corresponding government-imposed restrictions. Thus, for convenience, we chose the specified period of two and a half months to simulate one full epidemic wave.

We further evaluate the sensitivity of parameters across 30% and 60% adoption rates. Finally, we show more metrics in Appendix C.4 for these simulations.

Cost-effectiveness Analysis

To provide additional information on the potential real-life utility of these methods, we perform cost-effectiveness analyses. To quantify the health and economic impacts of different DCT methods, we respectively compute lost Disability-Adjusted Life Years (DALYs) and Temporary Productivity Loss (TPL) for simulations run on COVI-AgentSim. For a reliable cost-benefit analysis, it is necessary to understand the full patient journey and each agent's possible health state until reaching a post-epidemic steady state such that agents that were infected either recovered or died.

Such a trajectory, which is obtained by running the simulations for longer, helps assess whether the contact tracing methods actually avert early death, or simply delay it; miscategorizing the timing and/or death event greatly impacts the overall public health benefits of DCT methods. To account for this challenge, we run longer simulations of 180 days. By the end of these simulations, agents are either susceptible, recovered or dead.

A key challenge decision-makers face is how to allocate resources to maximize public health (Detsky and Naglie, 1990) with the lowest possible economic impact. A tool is needed to compare different health interventions and their associated costs, particularly to weigh the trade-off between improved health outcomes and higher economic costs. The incremental cost-effectiveness ratio (ICER) is a measure that allows for direct comparison across health interventions; the ICER is the difference in costs of two interventions divided by the difference in health outcomes (i.e. effectiveness) of the interventions. The ICER provides critical information for policy makers and enables decision-making on resource allocation and prioritization across healthcare services.

For mathematical definitions of DALYs, TPL, and ICER, we refer the reader to Appendix C.5.

4.3 Results

With the default parameter values, Figure 4.3 shows, for each method, the fraction of the population that has been infected up to any day, i.e, cumulative incidence (left) and the daily proportion of agents quarantined (right). We observe that PCT results in a lower infected fraction over 60 days at app adoption rates of 60% (top) and 30% (bottom). The fraction of quarantined agents is generally lower for PCT at 60% adoption, indicating that it better negotiates the epidemic control/mobility restriction trade-off. In contrast, the designed rules for PCT did not result in a lower quarantine rate at 30% adoption, suggesting a need for more sophisticated methods for designing these rules (see Chapter 5).

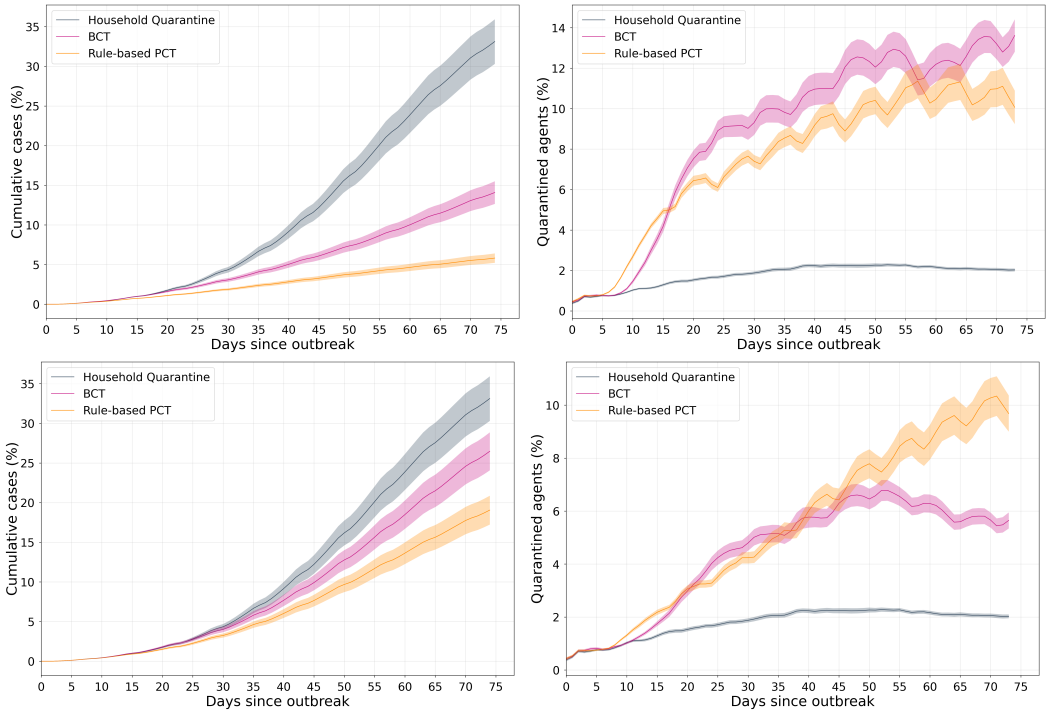

Figure 4.3: Left: Cumulative case counts for each method (fraction of the population) and **Right:** Mobility restriction (fraction of population quarantined), for 60% (**top**) and 30% (**bottom**) app adoption **Gist:** Both BCT and PCT significantly reduce cases as compared to HQ, even at lower app adoption; benefits increase with increasing adoption. With higher adoption, (top) PCT imposes less restriction while achieving much greater reduction in cases. However, at lower adoptions, the reduction in cases is achieved via more conservative recommendations, highlighting the need for more sophisticated predictors if app adoption is low. The plots show mean and 1-standard error bands of these quantities. Note that the HQ scenario includes (false) quarantines because household members of an infected individual are recommended quarantine irrespective of their infection status.

4.3.1 Adoption Rate

App adoption rate is a dominant factor affecting the performance of contact tracing methods. Figure 4.4 compares the trade-off metric for each method under varying app adoption rates. The observed $\frac{\hat{H}}{\hat{E}}$ ratios suggest that PCT improves upon BCT across all adoption rates, even in the context of low uptake (i.e. 20%).

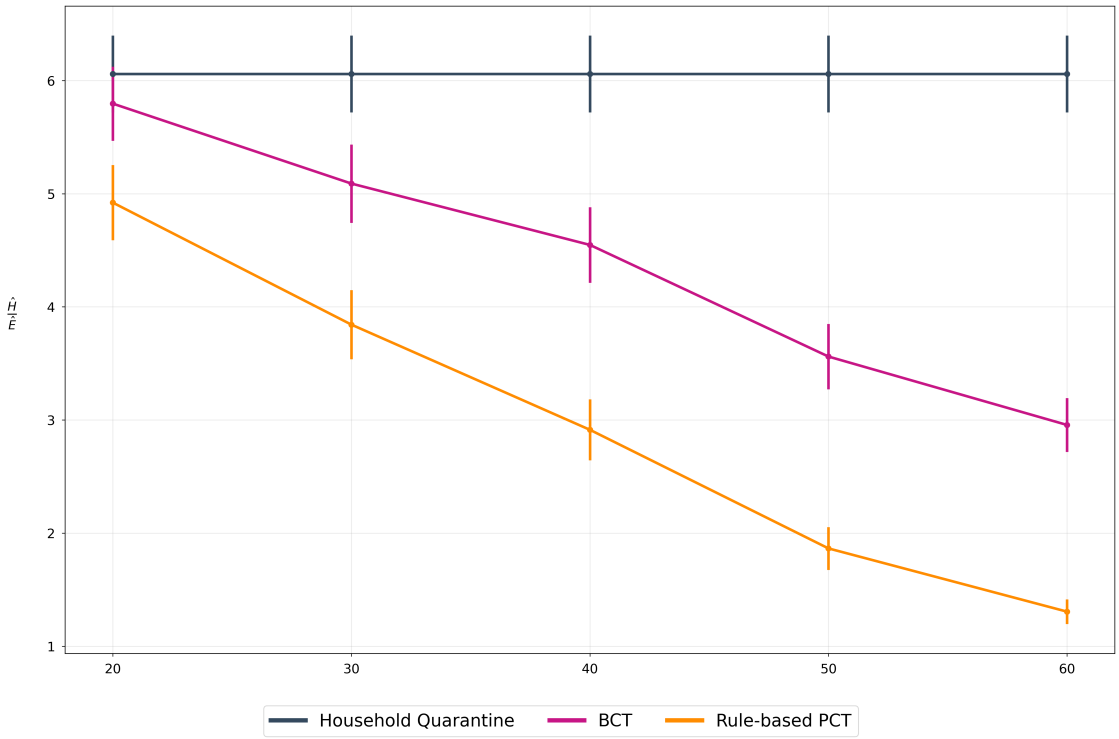

Figure 4.4: Adoption rate comparison. We compare all methods for adoption rates between 0% (HQ) and 60% of both BCT and PCT. **Gist:** Both BCT and PCT methods are able to improve over the HQ scenario, even at low adoption rates. We also observe that PCT is able to negotiate the health-economic trade-off better than BCT (lower the ratio, better is the trade-off). We further compare this performance across adoption rates in cost-benefit analysis.

4.3.2 Sensitivity Analyses

We compare how each tracing method performs across the range of parameter values listed in Table 4.1. We vary one parameter at a time, keeping others at their default values. Figure 4.5 shows sensitivity to varying the following: (A) Recommendation adherence: proportion of app-users following recommendations on a given day; (B) Symptom reporting: likelihood of daily symptom reporting by app-users; (C)

Testing capacity: daily percentage of the population that can receive an RT-PCR test; (D) Asymptomatic fraction: proportion of agents that never develop symptoms. We observe that both BCT and PCT consistently result in lower $\frac{\hat{H}}{\hat{E}}$ ratios (our metric to quantify public health and economy trade-off) compared to HQ, congruent with other studies (Ferretti et al., 2020; Abeler et al., 2020) demonstrating the usefulness of contact tracing apps as components of epidemic management.

Figure 4.5A suggests that recommendation adherence has a drastic impact on the performance of tracing methods while a modest decrease in performance is found as the probability of reporting symptoms increases (Figure 4.5B). The relative advantage of PCT over BCT is maintained across a broad range of testing capacities (Figure 4.5C) and PCT results in a lower $\frac{\hat{H}}{\hat{E}}$ ratio than BCT and HQ at every value of asymptomatic fraction (Figure 4.5D). At 30% adoption, the trade-off associated with PCT improves as the fraction of asymptomatic cases is increased; this relationship is, however, absent at 60% adoption.

4.3.3 Infectiousness of asymptomatic cases

Given the emergence of SARS-CoV-2 variants characterized by greater transmission potential, we assess how different methods perform under increased infectiousness of asymptomatic cases relative to symptomatic cases (Figure 4.6). For all methods, the epidemic control and mobility restriction trade-off worsens as relative infectiousness of asymptomatic cases increases, however, PCT methods remain superior to BCT and HQ.

4.3.4 Cost-effectiveness analysis

To better understand the potential societal impact and aid in decision making, Fig 4.7(a) shows the performance of each scenario across two dimensions quantifying the socio-economic burden of COVID-19: (1) Disability-Adjusted Life Years (DALYs; unit: years) quantifies years of healthy life lost due to disease or poor health (Kyu et al., 2018); (2) Temporary Productivity Loss (TPL; unit: $) measures the economic cost to society of individuals' temporary absence from work. Under a HQ scenario,

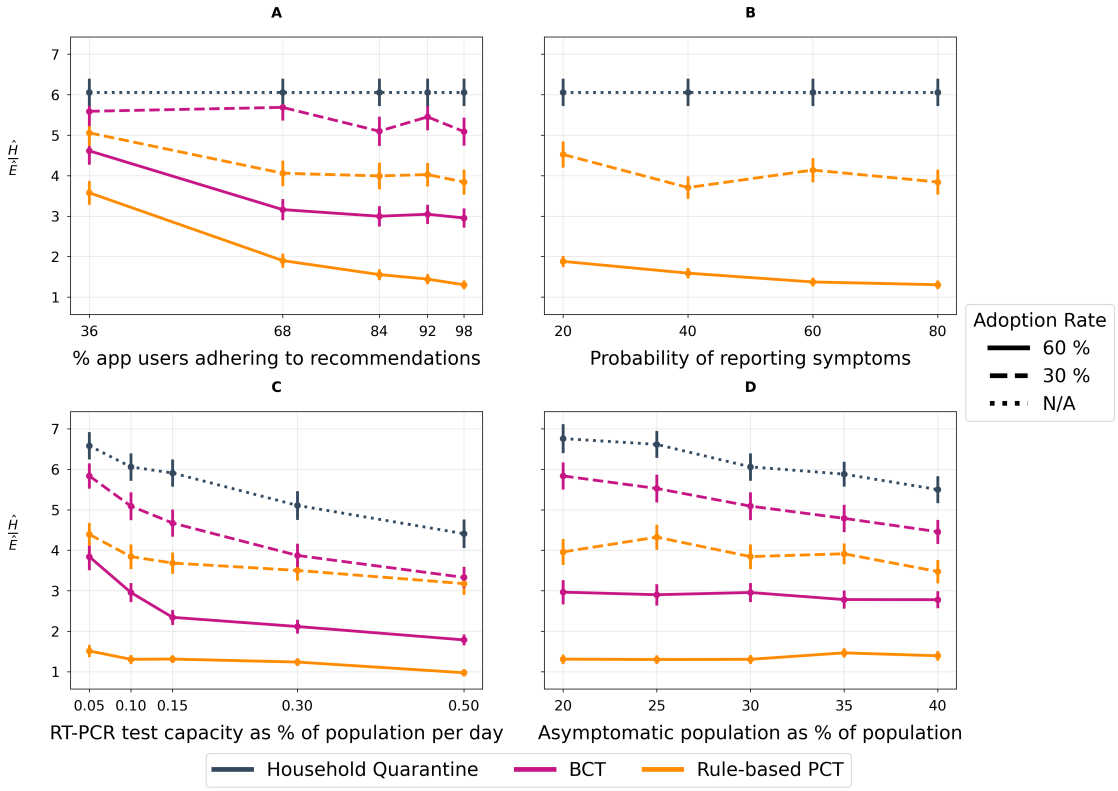

Figure 4.5: Sensitivity Analyses. All experiments measure the proportion of the population infected, $\hat{H}$, within 60 days of an outbreak normalized by mean daily contacts per person per day, $\hat{E}$. We plot $\frac{\hat{H}}{\hat{E}}$ for each tracing method across two app adoption rates as well as against a baseline household quarantining scenario. A lower $\frac{\hat{H}}{\hat{E}}$ ratio indicates a better trade-off between epidemic control and restriction of population mobility. We use N/A to represent irrelevance of adoption rate in the baseline scenario as no DCT app is deployed. **(A) Recommendation Adherence.** Illustrates the impact of varying recommendation adherence (e.g. the daily likelihood of getting a test, quarantining, reducing contacts given an in-app notification is received). **(B) Symptom reporting.** Illustrates the impact of varying the daily rate of symptom reporting. Note: the plot omits BCT because BCT doesn't incorporate symptoms in its inputs. **(C) RT-PCR Testing Capacity.** Illustrates the impact of varying the percentage of the population that can receive an RT-PCR test on any given day, ranging from the observed provincial testing capacity of 0.1% to a highly optimistic value of 0.5% of the population. **(D) Infectiousness and symptoms.** Illustrates the impact of varying the proportion of cases that will not develop symptoms.

with minimal population restrictions, we observe the least TPL but the highest quantity of DALYs, while all DCT scenarios save years of healthy life (lower DALYs) at an economic cost (higher TPL).

To further compare these scenarios pairwise, in Fig 4.7(b) we calculate incremental cost-effectiveness ratios (ICER), defined as the difference in cost (TPL) between

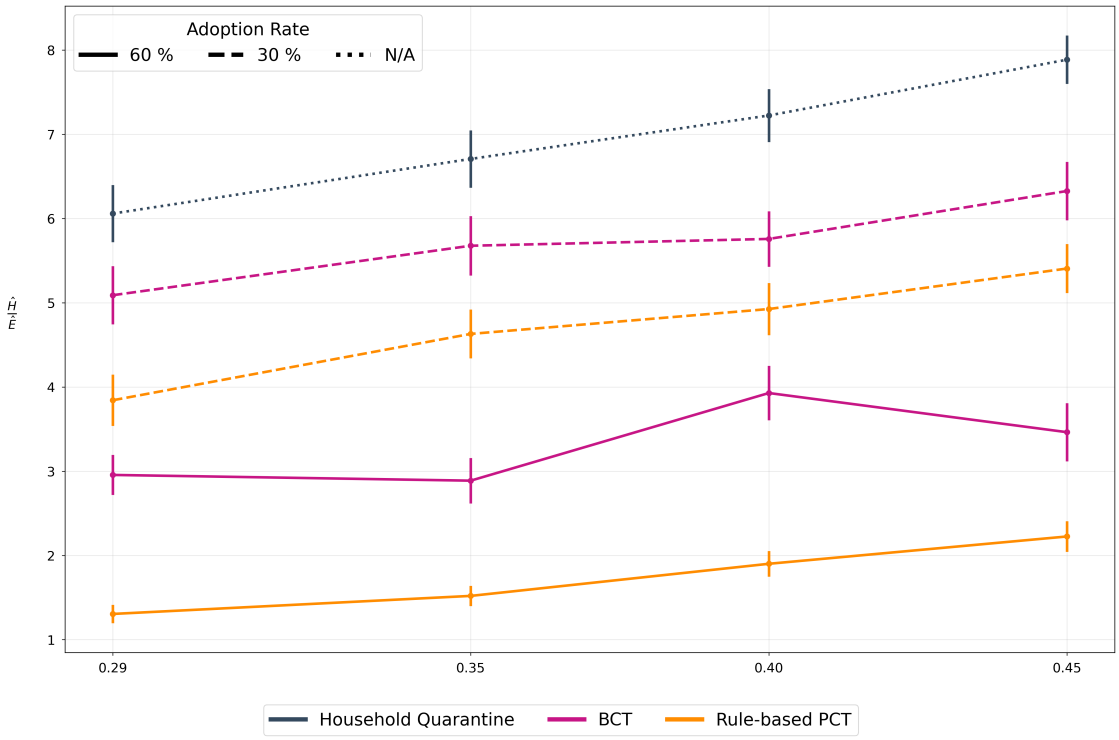

Figure 4.6: Asymptomatic infection ratio. We vary the relative infectiousness of asymptomatic cases. A value of f implies that the asymptomatic case can potentially infect f times as many people as compared to a symptomatic case. A value of 0.29 is the chosen minimum as described in the epidemiological literature while a higher value of 0.45 is a hypothetical situation describing a more infectious variant of the virus. Once again, we use N/A to represent irrelevance of adoption rate in the baseline scenario as no DCT app is deployed. **Gist** As the infectiousness of asymptomatic cases increases, their timely and accurate detection becomes increasingly important. Thus, all the scenarios show a degradation in performance. However, owing to the early warning signals of PCT, it retains its advantage across the range of infection ratios.

two scenarios divided by the difference in their effect (DALYs). Lower DALYs, TPL, and ICER are desired; see Appendix C.5 for a detailed mathematical description of these metrics. Fig 4.7(b) indicates that the ICER of PCT is well below that of BCT with HQ as a reference intervention across all adoption rates. Finally, at adoption rates of 30% or more, we observe that PCT is cost-saving with respect to BCT, i.e., PCT results in better health and economic outcomes than BCT, which makes the ICER of PCT with BCT as a reference intervention negative.

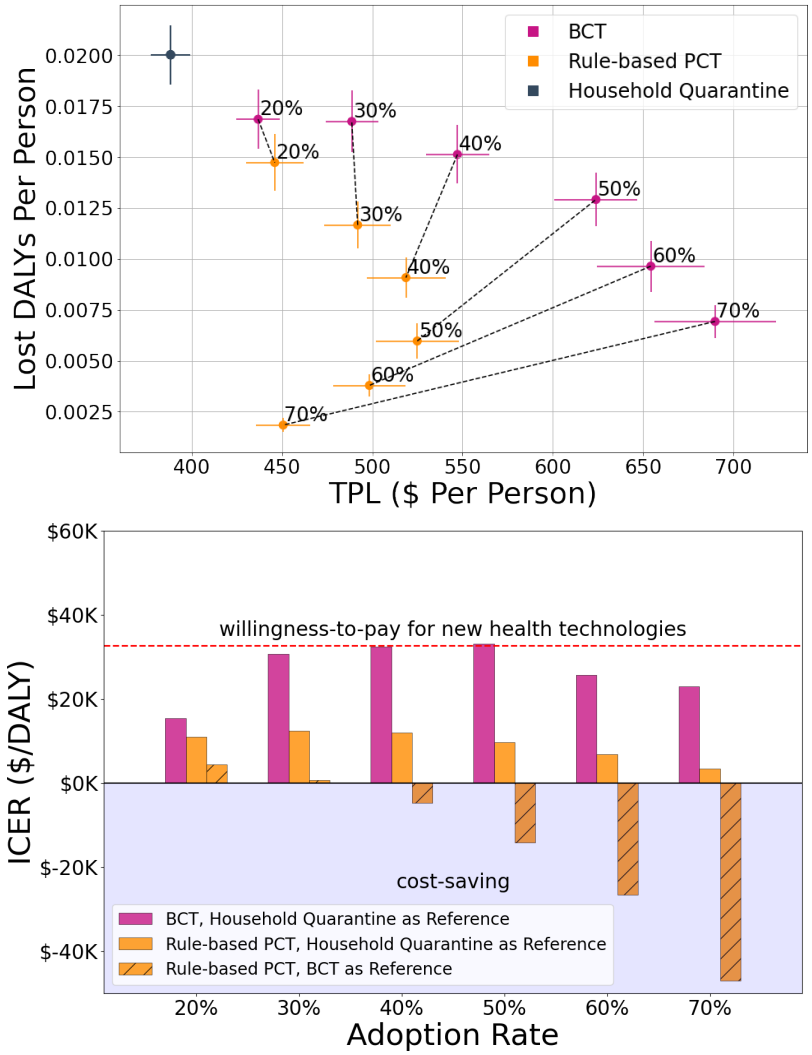

Figure 4.7: Cost-effectiveness analysis. (a) We evaluate the trade-off between temporary productivity loss (TPL) per person and disability-adjusted life years (DALYs) per person for each of the scenarios (HQ, BCT, PCT) at various adoption rates (annotations). **(b)** Further, we compute incremental cost-effectiveness ratios (ICER) of BCT (pink bars) and PCT (yellow bars) with respect to HQ (unshaded bars) and of PCT with respect to BCT (shaded bars) to quantify these trade-offs with a single metric The dashed red line represents a willingness-to-pay threshold for new health technologies (Laupacis et al., 1992) of $33K in 2020 Canadian dollars (see Appendix C.5 for calculations). **Gist** Increased adoption rates lead to better health outcomes for both BCT and PCT, with PCT performing better than BCT across all adoption rates. Additionally, above 30% adoption, PCT is able to leverage the additional information to dominate BCT, making PCT cost-saving with respect to BCT in this regime. This pareto dominance of PCT over BCT above 30% adoption is shown in figure (b) by negative ICER values of PCT with BCT as a reference intervention (shaded bar). Across all adoption rates, the ICER of PCT is well below the willingness-to-pay threshold for new health technologies. As a final note, increases in the adoption rate of PCT above 50% lead to increasingly cost-saving outcomes.

4.4 Discussion

4.4.1 Proactive Contact Tracing

This chapter presents PCT, a novel digital contact tracing framework. We evaluate how well *Rule-based PCT*, an instantiation of the PCT framework, negotiates the trade-off between epidemic control and restriction of mobility as compared to the existing DCT apps and a baseline household quarantining scenario. Sensitivity analyses conducted across user-behavior, public health policy, and virological parameters show that the *Rule-based PCT* consistently improves over BCT.

Several modeling studies to infer infection status of individuals rely on the partial availability of contact graphs (i.e. who meets whom) which sacrifices user anonymity (Baker et al., 2021; Bianconi et al., 2021). By using risk messages, which contain only a few bits (e.g. 2,3,4) at a time, and a well-designed local predictor, PCT is able to accurately identify infections while preserving user-privacy. To account for uncertainty in predictions, PCT provides personalized cautionary recommendations reducing users' mobility instead of completely halting their activity.

COVI-AgentSim was designed with the possibility of generating training data for machine learning algorithms (see Chapter 5). While this latter approach holds great promise, a significant advantage of rule-based PCT remains its interpretability. These rules can be further refined to reflect evolving knowledge and regional particularities.

4.4.2 Strengths

Previous studies suggest that effectiveness of DCT apps is strongly correlated with their level of adoption (Ferretti et al., 2020; Munzert et al., 2021), but significant benefits are nevertheless observable at low adoption (Rodríguez et al., 2021; Bianconi et al., 2021). In our simulations, PCT consistently fares better than BCT in negotiating the epidemic control/mobility restriction trade-off across the range of adoption rates.

Under limited RT-PCR testing, we expect asymptomatic cases to mainly be identified through contact tracing. Sensitivity analyses thus examined the impact of varying the asymptomatic fraction, as well as the infectiousness of asymptomatic agents. Figure 4.5D suggests that an increase in the number of asymptomatic agents in the population requires greater precision to control the epidemic while allowing for encounters to take place. Figure 4.6 shows that increasing the contagiousness of asymptomatic agents results in reduced performance of DCT. Importantly, in the PCT framework, risk messages provide preliminary clues about potentially infected cases even in the absence of test results, thereby generating early warning signals to rapidly caution infectious individuals, and breaking the chain of infections. Both figures show that PCT consistently exhibits a superior trade-off compared to BCT, which relies on testing.

A key advantage of PCT is its ability to integrate individualized data (i.e. symptoms, testing, etc.) in a cost-effective and privacy-preserving manner. Our cost-effectiveness analyses demonstrate that even at low adoption rates, PCT yields better health outcomes and a lower ICER with respect to HQ than BCT. The ICER of PCT with respect to HQ is well below standard willingness-to-pay thresholds (the Canadian threshold adjusted for inflation for new health technologies is $33K per DALY averted in 2020 Canadian dollars (Laupacis et al., 1992); see Appendix C.5 for calculations). Furthermore, for higher adoption rates ($\geq 40\%$), PCT is actually cost-saving compared to BCT. As the adoption rate increases, PCT results in both better health *and* economic outcomes, which leads to the reversal observed in Fig 4.7(a). This reversal corresponds to the decreasing ICERs observed in Fig 4.7(b). PCT dominates BCT by leveraging additional information, and thus overcomes a trade-off often perceived as unavoidable.

4.4.3 Limitations

The urgency of responding to COVID-19 pandemic led to numerous simulators, each with varying levels of fidelity. Due to technological and theoretical limitations, it remains a challenging task to simulate complex behavior of society during pandemic.

Thus, aligning with the call to action on simulators proposed by Squazzoni et al. (2020), we'd like to highlight that our work primarily aims at counterfactual analysis, not future predictions. Our goal is to understand what could have been, offering an alternative perspective to enhance future responses.

The level of detail in our agent-based model allows more realistic individual-level simulation, however the added complexity may also introduce limitations. For example, epidemiological parameters like asymptomatic fraction are inherently difficult to measure, and may change with new variants of the virus. Nevertheless, sensitivity analyses varied parameters we judged as having highly uncertain values. While the ordering in performance of methods remained invariant to these parameters, other sources of bias and uncertainty may be important and should be examined prior to real-world trials.

We opted for a univariate sensitivity analysis for simplicity and because multivariate analysis would be computationally expensive. Furthermore, multi-variate analysis would require input from decision-makers to stress-test appropriate scenarios. Decision-makers might also want to attribute non-equal weights to incidence and mean daily contacts in calculating $\frac{\hat{H}}{\hat{E}}$, given varying health and economic priorities.

We emphasize that PCT has been designed to leverage the agility and scalability of DCT apps while integrating estimation of infectiousness, which is implicitly performed by health agents during manual contact tracing. As a result, we do not consider DCT apps to replace manual tracing; the two approaches can be used complementarily, and their interaction would be an interesting avenue for future study (Wymant et al., 2021). In addition, as in Ferretti et al. (2020); Cencetti et al. (2021), we indirectly accounted for complementary policies (e.g., wearing mask, distancing, etc.) by varying the global mobility factor (see Methods).

Finally, cost-effectiveness analyses used an aggregated TPL calculation, as financial data was not available to allow for a micro-costing approach. Although this may fail to account for how costs vary by context, temporary decreases in productivity at the societal level may still be a concern for decision-makers. To establish the acceptability of our proposal, we used a willingness-to-pay threshold

adjusted for inflation according to the Canadian CPI. These thresholds are not standardized, i.e., they vary across the globe as well as the nature of intervention (e.g., technology and vaccine will have a different threshold). Thus, for a more detailed acceptability analysis we think it will be useful to establish these thresholds across important categories.

4.4.4 Next steps

As we show in Chapter 5, more sophisticated deep learning (DL) predictors trained on the output of COVI-AgentSim show great promise, particularly to overcome issues relating to low app adoption. Improving interpretability of DL methods, and using them to inform future iterations of the rule-based predictor, constitute an important next step. This direction will not only contribute to efforts in responsible AI deployment, but also lead to crucial methodological developments. Moreover, once the PCT app is deployed, incorporating real-world app usage data with the simulated data can be used to learn a more accurate generative model (simulator). This can improve understanding of infection propagation mechanisms across communities, further allowing tailored and efficient responses.

We acknowledge that app adoption can vary among regions or communities, which might induce disproportionate impacts on health and economic outcomes. Both *Rule-based PCT* and any deep learning method can be tailored to reflect such inequities in adoption through inference from time-limited anonymised app data. This direction of research, though challenging, will be necessary to ensure equal distribution of public goods.

Finally, we note that COVID-19 is an overdispersed epidemic (Lau et al., 2020), whereby a few people infect many while most people infect a few. Identifying the source of infection, which is the focus of *backward tracing*, may favour tracing efficiency. The PCT framework can support backward tracing by reserving bits of information in risk messages to indicate the likelihood of contagion; we aim to design such a predictor in future work.

To conclude, we presented a PCT framework to improve upon existing DCT framework. We further designed *Rule-based PCT* algorithm as one instantitation of the PCT framework, and compared its performance against the existing DCT approach. Given the speed and scale at which apps can be deployed as compared to vaccine development and testing, proactive contact tracing apps may prove effective during emerging epidemics and present a valuable complementary strategy to existing NPIs in the presence of vaccine-evading variants.

Acknowledgments

The work in this chapter was funded by: NSERC - The Natural Sciences and Engineereing Research Council of Canada (RGPIN-2019-04822 | nserc-crsng.gc.ca). We thank all members of the broader COVI project (`https://mila.quebec/en/project/covi/`) for their dedication and teamwork; it was a pleasure and honour to work with such a great team. This project could not have been completed without the resources of Compute Canada and Calcul Quebec, in particular the Beluga. I am grateful for Tim Ecott and Prateek Bansal for their support and helpful comments on the draft.

5

Predicting Infectiousness for Proactive Contact Tracing

Contents

About this work

Publication. Bengio et al. (2021), published in ICLR 2021

Peer-reviews. Peer-reviews can be accessed at `https://openreview.net/forum?id=lVgB2FUbzuQ`.

Code. Code is made publicly-available at `https://github.com/mila-iqia/COVI-ML`.

Contributions. My contributions to this project include:

- Identifying and brainstorming crucial challenges in the deployment of deep learning solutions for contact tracing applications
- Incorporating machine learning into the simulation framework for contact tracing
- Developing metrics to evaluate the effectiveness of various contact tracing methods
- Visualizing and presenting project results to stakeholders
- Designing and conducting experiments
- Contributing to the content and writing of the draft, as well as managing the peer-review process.

Note that my involvement in this project was primarily focused on the design aspect of the machine learning model, rather than its actual code. I was responsible for the design of input features and their extraction from the simulator, as well as considering potential neural network architectures. Throughout the process, we encountered various challenges and explored different ideas regarding the training protocol through experimentation. The data collection and actual training of the models were handled by my collaborator.

Talks. This work has been presented at the following venues,

- Probabilistic Risk Awareness for COVID-19, Bank of Canada Sept '20
- Early-Warning Signals of COVID-19, CIFAR/ELLIS Workshop, Oct '20
- ICLR 2021 (Spotlight talk)
- 10th UK–wide Doctoral Researcher Award, Sept '21
- AI UK Lightning Talk, March '23

Other information. Note that this chapter was published in a machine learning conference, which has shorter review periods compared to journals. There was a two-year gap between the publication of this chapter (published in 2021) and the previous chapter (published in 2023). During this time, we refined our terms and presentation of results for better scientific communication. As a result, certain portions of the following chapter have been updated to align with these updates.

Abstract

The COVID-19 pandemic spread rapidly worldwide, overwhelming manual contact tracing in many countries and resulting in widespread lockdowns for emergency containment. Large-scale **digital contact tracing (DCT)**[1] emerged as a potential solution to resume economic and social activity while minimizing spread of the virus. Various DCT methods were proposed, each making trade-offs between privacy, mobility restrictions, and public health. The most common approach, **binary contact tracing (BCT)**, models infection as a binary event, informed only by an individual's test results, with corresponding binary recommendations that either all or none of the individual's contacts quarantine. BCT ignores the inherent uncertainty in contacts and the infection process, which could be used to tailor messaging to high-risk individuals, and prompt proactive testing or earlier warnings. It also does not make use of observations such as symptoms or pre-existing medical

[1]All **bolded** terms are defined in the Glossary; Appendix D.1.

conditions, which could be used to make more accurate infectiousness predictions. In this chapter, we use a COVID-19 epidemiological simulator (see Appendix C.2) to develop and test deep learning methods that can be deployed to a smartphone to locally and proactively predict an individual's infectiousness (risk of infecting others) based on their contact history and other information, while respecting strong privacy constraints. Predictions are used to provide personalized recommendations to the individual via an app, as well as to send anonymized messages to the individual's contacts, who use this information to better predict their own infectiousness, under the framework of **proactive contact tracing (PCT)** introduced in Chapter 4. Similarly to other works, we find that compared to no tracing, all DCT methods tested are able to reduce spread of the disease and thus save lives, even at low adoption rates, strongly supporting a role for DCT methods in managing the pandemic. Further, we find a deep-learning based PCT method which improves over the BCT and Rule-based PCT methods for equivalent average mobility, suggesting PCT could help in safe re-opening of the economy.

5.1 Introduction

Until pharmaceutical interventions such as a vaccine became available, control of the COVID-19 pandemic relied on nonpharmaceutical interventions such as lockdown and social distancing. While these interventions were successful in limiting spread of the disease in the short term, the restrictive measures had important negative social, mental health, and economic impacts. **Digital contact tracing (DCT)**, a technique to track the spread of the virus among individuals in a population using smartphones, is an attractive potential solution to help reduce growth in the number of cases and thereby allow more economic and social activities to resume while keeping the number of cases low.

Most deployed DCT solutions use **binary contact tracing (BCT)**, which sends a quarantine recommendation to all recent contacts of a person after a positive test result. While BCT is simple and fast to deploy, and most importantly can help curb spread of the disease (Abueg et al., 2020), epidemiological simulations

by Hinch et al. (2020) suggest that using only one bit of information about the infection status can lead to quarantining of many healthy individuals while failing to quarantine infectious individuals. Relying only on positive test results as a trigger is also inefficient for a number of reasons: (i) Tests have high false positive rates (Li et al., 2020); (ii) Tests are administered late, only after symptoms appear, leaving the asymptomatic population, estimated 20%-30% of cases (Gandhi et al., 2020), likely untested; (iii) It is estimated that infectiousness is highest *before* symptoms appear, well before someone would get a test (Heneghan et al., 2020), thus allowing them to infect others way before kick starting the tracing process. (iv) Results typically take at least 1-2 days, and (v) In many places, tests are in limited supply.

Recognizing the issues with test-based tracing, in Chapter 4, we proposed a rule-based PCT algorithm leveraging other input clues potentially available on a smartphone (e.g., symptoms, pre-existing medical conditions). Probabilistic (non-binary) approaches, using variants of belief propagation in graphical models, e.g., (Baker et al., 2021; Satorras and Welling, 2021; Briers et al., 2020), could also make use of features other than test results to improve over BCT, although these approaches rely on knowing the social graph, either centrally or via distributed exchanges between nodes. The latter solution may require many bits exchanged between nodes (for precise probability distributions), which is challenging both in terms of privacy and bandwidth. In contrast, the PCT framework uses a predictor trained to output *proactive* (before current-day) estimates of expected infectiousness (i.e. risk of having infected others in the past and of infecting them in the future). The challenges of privacy and bandwidth motivated our particular form of **distributed inference** where we pretrain the predictor offline and do not assume that the messages exchanged are probability distributions, but instead just informative inputs to the node-level predictor of infectiousness.

We use a COVI-AgentSim, a COVID-19 agent-based model (see Chapter 4 and Appendix C.2) to compare different variants of PCT to other contact tracing methods under a wide variety of conditions. We develop deep learning predictors for PCT in concert with a professional app-development company, ensuring inference

models are appropriate for legacy smartphones. By leveraging the rich individual-level data produced by COVI-AgentSim to train predictors offline, we are able to perform individual-level infectiousness predictions locally to the smartphone, with sensitive personal data never required to leave the device. We find deep learning based methods to be consistently able to reduce the spread of the disease more effectively, at lower cost to mobility, and at lower adoption rates than other predictors. These results suggest that deep learning enabled PCT could be deployed in a smartphone app to help produce a better trade-off between the spread of the virus and the economic cost of mobility constraints than other DCT methods, while enforcing strong privacy constraints.

5.1.1 Summary of technical contributions

- In order to perform distributed inference with deep learning models, we develop an architectural scaffold whose core is any set-based neural network. We embed two recently proposed networks, namely Deep Sets (Zaheer et al., 2017) and Set Transformers (Lee et al., 2019) and evaluate the resulting models via the COVI-AgentSim testbed (Gupta et al., 2020b) (see Section 5.3.1).

- To our knowledge the combination of techniques in this pipeline is entirely novel, and of potential interest in other settings where privacy, safety, and domain shift are of concern. Our training pipeline consists of training an ML infectiousness predictor on the domain-randomized output of an agent-based epidemiological model, in several loops of retraining to mitigate issues with (i) non-stationarity and (ii) distributional shift due to predictions made by one phone influencing the input for the predictions of other phones (see Section 5.3.2).

- To our knowledge this is the first work to apply and benchmark a deep learning approach for probabilistic contact tracing and infectiousness risk assessment. We find such models are able to leverage weak signals and patterns in noisy,

heterogeneous data to better estimate infectiousness compared to binary contact tracing and rule-based methods (see Section 5.4).

5.2 Proactive Contact Tracing

To recap, **Proactive contact tracing (PCT)**, introduced in Chapter 4, is an approach to digital contact tracing which leverages the rich suite of features potentially available on a smartphone (including information about symptoms, preexisting conditions, age and lifestyle habits if willingly reported) to compute proactive estimates of an individual's expected infectiousness. These estimates are used to (a) provide an individualized recommendation and (b) propagate a graded risk message to other people who have been in contact with that individual (see Fig. 5.1). This stands in contrast with existing approaches for contact tracing, which are either binary (recommending all-or-nothing quarantine to contacts), or require centralized storage of the contact graph or other transfers of information which are incompatible with privacy constraints in many societies. Further, the estimator runs locally on the individual's device, such that any sensitive information volunteered does not need to leave the device.

In Section 5.2.1, we formally define the general problem PCT solves. In Section 5.2.2, we describe how privacy considerations inform and shape the design of the proposed framework and implementation. Finally, in Section 5.3.1, we introduce deep-learning based estimators of expected infectiousness, which we show in Section 5.4 to outperform DCT baselines by a large margin.

5.2.1 Problem Setup

We wish to estimate **infectiousness** $y_i^{d'}$ of an agent i on day d', given access to *locally observable information* $\vec{O}_i^d$ now on day $d \geq d'$ and over the past d_{max} days ($d' \geq d - d_{max}$). Some of the information available on day d is static, including reported age, sex, pre-existing conditions, and lifestyle habits (for example, smoking), denoted g_i, the health profile. Other information is measured each day: the health

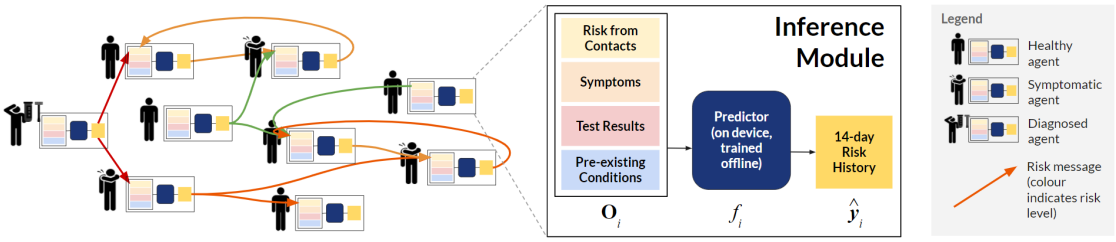

Figure 5.1: Proactive contact tracing overview. Diagram showing **Left:** the propagation of anonymized, graded (non-binary) risk messages between users and **Inset:** overview of the inference module deployed to each user's phone. The inference module for agent i takes in observables $\mathbf{O}_i$, and uses a pretrained predictor f to estimate that agent's risk (expected infectiousness) for each of the last 14 days. Anonymized selected elements of this risk vector are sent as messages to appropriate contacts, allowing them to proactively update their own estimate of expected infectiousness.

status h_i^d of reported symptoms and known test results. Finally, $\vec{O}_i^d$ also includes information about encounters in the last d_{max} days, grouped in e_i^d for day d. Thus:

$$\vec{O}_i^d = (g_i, h_i^d, e_i^d, h_i^{d-1}, e_i^{d-1}, \dots h_i^{d-d_{max}}, e_i^{d-d_{max}}) \tag{5.1}$$

The information $e_i^{d'}$ about the encounters from day d' is subject to privacy constraints detailed in Section 5.2.2 but provides indications about the estimated infectiousness of the persons encountered at d', given the last available information by these contacts as of day d, hence these contacts try to estimate their past infectiousness a posteriori. Our goal is thus to model the *history* of the agent's infectiousness (in the last d_{max} days), which is what enables the recommendations of PCT to be *proactive* and makes it possible for an infected asymptomatic person to receive a warning from their contact even before they develop symptoms, because their contact obtained sufficient evidence that they were contagious on the day of their encounter. Formally, we wish to model $P_\theta(\vec{y}_i^d|\vec{O}_i^d)$, where $\vec{y}_i^d = (y_i^d, y_i^{d-1}, \dots, y_i^{d-d_{max}})$ is the vector of present and past infectiousness of agent i and θ specifies the parameters of the predictive model. In our experiments we only estimate conditional expectations with a predictor f_θ, with $\vec{\hat{y}}_i^d = (\hat{y}_i^d, ..., \hat{y}_i^{d-d_{max}}) = f_\theta(\vec{O}_i^d)$ an estimate of the conditional expected per-day present and past infectiousness $\mathbb{E}_{P_\theta}[\vec{y}_i^d|\vec{O}_i^d]$.

The predicted expected values are used in two ways. First, they are used to generate messages transmitted on day d to contacts involved in encounters on

day $d' \in (d - d_{max}, d)$. These messages contain the estimates $\hat{y}_i^{d'}$ of the expected infectiousness of i at day d', quantized to 4 bits for privacy reasons discussed in Section 5.2.2. Second, the prediction for today $\hat{y}_i^d$ is also used to form a discrete recommendation level $\zeta_i^d \in \{0, 1, ..., 4\}$ regarding the behavior of agent i at day d via a recommendation mapping ψ, i.e. $\zeta_i^d = \psi(\hat{y}_i^d)$. At $\zeta_i^d = 0$ agent i is not subjected to any restrictions, $\zeta_i^d = 1$ is baseline restrictions of a post-lockdown scenario (as in summer 2020 in many countries), $\zeta_i^d = 4$ is full quarantine (also the behavior recommended for contacts of positively diagnosed agents under BCT), and intermediate levels interpolate between levels 1 and 4.

Here we make two important observations about contact tracing: (1) There is a trade-off between decelerating spread of disease, measured by the **reproduction number** R, or as number of cases,[2] and minimizing the degree of restriction on agents, e.g., measured by the average number of contacts between agents. Managing this tradeoff is a social choice which involves not just epidemiology but also economics, politics, and the particular weight different people and nations may put on individual freedoms, economic productivity and public health. The purpose of PCT is to improve the corresponding **Pareto frontier**. A solution which performs well on this problem will encode a policy that contains the infection while applying minimal restrictions on healthy individuals, but it may be that different methods are more appropriate depending on where society stands on that tradeoff. (2) A significant challenge comes from the feedback loop between agents: observables $\vec{O}_i^d$ of agent i depend on the predicted infectiousness histories and the pattern of contacts generated by the behavior ζ_j of *other agents* j. This is compounded by privacy restrictions that prevent us from knowing which agent sent which message; we discuss our proposed solution in the following section.

5.2.2 Privacy-preserving PCT

One of the primary concerns with the transmission and centralization of contact information is that someone with malicious intent could identify and track people.

[2]Note that the the number of cases is highly non-stationary; it grows exponentially over time, even where R, which is in the exponent, is constant.

Even small amounts of personal data could allow someone to infer the identities of individuals by cross-referencing with other sources, a process called **re-identification** (El Emam et al., 2011). Minimizing the number of bits being transmitted and avoiding information that makes it easy to triangulate people is a protection against **big brother attacks**, where a central authority with access to the data can abuse its power, as well as against **little brother attacks**, where malicious individuals (e.g., someone you encounter) could use your information against you. Little brother attacks include **vigilante attacks**: harassment, violence, hate crimes, or stigmatization against individuals which could occur if infection status was revealed to others. Sadly, there have been a number of such attacks related to COVID-19 (Russell, 2020). To address this, PCT operates with (1) no central storage of the contact graph; (2) de-identification (Sweeney, 2002) and encryption of all data leaving the phones; (3) informed and optional consent to share information (only for the purpose of improving the predictor); and (4) distributed inference which can achieve accurate predictions without the need for a central authority. Current solutions (Woodhams, 2020) share many of these goals, but are restricted to only binary CT. Our solution achieves these goals while providing individual-level graded recommendations.

Each app records **contacts** which are defined as being an encounter which lasts at least 15 minutes with a proximity under 2 meters. At regular intervals (e.g., every six hours), each application processes contacts, predicts infectiousness for days d to d_{max}, and may update its current recommended behavior. If the newly predicted infectiousness history differs from the previous prediction on some day (e.g., typically because the agent enters new symptoms, receives a negative or positive test result, or receives a significantly updated infectiousness estimate), then the app creates and sends small update messages to all relevant contacts. These heavily quantized messages reduce the network bandwidth required for message passing as well as provide additional privacy to the individual. We follow the communication and security protocol for these messages introduced by Alsdurf et al. (2020). We note that a naive method would tend to *over-estimate* infectiousness in

these conditions, because repeated encounters with the same person (a very common situation, for example people living in the same household) should carry a *lower* predicted infectiousness than the same number of encounters with different people.

5.3 Methodology for Infectiousness Estimation

5.3.1 Distributed Inference

Distributed inference networks seek to estimate marginals by passing messages between nodes of the graph to make predictions which are consistent with each other (and with the local evidence available at each node). Because messages in our framework are highly constrained by privacy concerns in that we are prevented from passing distributions or continuous values between identifiable nodes, typical distributed inference approaches (e.g., loopy belief propagation (Murphy et al., 1999), variational message passing (Winn and Bishop, 2005), or expectation propagation (Minka, 2001)) are not readily applicable. We instead propose an approach to distributed inference which uses a trained ML predictor for estimating these marginals (or the corresponding expectations). The predictor f is trained offline on simulated data to predict expected infectiousness from the locally available information $\vec{O}_i^d$ for agent i on day d. The choice of ML predictor is informed by several factors. First and foremost, we expect the input e_i^d (a component of $\mathbf{O}_i^d$) to be variable-sized, because we do not know *a priori* how many contacts an individual will have on any day. Further, privacy constraints dictate that the contacts within the day cannot be temporally ordered, implying that the quantity e_i^d is set-valued. Second, given that we are training the predictor on a large amount of domain-randomized data from many epidemiological scenarios (see Section 5.3.2), we require the architecture to be sufficiently expressive. Finally, we require the predictor to be easily deployable on edge devices like legacy smartphones.

To these ends, we construct a general architectural scaffold in which any neural network that maps between sets may be used. In this work, we experiment with Set Transformers (Lee et al., 2019) and a variant of Deep Sets (Zaheer et al., 2017). The former is a variant of Transformers (Vaswani et al., 2017), which model pairwise

interactions between all set elements via multi-head dot-product attention and can therefore be quite expressive, but scales quadratically with elements. The latter relies on iterative max-pooling of features along the elements of the set followed by a broadcasting of the aggregated features, and scales linearly with the number of set elements (see Figure 5.2).

In both cases, we first compute two categories of embeddings: per-day and per-encounter. Per-day, we have embedding MLP modules ϕ_h of health status, ϕ_g for the health profile, and a linear map $\phi_{\delta d}$ for the day-offset $\delta d = d - d'$. Per-encounter, the cluster matrix e_i^d can be expressed as the set $\{(r_{j\to i}^d, n_{j\to i}^d)\}_{j\in K_i^d}$ where j is in the set K_i^d of putative anonymous persons encountered by i at d, $r_{j\to i}$ is the risk level sent from j to i and $n_{j\to i}$ is the number of repeated encounters between i and j at d. Accordingly, we define the risk level embedding function $\phi_e^{(r)}$ and $\phi_e^{(n)}$ for the number of repeated encounters. $\phi_e^{(r)}$ is parameterized by an embedding matrix, while $\phi_e^{(n)}$ by the vector

$$(\phi_e^{(n)}(n))_{2i} = \sin\left({}^{n}/_{10000^i}\right) \text{ and } (\phi_e^{(n)}(n))_{2i+1} = \cos\left({}^{n}/_{10000^i}\right) \tag{5.2}$$

which resembles the positional encoding in Vaswani et al. (2017) and counteracts the spectral bias of downstream MLPs (Rahaman et al., 2019; Mildenhall et al., 2020). We tried alternative encoding schemes like thermometer encodings (Buckman et al., 2018), but they did not perform better in our experiments. Next, we join the per-day and per-encounter embeddings to obtain $\mathcal{D}_i^d$ and $\mathcal{E}_i^d$ respectively. Where $\otimes$ is the concatenation operation, we have:

$$\mathcal{D}_i^d = \{\phi_h(h_i^{d'}) \otimes \phi_g(g_i) \otimes \phi_{\delta d}(d' - d) \,|\, d' \in \{d, ..., d - d_{max}\}\} \tag{5.3}$$

$$\mathcal{E}_i^d = \{\phi_e^{(r)}(r_{j\to i}^{d'}) \otimes \phi_e^{(n)}(n_{j\to i}^{d'}) \otimes \phi_g(g_i) \otimes \phi_{\delta d}(d' - d) \,|\, j \in K_i^{d'}, \forall\, d' \in \{d, ..., d - d_{max}\}\}$$

The union of $\mathcal{D}_i^d$ and $\mathcal{E}_i^d$ forms the input to a set neural network f_S, that predicts infectiousness:

$$\vec{\hat{y}}_i^d = f_S(\mathcal{O}_i^d) \text{ where } \mathcal{O}_i^d = \mathcal{D}_i^d \cup \mathcal{E}_i^d \text{ and } \vec{\hat{y}}_i^d \in \mathbb{R}_+^{d_{max}} \tag{5.4}$$

The trunk of both models – deep-set (DS-PCT) and set-transformer (ST-PCT) – is a sequence of five set processing blocks (see Figure 5.2). A subset of outputs

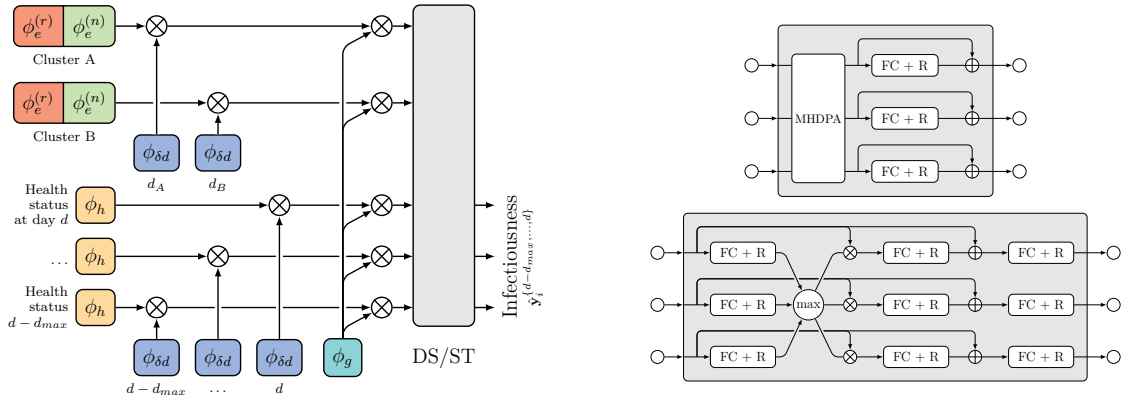

Figure 5.2: PCT model architecture. Diagram showing **Left**: The embedding network combining pre-existing conditions, day offsets, risk message clusters, and symptoms information to be fed into a stack of 5 either Deep-Set (DS) or Set Transformer (ST) blocks. **Right-top**: (ST) self-attention block featuring Multi-Head Dot Product Attention (MHDPA), Fully-Connected layers with ReLU (FC + R) and residual connections. **Right-bottom:** (DS) set processing block. Here, the $\otimes$ denotes concatenation and the $\oplus$ addition operation.

from these blocks (corresponding to $\mathcal{D}_i^d$) is processed by a final MLP to yield $\vec{\hat{y}}_i^d$. As a training objective for agent i, we minimize the Mean Squared Error (MSE) between $\vec{y}_i^d$ (which is generated by the simulator) and the prediction $\vec{\hat{y}}_i^d$. We treat each agent i as a sample in batch to obtain the sample loss $L_i = \text{MSE}(\vec{y}_i^d, \vec{\hat{y}}_i^d) = \frac{1}{d_{max}} \sum_{d'=d-d_{max}}^{d} (y_i^{d'} - \hat{y}_i^{d'})^2$. The net loss is the sum of L_i over all agents i.

5.3.2 Training Procedure

Unlike existing contact tracing methods like BCT and the NHSx contact tracing app (Briers et al., 2020) that rely on simple and hand-designed heuristics to estimate the risk of infection, our core hypothesis is that methods using a machine-learning based predictor can learn from patterns in rich but noisy signals that might be available locally on a smartphone. In order to test this hypothesis prior to deployment in the real world, we require a simulator built with the objective to serve as a testbed for app-based contact tracing methods. We used COVI-AgentSim (see Appendix C.2), which features an agent specific virology model together with realistic agent behavior, mobility, contact and app-usage patterns. With the simulator, we generate large datasets of $\mathcal{O}(10^7)$ samples comprising the input and target variables defined in

Sections 5.2.1 and 5.3.1. Most importantly for our prediction task, this includes a continuous-valued infectiousness parameter as target for each agent.

We use 240 runs of the simulator to generate this dataset, where each simulation is configured with parameters randomly sampled from selected intervals (See Appendix D.3.2). These parameters include the adoption rate of the contact tracing app, initial fraction of exposed agents in the population, the likelihood of agents ignoring the recommendations, and strength of social distancing and mobility restriction measures. There are two reasons for sampling over parameter ranges: First, the intervals these parameters are sampled from reflect both the intrinsic uncertainty in the parameters, as well as the practical setting and limitations of an app-based contact tracing method (e.g., we only care about realistic levels of app usage). Second, randomly sampling these key parameters significantly improves the diversity of the dataset, which in turn yields predictors that can be more robust when deployed in the real world. The overall technique resembles that of **domain-randomization** (Tobin et al., 2017; Sadeghi and Levine, 2016) in the Sim2Real literature, where its efficacy is well studied for transfer from RGB images to robotic control, e.g., (Chebotar et al., 2019; Andrychowicz et al., 2020).

We use 200 runs for training and the remaining 40 for validation (full training and other reproducibility details are in Appendix D.3). The model with the best validation score is selected for *online* evaluation, wherein we embed it in the simulator to measure the reduction in the R as a function of the average number of contacts per agent per day. While the evaluation protocol is described in Section 5.4, we now discuss how we mitigate a fundamental challenge that we share with offline reinforcement learning methods, that of **auto-induced distribution shift** when the model is used in the simulation loop (Levine et al., 2020; Krueger et al., 2020).

To understand the problem, consider the case where the model is trained on simulation runs where PCT is driven by ground-truth infectiousness values, i.e. an *oracle* predictor. The predictions made by an oracle will in general differ from ones made by a trained model, as will resulting dynamics of spread of disease. This leads to a distribution over contacts and epidemiological scenarios that the model

has not encountered during training. For example, oracle-driven PCT might even be successful in eliminating the disease in its early phases – a scenario unlikely to occur in model-driven simulations. For similar reasons, oracle-driven simulations will also be less diverse than model-driven ones. To mitigate these issues, we adopt the following strategy: First, we generate an initial dataset with simulations driven by a *noisy-oracle*, i.e. we add multiplicative and additive noise to the ground-truth infectiousness to partially emulate the output distribution resulting from trained models. The corresponding noise levels are subject to domain-randomization (as described above), resulting in a dataset with some diversity in epidemiological scenarios and contact patterns. Having trained the predictor on this dataset (until early-stopped), we generate another dataset from simulations driven by the thus-far trained predictor, in place of the noisy-oracle. We then fine-tune the predictor on the new dataset (until early stopped) to obtain the final predictor. This process can be repeated multiple times, in what we call **iterative retraining**. We find three steps yields a good trade-off between performance and compute requirement.

5.4 Experiments

We evaluate the proposed PCT methods, **ST-PCT** and **DS-PCT**, and benchmark them against: BCT; a rule-based PCT, we call it **Heuristic**, method proposed in Chapter 4 and a baseline **No Tracing (NT)** scenario which corresponds to recommendation level 1 (some social distancing). Since Chapter 4 already establishes the superiority of Rule-based PCT over BCT and Household quarantine, our objective in this chapter to is to understand how much improvement can we expect from deep learning based PCT over Rule-based PCT and BCT.

Experiment 1. In Figure 5.3, we plot the Pareto frontier between spread of disease (R) and the amount of restriction imposed. To traverse the frontier at 60% adoption rate, we sweep through multiple values of the simulator's **global mobility scaling factor**, a parameter which controls the strength of social distancing and mobility restriction measures. Each method uses 3000 agents for 50 days with 12 random

seeds, and we plot the resulting number of contacts per day per human. We fit a Gaussian Process Regressor to the simulation outcomes and highlight the average reduction in R obtained in the vicinity of $R \approx 1$, finding DS-PCT and ST-PCT reduce R to below 1 at a much lower cost to mobility on average. For subsequent plots, we select the number of contacts per day per human such that the no-tracing baseline yields a realistic post-lockdown R of around 1.2 (Brisson et al., 2020)[3].

Experiment 2. Figure 5.4 compares various DCT methods in terms of their cumulative cases and their fraction of **false quarantine**: the number of healthy agents the method wrongly recommends to quarantine. Again, all DCT methods have a clear advantage over the no-tracing baseline, while the number of agents wrongly recommended quarantine is much lower for ML-enabled PCT. [4]

Experiment 3. In Figure 5.5 **Left** we plot the bootstrapped distribution of mean R for different DCT methods. Recall R is an estimate of how many other agents an infectious agent will infects, i.e. even a numerically small improvement in R could yield an exponential improvement in the number of cases. We find that both PCT methods yield a clear improvement over BCT and the rule-based heuristic, all of which significantly improve over the no-tracing baseline. We hypothesize this is because deep neural networks are better able to capture the non-linear relationship between features available on the phone, interaction patterns between agents, and individual infectiousness.

Experiment 4. In Figure 5.5 **Right** we investigate the effect of **iterative retraining** on the machine-learning based methods, DS-PCT and ST-PCT. We evaluate all 3 iterations of each method in the same experimental setup as in Figure 5.5 and find that DS-PCT benefits from the 3 iterations, while ST-PCT saturates

[3]To estimate this number, we use the GP regression fit in Figure 5.3 and consider the x value for which the mean of the NT process is at 1.2. We find an estimate at 5.61, and select only the runs yielding effective number of contacts that lie within the interval $(5.61 - 0.5, 5.61 + 0.5)$.

[4]Note that the baseline no tracing method also has false quarantines, because household members of an infected individual are also recommended quarantine, irrespective of whether they are infected.

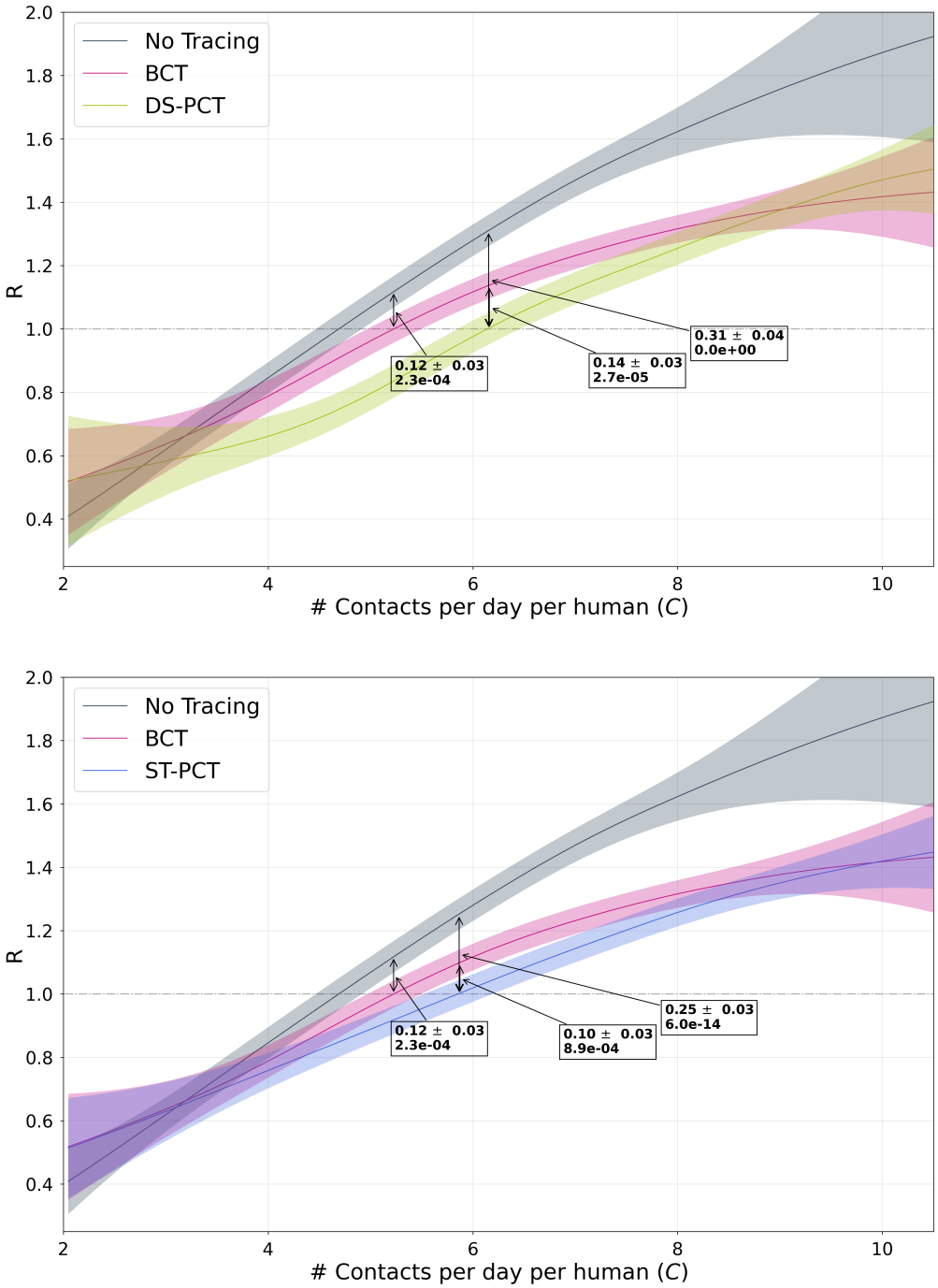

Figure 5.3: Pareto frontier of mobility and disease spread: reproduction number R as a function of mobility. In boxes, average difference +/- standard error, p-value under null hypothesis of no difference. Note that small differences in R over time produce large changes to the number of cases. **Gist:** All methods span a wide range of R values in their Pareto frontier, including values for the No Tracing scenario which have R below 1, achieved by imposing strong restrictions on mobility (e.g., a lockdown). However DCT methods are able to reduce R at much lower cost to average mobility. Compared to NT, BCT has a 12% advantage in R, DS-PCT (**left**) 31% and ST-PCT 25% (**right**).

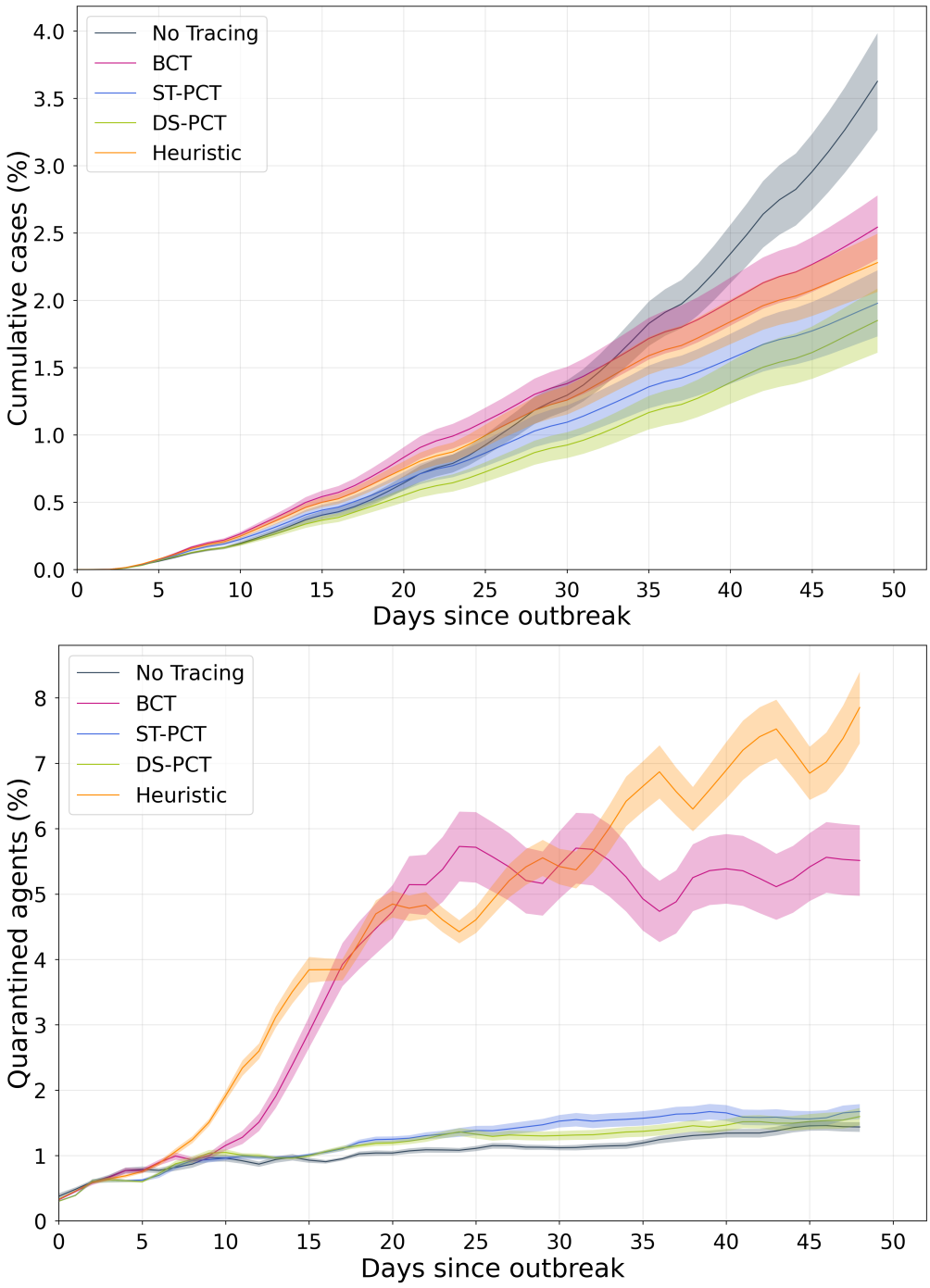

Figure 5.4: Left: Cumulative case counts for each method, 60% adoption, 50 days, with all runs normalized to 5.61 ± 0.5 effective contacts per day per agent. **Right:** Mobility restriction for the same experiments (fraction of quarantines). **Gist:** For a similar number of cumulative cases to other DCT methods (left), PCT methods impose very little mobility restriction (right), close to NT. Note: Part of the axis values have been omitted for clarity.

at the second iteration. We hypothesize that this is a form of overfitting, given that the set transformer (ST) models all-to-all interactions and is therefore more expressive than deep-sets (DS).

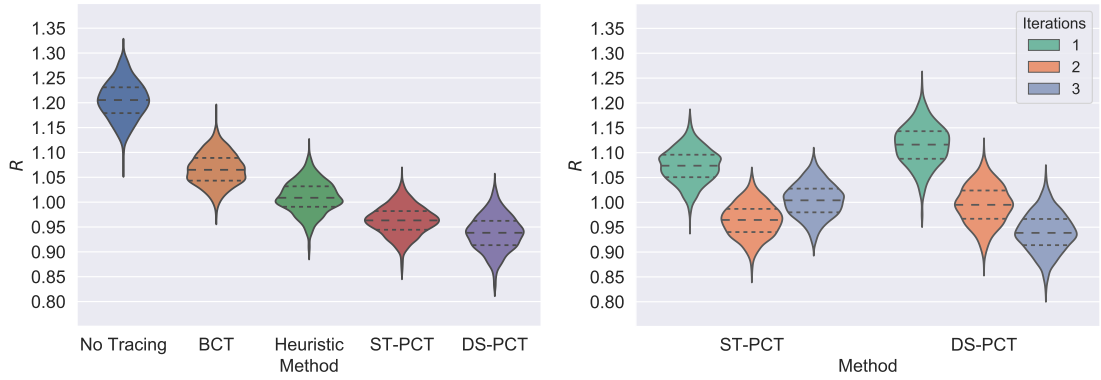

Figure 5.5: Method Comparison and Retraining. **Left:** Distribution (median and quartiles) of R at 60% adoption where the ML methods outperform the Heuristic (Rule-based PCT) and BCT methods proposed in Chapter 4. **Right:** Iterative re-training significantly improves model performance. Upon the second re-training, however, it seems that ST-PCT begins to overfit. Note: Part of the axis values have been omitted for clarity.

Experiment 5. In Figure 5.6, we analyze the sensitivity of the various methods to adoption rate, which measures what percent of the population actively uses the CT app. Adoption rate is an important parameter for DCT methods, as it directly determines the effectiveness of an app. We visualize the effect of varying the adoption rate on the reproduction number R. Similarly to prior work (Abueg et al., 2020), we find that all DCT methods improve over the no-tracing baseline even at low adoptions and PCT methods dominate at all levels of adoption.

5.5 Conclusion

Our results demonstrate the potential benefit of digital contact tracing approaches for saving lives while reducing mobility restrictions and preserving privacy, independently confirming previous reports (Ferretti et al., 2020; Abueg et al., 2020). Of all methods in our study, we find that deep learning based PCT provides the best trade-off between restrictions on mobility and reducing the spread of

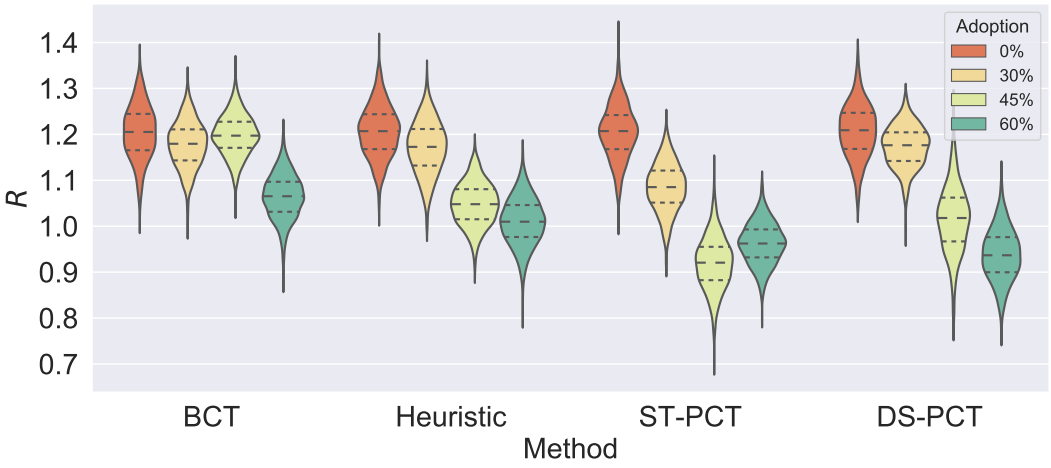

Figure 5.6: Adoption rate comparison. We compare all methods for adoption rates between 0% (NT) and 60%. **Gist:** All methods are able to improve over NT, even at low adoption rates. At 30% and 45%, ST-PCT performs the best by a relatively wide margin while DS-PCT outperforms it at 60%. Note: Part of the axis values have been omitted for clarity.

disease under a range of settings, making it a potentially powerful tool for saving lives in a safe deconfinement.

This area of research holds many interesting avenues of future work in machine learning, including: (1) The comparison of methods for fitting parameters of the epidemiological simulator data using spatial information (not done here to avoid the need for GPS), (2) Using reinforcement learning to learn (rather than expert hand-tune) the mapping from estimated infectiousness and individual-level features (number of contacts, age, etc.), and thereby target desirable outcomes for heterogeneous populations.

Accurate and practical models for contact tracing are only a small part of the non-pharmaceutical efforts against the pandemic, a response to which is a complex venture necessitating cooperation between public health, government, individual citizens, and scientists of many kinds – epidemiologists, sociologists, behavioral psychologists, virologists, and machine learning researchers, among many others. We hope this work can play a role in fostering this necessary collaboration.

Acknowledgements

We thank all members of the broader COVI project `https://covicanada.org` for their dedication and teamwork; it was a pleasure and honour to work with such a great team. We also thank the Mila and Empirical Inference communities, in particular Xavier Bouthillier, Vincent Mai, Gabriele Prato, and Georgios Arvanitidis, for detailed and helpful feedback on early drafts. The authors gratefully acknowledge the following funding sources: NSERC, IVADO, CIFAR. This project could not have been completed without the resources of MPI-IS cluster, Compute Canada & Calcul Quebec, in particular the Beluga cluster. In gratitude we have donated to a project to help understand and protect the St.Lawrence beluga whales for whom the cluster is named `https://baleinesendirect.org/`

6
Summary and Extensions

Contents

6.1 Summary

In this thesis, we looked at two applications of imitation learning:

- **Learning to branch**: MILP solvers use the B&B framework to iteratively divide the MILP into smaller subproblems to implicitly enumerate all feasible solutions smartly. One of the critical components of the B&B framework is deciding which variable to branch on. A gold standard for this decision is a lookahead greedy strategy called strong branching, which is, in its full form, computationally too expensive to be of any practical relevance. Before our work, an imitation learning framework had been proposed to mimic the strong branching heuristic. We identified and addressed the various shortcomings of that framework.

- **Contact Tracing**: During the COVID-19 pandemic, we saw a massive interest in innovating Digital Contact Tracing apps. The adopted apps had a simple rule-based risk estimation model, thereby not leveraging sophisticated risk estimation modeling possible if we consider the rich information that can be available through the users. Quite obviously, there are privacy concerns in using such user information. Thus, we proposed the Proactive Contact Tracing framework that enables risk modeling while preserving privacy. In addition, we proposed an imitation learning framework to design a neural network-based risk estimation model.

Before discussing possible extensions in future research, we summarize the results of each chapter below.

In **Chapter 2**, we experimented with a wide range of neural network architectures, spanning the expressivity and computational requirements. In doing so, we found that the architecture that uses GNNs at the root node (original MILP) to modulate the outputs of a vanilla feedforward network at all other nodes is better suited to imitate the strong branching heuristic with a relatively inexpensive (computationally) architecture. This computational efficiency is achieved by giving away superior

expressivity, and therefore imitation accuracy, of GNNs. This advantage of the hybrid models shows up when these models are deployed on CPU-only machines. Due to the low computational requirements, hybrid models exhibit faster running times compared to GNNs on CPU-only machines.

To enhance the generalization performance of these hybrid models, we looked at training protocols such as: Depth-based penalty for the observations to ensure reduced drift in the data distribution; Knowledge distillation from expert GNNs to help in training the hybrid models; Auxiliary tasks to enforce the maximum separation in representations that are used to modulate the output of feedforward networks.

In **Chapter 3**, we discovered a key characteristic of the strong branching heuristic, which we call *lookback phenomenon*. Specifically, we found that often in a B&B tree, the parent nodes' second best choice is the child nodes' best choice. This phenomenon has been shown to occur in both real-world problems and synthetically generated instances. Thus, we proposed a new loss function for the imitation learning framework for learning to branch. Notably, we proposed a second-best epsilon-smoothed target in place of the initially proposed one-hot encoded target in the cross-entropy loss. We also proposed a parent-as-target regularizer, a cross-entropy loss between the outputs of the parent node and the child node. This term encourages the GNNs to output parent logits that follow a similar ranking as the child logits whenever the lookback phenomenon occurs.

Finally, and more importantly, due to the highly practical nature of the research contributions, it is necessary to report performance metrics based on how they can be used in industry. Thus, we proposed a methodology to incorporate hard-to-formulate practitioners' objectives in the models' hyperparameter selection. We showed that the models trained on the proposed imitation learning framework outperform the SOTA GNNs.

In **Chapter 4**, we fleshed out a digital contact tracing framework designed to use rich information sources privately and enable sophisticated risk estimation modelling. The Proactive Contact Tracing framework works through smartphone applications, wherein the users exchange a communication key with their nearby

contacts. This key lets the users notify their past contacts at a later date if they are tested positive or if their risk levels change. A user's risk level is computed based on the risk levels of past contacts and the information entered by the user, such as preexisting conditions, symptoms or test results.

The risk levels summarise users' health (e.g., a level of 0 means they are perfectly healthy). Hence, a higher number of categories positively correlates with the amount of information sent across the users. However, the more categories, the more the requirement on the communication network, thereby deterring the users from using the app. In our work, we have let these risk levels be one of sixteen categories, which can be encoded using 4 bits. But one can design the system using just two bits, thereby categorizing users' risk levels into one of four levels. These risk levels are converted at the final stage into one of four recommendation levels with clear behavioral directives.

Finally, we implemented an agent-based model to simulate the spread of the COVID-19 virus. This simulator gave us the platform to design various risk estimation models. As a result, we proposed a rule-based PCT model that uses heuristic rules based on the observations of epidemiologists to estimate user risk. Through extensive sensitivity analysis, we showed that rule-based PCT exhibits a superior trade-off over the existing DCT apps that recommend binary directives of quarantine or not. In **Chapter 5**, we focused on designing a neural network-based risk estimation model for the PCT framework proposed in Chapter 4. This is in contrast to the heuristic-based risk estimation model that most of the existing apps have. Given the unique representation learning capability of neural networks, the rich and noisy data available in the app can be efficiently mapped to the required outputs. Thus, this work is the first attempt at using deep learning techniques to build a risk estimation model.

We use the simulator designed in Chapter 4, to propose an imitation learning framework. Specifically, we collect observations about the ground truth viral load and the information the user feeds to the app. Various designs of neural network architecture as well as the training protocols are considered to address the issues of

generalization to the real world. We used domain randomization to collect data across a wide range of simulator parameters, thereby making the model robust to the assumptions baked in the simulator, which may not carry over to the real world. To make the models robust to the distribution shift encountered after the deployment of the model, we resorted to iterative training in which we would collect the observations by deploying the trained model and do so for a few iterations until the performance stabilizes.

Finally, we showed that compared to the rule-based PCT, the deep learning-based PCT could better negotiate the trade-off between the population restrictions and the spread of the disease.

6.2 Discussion

We now turn to a critical discussion of the different methods presented in this thesis, in hindsight and in light of work that has appeared since their publication.

6.2.1 Learning to Branch

Hybrid Models for Learning to Branch. In recent years, we have seen considerable promise in attention models. It has also been known that the attention models generalize GNNs, and as such, GNNs can be replaced by attention models with modulated attention scores to reflect edge features (Maziarka et al., 2020). Although it is empirical, with appropriate engineering to ensure fast computations, we can hope for a superior imitation performance using a modulated attention model at the root node.

In addition to designing superior architectures, incorporating superior inductive biases remains an open question. Thus, an improved imitation framework such as Gupta et al. (2022) could be a possible avenue to explore. In hindsight, one can experiment with various modulation schemes. The modulation used in the paper enforces the same element-wise scaling and shifting at all of the B&B nodes. A conditional modulation could be a possible improvement though it will have to be justified against an increase in the computational cost.

Lookback for Learning to Branch. As shown in Figure B.2, the models trained with the proposed framework still lag in imitating the strong branching decisions. Thus, further research into better ways to incorporate the lookback phenomenon in the models could be a promising avenue. For example, one can think of a classifier to predict whether the lookback phenomenon will occur at a particular node. If so, we only need to use the parent logits instead of recomputing the child logits. A complementary research direction would be to discover more such dependence between the observations and incorporate them into the imitation learning framework. We display one such example in Figure B.4 that shows the proportion of time sibling nodes in a B&B tree have the same decisions.

6.2.2 Proactive Contact Tracing

As computer scientists, we are trained to think critically about potential risks and to design safeguards against worst-case scenarios. While we have made significant progress in managing the COVID-19 pandemic with tools such as digital contact tracing apps, its important to consider the limitations of these tools and prepare for potential future pandemics.

One way to do this is to imagine scenarios in which a virus is more severe than SARS-CoV-2 and the current management tools may not be as effective. See Appendix E for such hypothetical viruses. By doing so, we can identify potential weaknesses in our current approach and work to address them.

In this section, we will discuss some limitations of the PCT framework and outline hypothetical solutions. By acknowledging these limitations and developing solutions, we can be better prepared for future pandemics and ensure that we have the tools to effectively manage them.

Scientific communication. This project was initiated in the middle of the pandemic. It was a fast-paced environment with daily brainstorming of ideas. It took a while before we could grasp various technological and socio-political constraints. During this time, we also understood the current paradigm of contact tracing applications

and their shortcomings. User privacy seemed of significant importance as that was also correlated with user adoption. Thus, we addressed this issue in our PCT framework and proposed the framework to the Québec Government at the time. However, scientific communication and navigating the political environment was not our strength. As such, we realized that we had yet to consider non-scientific factors such as communication and political buy-in. The work in Gupta et al. (2023) is an attempt to separate machine learning concepts and explain the framework in plain language; it is intended for epidemiologists. The pitch to the policymakers has to be different. Thus, the work could be extended to provide a clear and concise message for policymakers. This might require looking at the metrics and designing experiments that accurately reflect the political constraints.

Sensitivity to the number of communication bits. In our proposal, we have used four bits to communicate risk levels across the users. One avenue to explore further will be to test the sensitivity of the number of bits on the efficacy of the framework. These results could help assess trade-offs between privacy (number of bits communicated) and the benefits of the framework.

Lightweight training and faster simulations. Given this work's highly time and computationally-intensive nature, it was impossible for us to run a sensitivity analysis in Chapter 5 across various simulator parameters as we did in Chapter 4. Thus, a possible extension of our work could be (a) to design a lightweight imitation learning framework that could be trained faster and with fewer training iterations, and/or (b) to design a faster and scalable simulator. For example, our experiments with deep learning PCT in Chapter 5 were done on 3K agents as compared to 10K agents in Chapter 4. Depending on the time to simulate an infection spread, it could also be used to experiment with reinforcement learning techniques for PCT.

Inductive biases. A possible avenue for further extension is to incorporate inductive biases based on people's behavior or viral load characteristics. For example, the viral load evolution for the SARS-CoV-2 virus is a piecewise function (see Figure C.7).

Thus, a possible avenue to explore further could be to parameterize the neural network's output to be representative of the characteristics of this piecewise function instead of the real values as it is currently done.

Simulator reliability. A simulator can only be a crude approximation of reality. Thus, we need a way to improve the simulator to represent reality as closely as possible. Currently, we used domain randomization (Tobin et al., 2017) to hedge against this issue to train robust deep learning models. However, to align the simulator more closely to reality, we propose incorporating real-world app usage data (after the app is launched) along with the simulated data to improve the simulator (Figure 6.1). A high-level idea is to parametrize the simulator (using neural networks with learnable weights) so that the differences in the simulated data and the real-world observations are used as a feedback to train the weights.

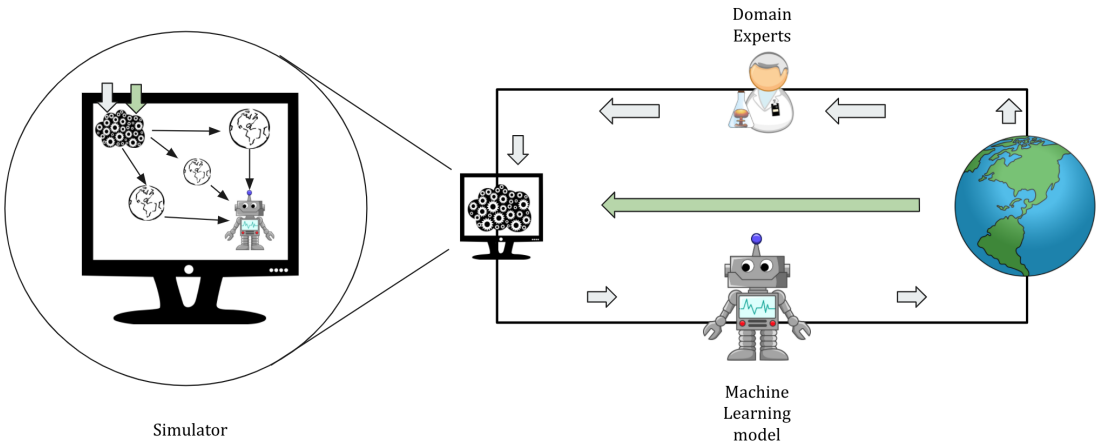

Figure 6.1: We can align the simulator to reality by using app-usage data (green arrow) with simulated data. The methods like wake-sleep algorithm (Hinton et al., 1995) and amortized variational inference (Kingma and Welling, 2013) can be used here.

Interpretable models. We think that for better user-adherence, we should be able to give an app-user the explanation behind the recommended behaviors. For example, if an app-user is asked to avoid public transport, an explanation (e.g., the appearance of symptoms in the past contact) will make the app-user more likely to follow the recommendation. The rule-based predictors are better suited

to serve this purpose. However, it is not easy to extract an explanation from the deep learning model discussed in Chapter 5. To this end, apart from leveraging interpretability research (Gilpin et al., 2018; Murdoch et al., 2019; Rudin, 2019) being done for deep learning models, we think a hybrid approach might also be helpful. For example, the rules can rely on the likelihood of the app-user showing symptoms or test results in the next seven days, which is an output of a learned deep learning model (Figure 6.2).

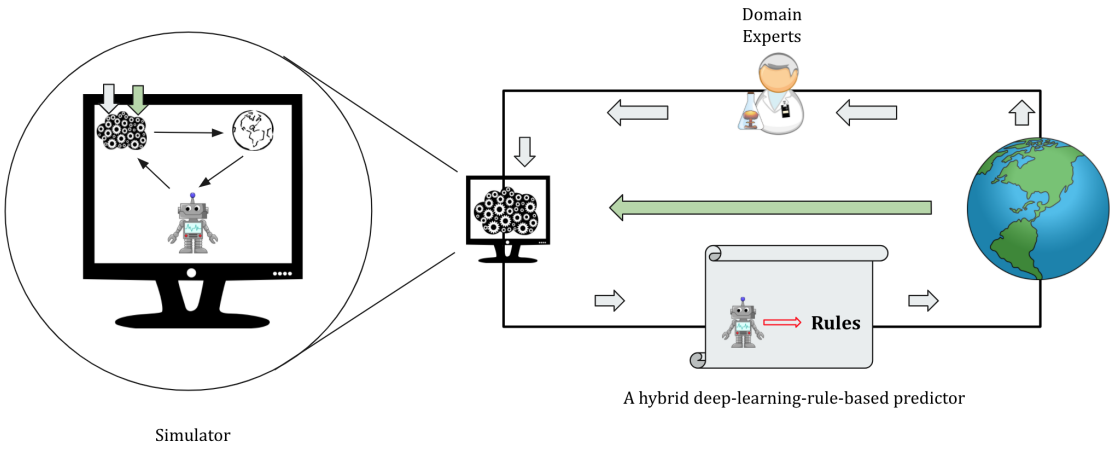

Figure 6.2: In addition to leveraging interpretability research (Gilpin et al., 2018; Murdoch et al., 2019; Rudin, 2019) to design deep learning models capable of generating explanations, we can also opt for a hybrid predictor that has deep learning powered rules.

Offline Reinforcement Learning. Building a simulator takes time. Various assumptions that go into building a simulator might differ from one scenario to another (e.g., different viral load curves or mobility patterns). Additionally, we would not have a reliable simulator until sufficient data from scientific experiments and clinical observations are collected.

In such a case, we think that the research in Offline-RL (Levine et al., 2020) could be used to address this issue. The research community has been applying it to solve problems related to autonomous driving and healthcare. One does not have access to a high-fidelity simulator in these domains. In our context, a roll-out of some basic rules will yield observation-reward pairs. The objective of Offline-RL will

be to learn a policy that improves upon the rolled-out policy (rule-based predictor) without trying them out on a simulator (Figure 6.3).

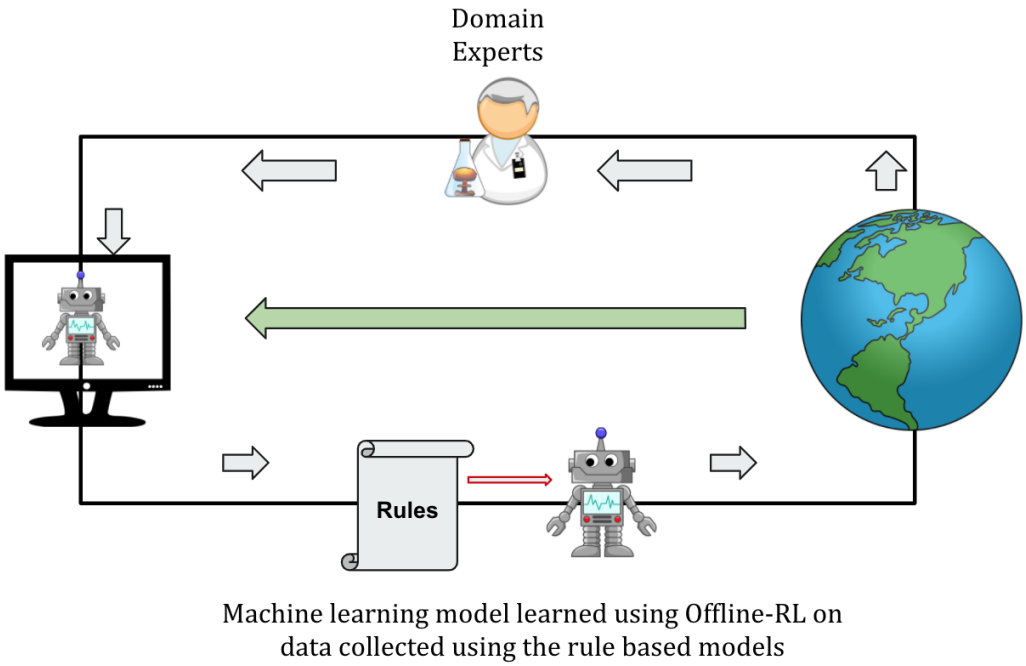

Figure 6.3: Offline-RL (Levine et al., 2020) could potentially help when a fully reliable simulator is not available (e.g., at the very beginning of a pandemic)

The following reasons support the above proposal,

- First, at the onset of a pandemic, when there is not much information about the virus (through clinical observations and scientific experiments) to build a reliable simulator, a conservative rule-based policy could be rapidly deployed. This will not only help us collect initial data points but also, through Offline-RL, we will be able to improve upon the rule-based policy. Subsequent iterations will be very much like the traditional RL.

- Second, in the absence of a scalable and faster simulator, we can hope that Offline-RL on the rule-based policy can give us a better policy with fewer iterations. For example, a supervised framework requires three iterations of training where each iteration collects data from more than 1K simulation runs. It takes 2-3 weeks to have a policy. If time is a constrained resource such that

only one iteration is possible, we can hope to have a better policy through Offline-RL in one iteration as compared to the supervised policy. Note that it is cheaper to run rule-based policies as compared to neural network policies (3 hours for rule-based vs 10 hours for deep learning with 10K simulated agents).

- Finally, a deep learning solution might not be trusted well by authorities from the very beginning. So, it might be a useful practice to have a curriculum of deployment that gradually, and reliably, injects a smarter solution into the system.

Backward tracing. COVID-19 is an overdispersed phenomenon[1], in which a few people infect many while many infect a few. Thus, the reproductive number for COVID-19 may not paint as bad a picture of the outbreak as it might be in reality. For example, the reproductive number of 2 means that an infected person typically transmits the virus to two other people. However, due to dispersion, it may be that a few people infect 10 while a lot of them infect just 1, such that there are 2 mean contagions per infected person. Therefore, it is increasingly important to detect the source of infection as early as possible. This is the objective of backward tracing, which differs from forward tracing (the central idea of PCT) as we are interested in identifying individuals whom the source (e.g., person who reports a positive test result) might have infected.

Recall that the PCT framework relies on warning signals that encode transmitted viral load (for an encounter) in N-bits. It serves as a proxy for the likelihood of infecting the other person involved in an encounter. Though non-trivial, we think that reserving some bits (out of N-bits sent in a warning signal) for the likelihood of getting infected could help to identify the source of infection. A deep learning-based solution could leverage the ground truth to get better at this prediction. However, appropriate rules for a rule-based predictor are not clear at the moment.

[1]This Overlooked Variable Is the Key to the Pandemic, By Zeynep Tufekci. The Atlantic. Available from: `https://www.theatlantic.com/health/archive/2020/09/k-overlooked-variable-driving-pandemic/616548/`

Network optimization. Recall that every time a predictor updates an individual's estimated viral load, it is sent across as a warning signal through a communication channel. Such warning signal updates could heavily consume the network bandwidth while being redundant. For example, suppose the rule-based predictor only cares about the maximum warning signal. In that case, it might be economical to prioritize bandwidth to high-intensity warning signals while ignoring the lower ones.

Thus, not all warning signals are going to be informative. Therefore, it might be helpful to include this constraint as a part of the imitation learning framework. We think that addressing this challenge will help us improve the framework's adoption even further.

Appendices

A

Hybrid Models for Learning to Branch

Contents

A.1 Inefficiency in using GNNs for solving MILPs in parallel

In this section, we argue that the GNN architecture looses its advantages in the face of solving multiple MILPs at the same time. In the applications like multi-objective optimization (Kirlik and Sayın, 2014), where multiple MILPs are solved in parallel, a GNN for each MILP needs to be initialized on the GPU because of the *sequentially asynchronous* nature of solving MILPs. Not only is there a limit to the number of such GNNs that can fit on a single GPU because of memory constraints, but also several GNNs on a single GPU results in an inefficient GPU utilization.

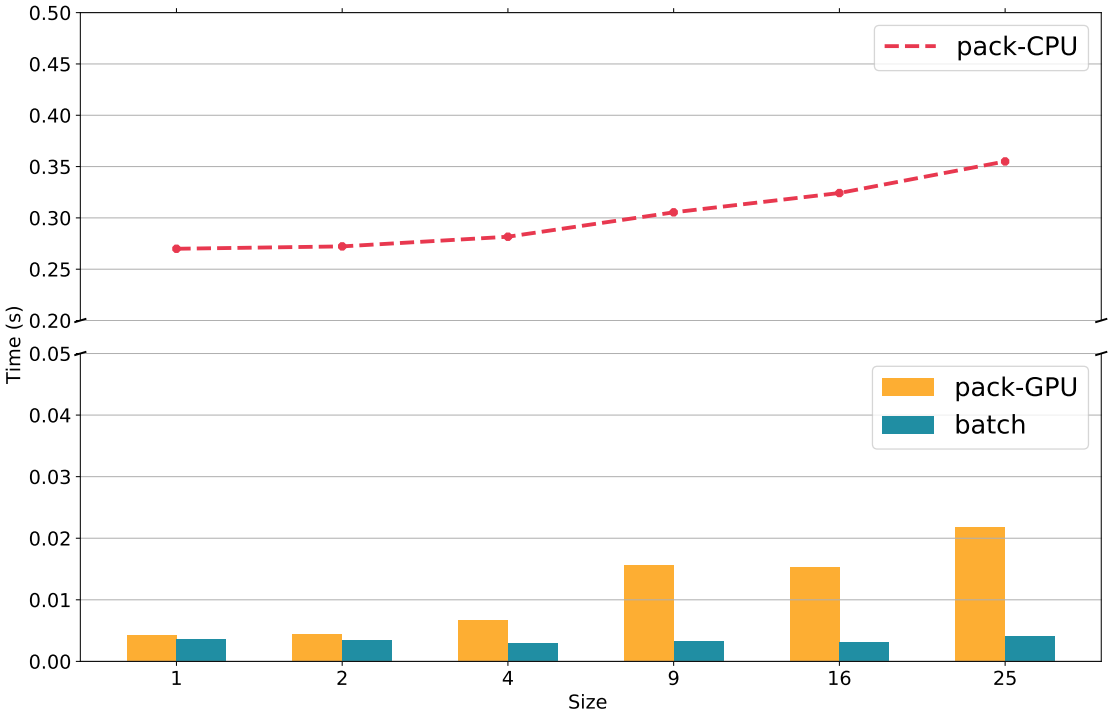

Figure A.1: *Packing* several GNNs together on a GPU keeps it underutilized. "Size" is the number of inputs batched together (unrealistic scenario) or number of inputs simultaneously put on a GPU separately.

One can, for instance, try to time multiple MILPs such that there is a need for a single forward evaluation on a GPU, but, in our knowledge, it has not been done and it results in frequent interruptions in the solving procedure. An alternative, much simpler, method is to *pack* multiple GNNs on a single GPU such that each GNN is dedicated to solving one MILP. For example, we were able to put 25 GNNs

on Tesla V100 32 GB GPU. Figure A.1 shows the inefficient utilization of GPUs when multiple GNNs are packed on a single GPU.

A.2 Input Features

We use the features that were used by Gasse et al. (2019) and Khalil et al. (2016). There are a total of 92 features that we use for $\mathbf{X}$ as described in the Table A.1.

Table A.1: Features in $\mathbf{X}$, an input to MLP.

	count
Variable features from $\mathbf{G}$	13
Variable features from Khalil et al. (2016)	72

The list of features in $\mathbf{G}$ are described in the Table A.2. We follow the preprocessing procedure as described in Gasse et al. (2019).

Table A.2: Features of $\mathbf{G}$. It is constructed as a variable-constraint bipartite graph, where the vertices of this graph have features as described here. These features are same as used in Gasse et al. (2019).

	Type	Description
Variable Features (Total: 13)		
Variable type (1)	Categorical	Allowed values - Binary, Integer, Continuous, Implied Integer
Normalized coefficient (1)	Real	Objective coefficient of the variable normalized by the euclidean norm of its coefficients in the constraints
Specified bounds (2)	Binary	Does the variable has a lower bound (upper bound)?
Solution bounds (2)	Binary	If the variable is currently at its lower bound (upper bound)?
Solution fractionality (1)	Real $\in [0, 1)$	Fractional part of the variable i.e. $x - \lfloor x \rfloor$, where x is the value of the decision in variable in the current LP solution
Basis (1)	Categorical	One of 4 classes - Lower (variable is at the lower bound), Basic (variable has a value between the bounds), Upper (variable is at the upper bound), Zero (rare case)

Reduced cost (1)	Real	Amount by which objective coefficient of the variable should decrease so that the variable assumes a positive value in the LP solution
Age (1)	Real	Number of LP iterations since the last time the variable was basic normalized by total number of LP iterations
Solution value (1)	Real	Value of the variable in the current LP solution
Primal value (1)	Real	Value of the variable in the current best primal solution
Average primal value (1)	Real	Average value of the variable in all of the previously observed feasible primal solutions
Constraint Features (Total: 5)		
Cossine similarity (1)	Real	Cossine of angle between the vector represented by objective coefficients and the coefficients of this constraint
Bias (1)	Real	Right hand side of the constraint normalized by the euclidean norm of the row coefficients
Age (1)	Real	Number of iterations since the last time the constraint was active normalized by total number of LP iterations
Normalized dual value (1)	Real	Value of dual variable corresponding to the constraint normalized by the product of norms of the row coefficients and the objective coefficients
Bounds (1)	Binary	If the constraint is currently at its bounds?
Edge Features (Total: 1)		
Normalized coefficient (1)	Real	Coefficient of the variable normalized by the norm of the coefficients of all the variables in the constraint

A.3 Preliminary results on running times of various architectures

In order to have a rough idea of relative improvement in runtimes across various architectures, we considered 20 instances in each difficulty level for each problem class

and use the solver to obtain observations at each node. Different functions are then used to evaluate decisions at each node, and the total time taken for all the function evaluations across the nodes in an instance is observed. Note that the quality of decision is not important here; we are only interested in time taken per decision.

Specifically, we considered 5 forms of architectures as listed in Table A.3. To cover the entire spectrum of strong inductive biases, we constructed an attention mechanism as explained in A.4. At the same time, to cover the spectrum of cheaper architectures we consider simple dot product as the form of predictor. Finally, we consider a scenario where we only consider MLPs as predictors everywhere in the B&B tree.

Table A.3: Different architectures considered for preliminray runtime comparison.

Type	Description
GNN ALL	Use a Graph Convolution Neural Network (GNN) Gasse et al. (2019) at all *tree nodes*
ATTN ALL	Attention mechanism (section A.4) at all nodes
GNN DOT	Use GNN at the *root nodes* and a dot product with $\mathbf{X}$ to compute scores at every node
ATTN DOT	Use attention mechanism at the root node and dot product with $\mathbf{X}$ to compute scores at every node
MLP ALL	Use a 3-layer multilinear perceptron at all the nodes

Figure A.2 shows the relative performance (rel. to GNN) of various deep learning architectures across 4 sets of problems. It is evident that MLP ALL and GNN DOT are favored across the problem sets. This observation inspired the range of *hybrid architectures* that we explored in the paper. We also observe that a superior inductive bias like that of attention mechanism and transformers (Vaswani et al., 2017) has a better runtime performance on GPUs as illustrated in Figure A.3, which we suspect is massive parallelization employed in the computations of attention. However, their performance on CPUs is not better than GNNs.

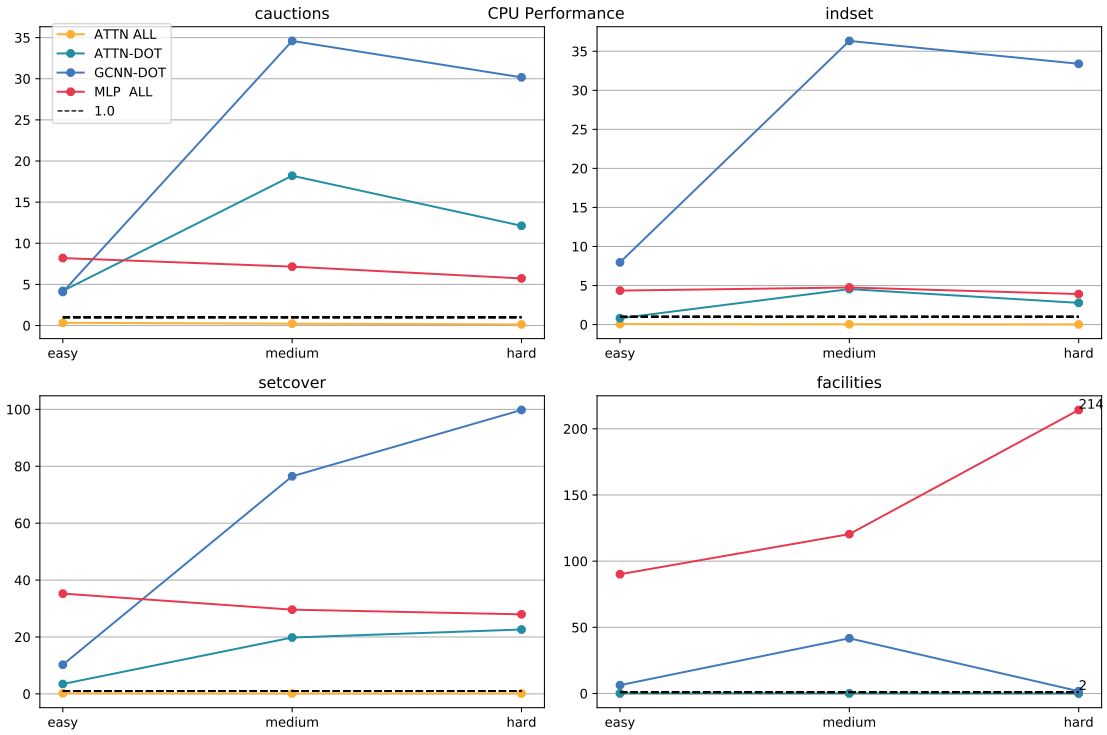

Figure A.2: Relative time performance of various methods with respect to GNN ALL on CPU (average values). A value of 10 implies that the method is 10 times faster (on arithmetic average) than GNN ALL implying that one can afford to perform 10 times worse than GNN ALL in iterative performance when using CPUs. Note that this is just a rough estimation.

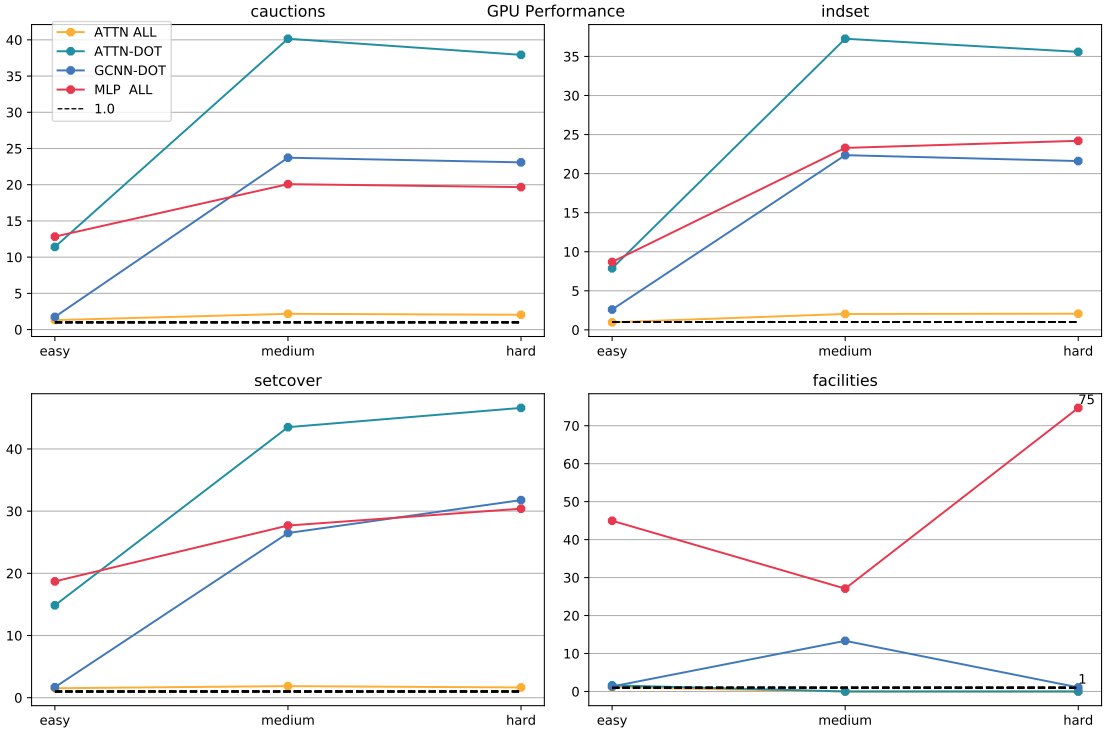

Figure A.3: Relative time performance of various methods with respect to *GNNALL* on GPU (average values). A value of 10 implies that the method is 10 times faster (on arithmetic average) than *GNNALL* implying that one can afford to perform 10 times worse than *GNNALL* in iterative performance when using CPUs. Note that this is just a rough estimation.

A.4 Attention Mechanism for MILPs

To cover the entire spectrum of computational complexity and expressivity of inductive bias, we implemented a transformer (Vaswani et al., 2017) as an architecture to replace GNNs. Specifically, we let the variables (constraints) attend to all other variables (constraints) via multi-headed self-attention mechanism. Finally, a modulated attention mechanism between *variable representations* as queries and *constraint representation* as keys outputs final *variable representations*, which is passed through the softmax layer for classification objective. Here, we use modulation scheme as explained in Shaw et al. (2018). Precisely, an edge in variable-constraint graph is used to increment the attention score with a learnable scalar value. We illustrate the architecture in Figure A.4.

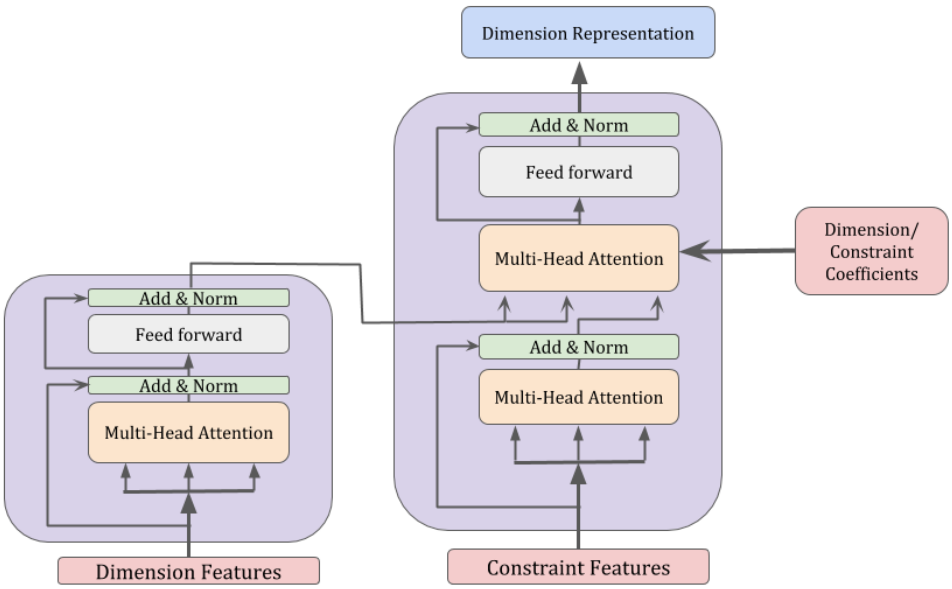

Figure A.4: Multi-Head Attention mechanism where a variable or constraint can attend to all other variables or constraints. Finally,variables are used as a query to attend to constraints, where the attention is modulated through variable-constraint features. Modulation of attention scores follow Shaw et al. (2018)

A.5 Data Generation & Training Specifications

For our experiments, we used 10,000 training instances for the training dataset and collected 150,000 observations from the tree nodes of those instances. In a similar manner, we generated 20,000 instances each for the validation and testing set, resulting in 30,000 observations each for validation and testing respectively.

We held all the training parameters fixed across the models. Specifically, we used a learning rate of $1e^{-3}$, training batch size of 32, a learning rate scheduler to reduce learning rate by 0.2 if there is no improvement in the validation loss for 15 epochs, and an early stopping criterion of 30 epochs. Our epoch consisted of 10K training examples and 2K validation samples. We used a 3 layered MLP with 256 hidden units in each layer, while we used GNN model with an embedding size of 64 units. We used this configuration across all the models discussed in the main paper. Due to the large size of instances in capacitated facility location, we used a learning rate of 0.005, early stopping criterion of 20 epochs with a patience of 10 epochs.

Further, for knowledge distillation we used $T = 2$ (temperature) and $\alpha = 0.9$ (convex mixing of soft and hard objectives). These are the recommended settings in Hinton et al. (2015). We did a hyperparameter search for $\beta = \{0.01, 0.001, 0.0001\}$ for ED and MHE. Following are the values for β that resulted in the best performing models.

Table A.4: Best AT models

	AT
Capacitated Facility Location	MHE
Combinatorial Auctions	MHE
Set Covering	ED
Maximum Independent Set	MHE

We implemented all the models using PyTorch (Paszke et al., 2017), and ran all the CPU evaluations on an Intel(R) Xeon(R) CPU E5-2650 v4 @ 2.20GHz. GPU evaluations for GNN were performed on NVIDIA-TITAN Xp GPU card with CUDA 10.1.

A.6 Depth-dependent loss weighting scheme

In Figure A.5, we plot the sorted ratio of number of nodes to the minimum number of nodes observed for an instance across all the weighting schemes. Here we attempt to breakdown the performance of different branching strategies learned using different loss-weighting schemes. We observe that the *sigmoidal* scheme achieves the best performance.

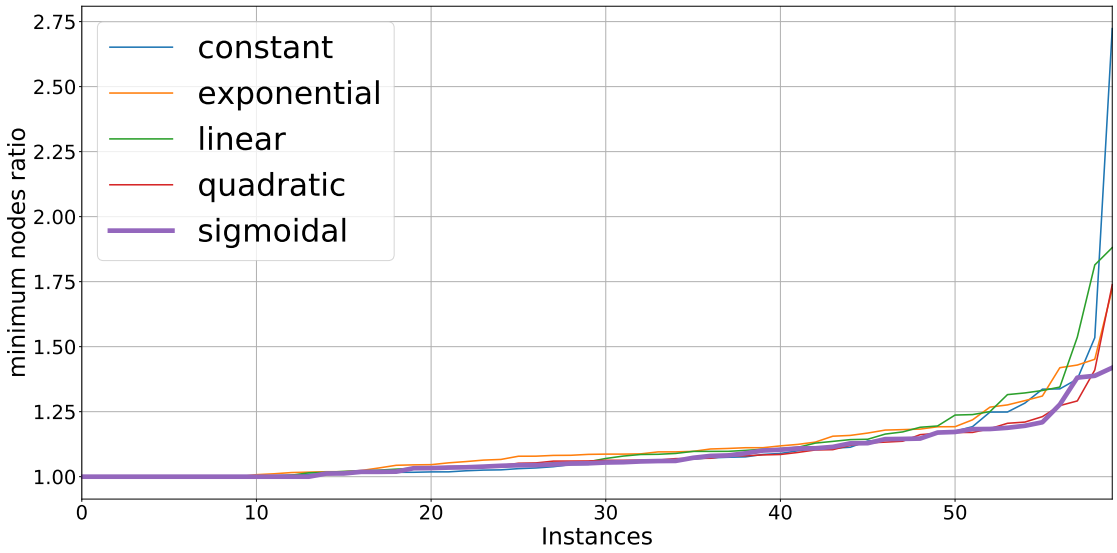

Figure A.5: Performance of different weighing schemes across "big" instances. "Sigmoidal" evolves slowest among all.

A.7 Model Results

Table A.5 below is a list of all the architectures along with their performance on the test sets. Please note that training with auxiliary task is not possible with HyperSVM type of architecture so their results are not reported in the table.

Table A.5: Top-1 accuracy of various models.

		Combinatorial Auctions	Capacitated Facility Location	Set Cover	Maximum Independent Set
Expert	GCNN	46.53 ± 0.12	68.47 ± 0.22	54.82 ± 0.14	58.73 ± 0.54
Existing methods	Extratrees	37.97 ± 0.10	60.06 ± 0.18	42.66 ± 0.22	19.35 ± 0.55
	LMART	38.70 ± 0.16	63.28 ± 0.06	46.31 ± 0.28	50.98 ± 0.37
	SVMRank	39.14 ± 0.12	63.47 ± 0.06	46.23 ± 0.11	49.29 ± 0.23
Simple Network	MLP	43.53 ± 0.13	65.26 ± 0.16	49.53 ± 0.04	53.06 ± 0.03
Concatenate	CONCAT (Pre)	44.16 ± 0.03	65.50 ± 0.10	49.98 ± 0.18	52.85 ± 0.34
	CONCAT (e2e)	44.09 ± 0.08	66.59 ± 0.04	50.17 ± 0.04	53.23 ± 0.41
	CONCAT (e2e & KD)	44.19 ± 0.05	66.53 ± 0.13	50.11 ± 0.09	53.66 ± 0.32
Modulate	FiLM (Pre)	44.12 ± 0.09	65.78 ± 0.06	50.00 ± 0.09	53.16 ± 0.51
	FiLM (e2e)	44.31 ± 0.08	66.33 ± 0.33	50.16 ± 0.05	53.23 ± 0.58
	FiLM (e2e & KD)	44.10 ± 0.09	66.60 ± 0.21	50.31 ± 0.19	53.08 ± 0.30
HyperSVM	HyperSVM (e2e)	43.19 ± 0.02	66.05 ± 0.06	49.78 ± 0.23	50.04 ± 0.31
	HyperSVM (e2e & KD)	42.55 ± 0.03	66.07 ± 0.05	49.53 ± 0.13	49.34 ± 0.43
HyperSVM-FiLM	HyperSVM-FiLM (e2e)	43.64 ± 0.18	65.54 ± 0.32	49.81 ± 0.27	50.17 ± 0.58
	HyperSVM-FiLM (e2e & KD)	43.28 ± 0.48	65.52 ± 0.34	49.73 ± 0.05	49.73 ± 0.39
	FiLM (e2e & KD & AT)	**44.56 ± 0.13**	**66.85 ± 0.28**	**50.37 ± 0.03**	**53.68 ± 0.23**

A.8 Overfitting in maximum independent set

Table A.6 shows that the FiLM models overfit on small instances of maximum independent set. To address this problem, we used a mini dataset of 2000 observations obtained by running data collection on medium instances of maximum independent set. Further, we regularized the FiLM parameters of the FiLM model to yield much simpler models based on the performance on this mini-dataset, which is not too expensive to obtain owing to the size of observations. The results of the regularized models are in Table A.7 For a fair comparison, we regularized GNN and used the best performing model to report evaluation results in the main paper.

Table A.6: FiLM models for maximum independent set overfits on small instances

	small			medium			big		
	Time	Wins	Nodes	Time	Wins	Nodes	Time	Wins	Nodes
FiLM	52.96	**39**/ 55	492	1515.19	9/ 17	2804	2700.02	0/ 0	nan
GNN-CPU	**44.07**	21/ 60	**432**	**371.81**	**36**/ 40	**558**	**1981.43**	**10**/ 10	**6334**
GNN-GPU	31.70	0/ 59	432	264.01	0/ 43	558	1772.12	0/ 13	6313

Table A.7: Top-1 accuracy of regularized models on 2000 observations from medium random instances of maximum independent set.

weight decay	FiLM	GNN
1.0	55.15 ± 0.07	31.6 ± 6.63
0.1	**56.13 ± 0.32**	**37.23 ± 0.92**
0.01	53.25 ± 0.95	26.8 ± 16.71
0.0	19.18 ± 4.24	34.08 ± 4.8

A.9 Performance of HyperSVM architectures

HyperSVM architectures are the cheapest in computation. In this section we ask if we are willing to lose machine learning accuracy by 1-2% in HyperSVM architecture, can we still get faster running times with HyperSVM type of architectures? Table A.8 compares solver performance by using HyperSVM architecture and FiLM architecture. We observe that HyperSVM types of architectures loose their ability to generalize on larger scale instances.

Table A.8: FiLM architecture generalizes to larger instances better than the HyperSVM types of architectures. We compare the performance of the best FiLM architecture from the Table A.5 with the best HyperSVM architecture in the same table.

	small			medium			big		
	Time	Wins	Nodes	Time	Wins	Nodes	Time	Wins	Nodes
FiLM	**24.67**	**53**/ 60	**109**	**136.42**	**51**/ 60	**336**	**531.70**	**46**/ 57	**345**
HyperSVM	27.26	7/ 60	110	158.97	9/ 60	345	614.36	11/ 57	346
MLP	27.61	4/ 60	114	156.30	11/ 60	347	595.31	9/ 56	334

(a) Capacitated Facility Location

	small			medium			big		
	Time	Wins	Nodes	Time	Wins	Nodes	Time	Wins	Nodes
FiLM	**8.73**	**59**/ 60	**147**	**63.75**	**60**/ 60	**2169**	**1843.24**	**24**/ 26	**38530**
HyperSVM-FiLM	9.73	1/ 60	148	72.53	0/ 60	2217	2061.56	1/ 22	47277
MLP	9.98	0/ 60	157	77.48	0/ 60	2299	1984.26	1/ 24	40188

(b) Set Cover

A.10 Scaling to twice the size of Big instances

In this section we investigate the generalization power of FiLM models trained on dataset obtained from small instances. Specifically, we generate 20 random instances of size double that of big instances, and use the trained models of FiLM

and GNN to compare their performance against RPB. We use the time limit of 7200 seconds, i.e. 2 hours to account for longer solving running times as one scales out to bigger instances. We observe that the power to generalize is largely dependent on the problem family. For example, FiLM models can still outperform other strategies on scaling out on capacitated facility location problems. However, we found that RPB remains competitive on setcover problems. We note that this shortcoming of the FiLM models can be overcome via larger size of hidden layers and training on slightly larger instances than what has been used in the main paper. Table A.9 shows these results.

Table A.9: Performance of branching strategies on twice the size of biggest instances (2 × Big) considered in the main paper. 20 "Bigger" instances were solved using 3 seeds each resulting in a total of 60 runs. We see that FiLM models still remain competitive, and it is highly dependent on the family of problem.

	facilities			setcover		
Model	Time	Wins	Nodes	Time	Wins	Nodes
RPB	7200.14	0 / 0	n/a	**6346.17**	**9** / 12	135 132
GNN	7111.83	1 / 5	n/a	7200.25	0 / 0	n/a
FiLM (ours)	**7052.27**	**4** / 4	n/a	6508.78	3 / 10	**107 187**
GNN-GPU	6625.55	– / 13	n/a	6008.14	– / 12	93 909

A.11 Effect of different training protocols on B&B performance

Table A.10 shows the performance of different training protocols on B&B performance. Although the models trained with auxiliary tasks (AT) and knowledge distillation (KD) are a clear winner up until problem sets of size medium, there is a tie between the models trained with KD and those trained with KD & AT. However, it is worth noting that the difference in B&B performance across different training protocols is not huge, and such performance evaluation can be read from the test accuracy of trained models. For example, as evident in Table A.5, difference in test accuracy of FiLM (e2e & KD) and FiLM (e2e & KD & AT) is not significant for problem sets - Capacitated Facility Location and Set Covering, but its significant enough for problem sets - Combinatorial Auctions and Maximum Independent Set. **Given that the inference cost is independent of training protocols, to attain better**

Table A.10: Effect of training protocols on the performance of *branching strategies*. We report geometric mean of solving times, number of times a method won (in solving time) over total finished runs, and geometric mean of number of nodes. Refer to section 5 (main) for more details. The best performing results are in **bold**. *Models were regularized to prevent overfitting on small instances.

	Small			Medium			Big		
Model	Time	Wins	Nodes	Time	Wins	Nodes	Time	Wins	Nodes
e2e	25.05	22 / 60	114	154.12	4 / 60	340	550.03	15 / 57	339
e2e + KD	28.00	2 / 60	111	143.12	18 / 60	343	**507.50**	**26** / 57	**325**
e2e + KD + AT	**24.67**	**36** / 60	**109**	**136.42**	**38** / 60	**336**	531.70	16 / 57	345
				Capacitated Facility Location					
e2e	9.70	1 / 60	152	71.18	1 / 60	2186	1869.57	4 / 25	**40 341**
e2e + KD	9.81	0 / 60	**146**	70.88	4 / 60	2173	**1842.29**	**15** / 25	40 437
e2e + KD + AT	**8.73**	**59** / 60	147	**63.75**	**55** / 60	**2169**	1843.24	8 / 26	40 881
				Set Covering					
e2e	2.45	1 / 60	**72**	17.59	5 / 60	702	225.88	17 / 60	8939
e2e + KD	2.35	0 / 60	73	17.59	2 / 60	720	241.95	7 / 59	8846
e2e + KD + AT	**2.13**	**59** / 60	73	**15.71**	**53** / 60	**686**	**217.02**	**36** / 60	**8711**
				Combinatorial Auctions					
e2e	205.63	2 / 54	559	1103.47	1 / 29	1137	2457.94	1 / 4	**2268**
e2e + KD	333.52	1 / 52	486	926.12	1 / 41	604	2503.65	0 / 7	1953
e2e + KD + AT	**52.96**	**54** / 55	**410**	**131.45**	**54** / 54	**331**	**1823.29**	**14** / 15	3049
				Maximum Independent Set*					

generalization performance, we recommend FiLM (e2e & KD & AT) only when these models have significantly better accuracy than FiLM (e2e & KD).

B

Lookback for Learning to Branch

Contents

B.1 Lookback property on some real-world instances

We look at the frequency of the lookback property on some of the real-world instances. Specifically, we study this property in the context of MIP instances related to wildlife management, first proposed by Dilkina et al. (2017). These instances have been used by the MIP community to demonstrate the efficacy of various proposals on the real-world MIP instances (Nair et al., 2020; Hutter et al., 2010; He et al., 2014a). Table B.1 shows that at least 30% of the parent-child pairs exhibit this property.

Table B.1: Frequency of the lookback property in the real-world instances is as prevalent as in the synthetic instances considered in the main paper. These instances are made available by Dilkina et al. (2017).

Instances	Description	number of parent-child pairs collected	number of parent-child pairs exhibiting the lookback property	Frequency of the lookback property
CORLAT	Corridor planning in wildlife management	5082	1765	34.73%
RCW	Red-cockaded woodpecker diffusion conservation	5115	1952	38.16%

B.2 Lookback property as a function of depth-decile

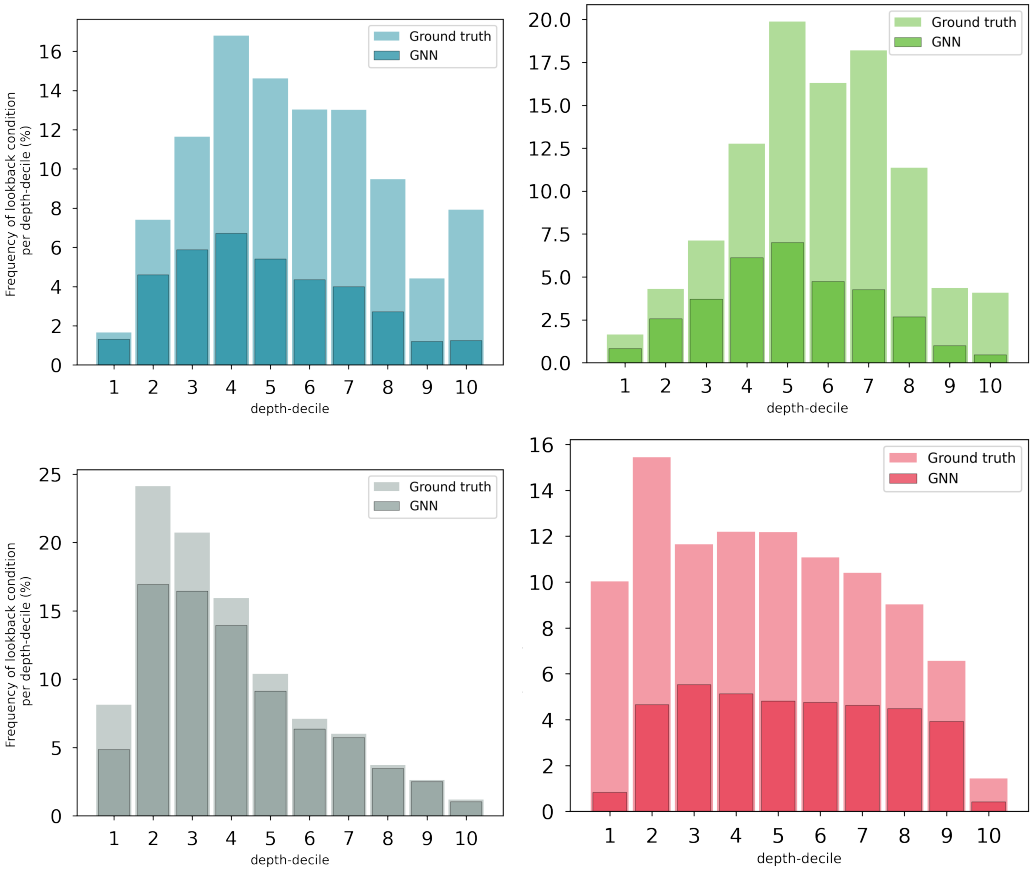

Figure B.1: The gap between the frequency of the lookback condition per *depth-decile* for the strong branching oracle (Ground truth) and the traditionally trained GNNs (GNN) presents an opportunity to improve the GNN models. See Section 3.4 for the dataset collection procedure.

B.3 Model Selection Objectives

To incorporate the objective of maximum solved instances, we select the branching strategy as per,

$$\begin{aligned} \mathcal{B}^{'} &= \{j \mid j \in \mathcal{B}, \quad w(j) = \max_{\mathcal{B}} w(b)\}, \\ b^{\star} &= \underset{b \in \mathcal{B}^{'}}{\operatorname{argmin}} n(b, \mathcal{B}^{'}) \end{aligned} \tag{B.1}$$

Finally, to select the strategy only with the minimum node count, we select the branching strategy as per,

$$b^{\star} = \underset{b \in \mathcal{B}}{\operatorname{argmin}} n(b, \mathcal{B}) \tag{B.2}$$

B.4 Instance Specifications

We follow the procedure outlined by Gasse et al. (2019) to generate and scale the instances. A well commented and functional code to generate these instances can be found on the authors' Github repository[1]. For convenience, we describe the procedure here.

B.4.1 Combinatorial Auction

These instances are generated following the arbitrary scheme described in the section 4.3 of Leyton-Brown et al. (2000). The scalable parameters are the number of items and the number of bids.

Table B.2: Parameters for Combinatorial Auctions

Instance Size	number of items	number of bids	use
Small-Training	100	500	Dataset Collection & Training
Small-Validation	100	500	Dataset Collection & Validation
Medium-Validation	200	1000	Validation Evaluation
Small	100	500	Test Evaluation
Medium	200	1000	Test Evaluation
Big	300	1500	Test Evaluation
Bigger	350	1750	Test Evaluation

B.4.2 Set Covering

These instances are generated using the method described in Balas and Ho (1980). The scalable parameters are number of items, where the number of sets are fixed to 1000.

[1] `https://github.com/ds4dm/learn2branch/blob/master/01_generate_instances.py`

Table B.3: Parameters for Minimum Weighted Set Cover. Number of sets is fixed to 1000.

Instance Size	number of items	use
Small-Training	500	Dataset Collection & Training
Small-Validation	500	Dataset Collection & Validation
Medium-Validation	1000	Validation Evaluation
Small	500	Test Evaluation
Medium	1000	Test Evaluation
Big	2000	Test Evaluation

B.4.3 Capacitated Facility Location

These instances are generated the method described by Cornuéjols et al. (1991). Fixing the number of facilities to 100, the scalable parameter is the number of customers.

Table B.4: Parameters for Capacitated Facility Location. Number of facilities is fixed to 100.

Instance Size	number of customers	use
Small-Training	100	Dataset Collection & Training
Small-Validation	100	Dataset Collection & Validation
Medium-Validation	200	Validation Evaluation
Small	100	Test Evaluation
Medium	200	Test Evaluation
Big	400	Test Evaluation

B.4.4 Maximum Independent Set

These instances are generated by formulating the maximum independent set problem on a randomly generated Barabási-Albert with edge probability of 0.4. The scalable parameter is the number of nodes.

Table B.5: Parameters for Maximum Independent Set. Affinity is fixed to 4.

Instance Size	number of nodes	use
Small-Training	750	Dataset Collection & Training
Small-Validation	750	Dataset Collection & Validation
Medium-Validation	1000	Validation Evaluation
Small	750	Test Evaluation
Medium	1000	Test Evaluation
Big	1500	Test Evaluation

B.5 Data Generation

For each of the problem family, we generate 10,000 *Small* random instances to collect training data, 2,000 *Small* random instances to collect the validation data, and 20 *Medium* instances for model selection. We use SCIP Gleixner et al. (2018) with strong branching heuristic to solve the randomly generated instances and collect data of the form $\mathcal{D} = \{(\mathcal{G}_i^0, \mathbf{s}_i^0, \mathcal{G}_i^1, \mathbf{s}_i^1) \mid i \in \{1, 2, 3, ..., N\}\}$, as described in the main text. We collected a total of 150,000 training observations and 30,000 validation observations.

The hand-engineered features to the GNNs are same as Gasse et al. (2019). For a full description of these features, pleasee see Section 2 in Supplementary material of Gupta et al. (2020a).

B.6 Training Specifications

Our models are all implemented in PyTorch (Paszke et al., 2017). Following Gupta et al. (2020a), we didn't change any of the training parameters, for example, we used the Adam (Kingma and Ba, 2014) optimizer with the learning rate of $1e^{-3}$, training batch size of 32, and a learning rate scheduler to reduce the learning rate by a factor of 0.2 in the absence of any improvement in the validation loss for the last 15 epochs Moreover, we use the early stopping criterion to stop the training if the validation loss doesn't improve over 30 epochs. We validate the performance of model on the validation dataset after every epoch consisting of 10K random training samples. For each problem family, we trained models with five random seeds.

B.7 TunedRPB

We consider two parameters in RPB to negotiate the trade-off between the compactness of the resulting B&B tree and the computational time. Broadly, RPB performs strong branching on MaxLookahead candidates until it has collected enough information to reliably act on it. The selection of candidates is prioritized by the reliability of *pseudoscores* (statistically learned score to estimate bound improvement per fractional rounding of the variable) collected during the strong branching operations. If the minimum psuedoscore obtained by withere rounding up or rounding down of the integer-constrained vairable is less than MaxReliable, the candidate is deemed unreliable, thereby prioritizing it in the next rows. Thus, we observe the following trade-off by varying these two parameters: (i) MaxReliable: lower values prefer faster solving times at the expense of larger trees, and (ii) MaxLookahead: higher values prefer shorter trees at the expense of more computational time.

To have a version of RPB that is trained on the training instances of interest, we run a hyperparameter grid-search on the following values -

1. MaxLookahead: $\{6, 7, 8, 9, 10, 11\}$
2. MaxReliable: $\{3, 4, 5, 6, 7, 8\}$

Specifically, we followed the procedure as described in Section 3.5 to solve 100 randomly generated medium instances, and select the best performing hyperparameters according to Eq. (3.9).

B.8 Evaluation Specification

The evaluation on *Big* instances is performed by using SCIP 6.0.1 installed on an Intel(R) Xeon(R) CPU E5-2650 v4 @ 2.20GHz. The learned neural networks, as branching models, are run on NVIDIA-TITAN Xp GPU card with CUDA 10.1. All of the main evaluations are done, as is a standard practice, while ensuring that the ratio of the number of cores on the machine to the load average is more than 4. This condition ensures that each process on the machine has at least 4 CPUs at a time.

The model selection is carried out by solving *Medium* instances on the shared cluster as specified by Nair et al. (2020) (see Section 12.7). Specifically, we solve a benchmark MIPLIB problem ('vpm2.mps') every 60 seconds to collect the solving time statistics. Given the solving times of the benchmark problem on the reference machine, we recalibrate the solving time of the instance, which is used as the final running time. We solve 20 independently generated *Medium* instances with five seeds resulting in 100 independent evaluations.

We followed the standard procedure used proposed by Gasse et al. (2019) to set SCIP for evaluating branching strategies. Specifically, we used the following SCIP parameters to evaluate the branching strategies

1. Cutting planes are allowed only at the root node
2. No restarts are allowed

B.9 Hyperparameters & Model Selection

We use the ridge regression to penalize the parameters θ_y and θ_z. In addition to λ_{l2}, θ_{PAT} searches over $target$ and λ_{PAT}. The following values were used for the hyperparameter search -

1. $\lambda_{l2} = \{0.01, 0.1, 1.0\}$
2. $\lambda_{PAT} = \{0.01, 0.1, 0.2, 0.3\}$
3. $target = \{\mathbf{y}, \mathbf{z}\}$

Finally, Table B.6 shows the best performing hyperparameters according to Eq. (3.9) for θ_y, θ_z, and θ_{PAT}. As observed in Gupta et al. (2020a), we found that the $l2$-regularization is useful for the performance of indset models.

Table B.6: Best performing hyperparameters according to Eq. (3.9)

problem family	$\theta_y\{\lambda_{l2}\}$	$\theta_z\{\lambda_{l2}\}$	$\theta_{PAT}\{target, \lambda_{l2}, \lambda_{PAT}\}$
cauctions	0.0	0.0	$\{z, 0.0, 0.1\}$
setcover	0.0	0.0	$\{y, 0.0, 0.1\}$
facilities	0.0	0.0	$\{z, 0.0, 0.1\}$
indset	0.1	0.1	$\{z, 0.01, 0.2\}$

B.10 Performance on medium validation instances

Table B.7 shows the data that was used to plot the points in Figure 3.4. Due to the huge variability in the performance metrics of indset we omit the performance of $\theta_{accuracy}$ from Figure 3.4.

Table B.7: Data for Figure 3.4

Model	Metric	cauctions	setcover	facilities	indset
θ_y	Time	14.66	38.21	159.60	73.16
	Nodes	1077	1546	421	3102
θ_z	Time	14.82	38.08	157.64	76.61
	Nodes	1089	1527	437	3335
θ_{yPAT}	Time	13.83	37.92	160.13	74.80
	Nodes	1031	1503	429	3201
θ_{zPAT}	Time	13.65	37.89	153.81	71.32
	Nodes	1022	1513	420	2943
$\theta_{accuracy}$	Time	14.22	37.92	159.59	572.39
	Nodes	1083	1503	421	9430

B.11 Full performance of Combinatorial Auction models on Big and Bigger instances

Since *Big* Combinatorial Auction instances are solved by all the strategies, we show them in Table B.8. We extend the size of these instances to *Bigger* and show the evaluation in the main paper without FSB as there are only 5 instances solved distorting the comparison of Time (c) and Node (c). See Table B.9 for the full results.

Table B.8: Performance of variable selection strategies on *Big* instances for Combiatorial Auctions instances. See Appendix B.4 for the instance scaling parameters.

Model	Time	Time (c)	Wins	Solved	Nodes (c)
FSB	2075.60	1643.78	0	53	336
RPB	202.78	113.47	1	100	4640
TUNEDRPB	205.57	112.80	0	100	4611
GNN	**139.30**	**72.17**	**64**	100	4142
GNN-PAT (ours)	141.63	73.31	35	100	**3754**

Table B.9: Performance of variable selection strategies on *Bigger* instances for Combinatorial Auctions instances. See Appendix B.4 for the instance scaling parameters.

Model	Time	Time (c)	Wins	Solved	Nodes (c)	Optimality Gap (16)
FSB	2591.81	1793.64	0	10	257	0.033 102
RPB	626.81	88.22	1	80	5719	0.009 555
TUNEDRPB	644.20	97.72	0	80	6191	0.009 605
GNN	507.06	**59.22**	14	80	**5630**	0.008 307
GNN-PAT (ours)	**477.26**	65.62	**69**	**84**	5757	**0.007 844**

B.12 Performance on small and medium instances

Table B.10 shows the performance of the selected strategies as per Eq. (3.9) on small and medium instances. This table is a counterpart to the Table 3.2.

Table B.10: Performance of *branching strategies* on evaluation instances. We report geometric mean of solving times, number of times a method won (in solving time) over total finished runs, and geometric mean of number of nodes. Refer to section 2.5 for more details. The best performing results are in **bold**.

	Small				Medium			
Model	Time	Wins	Solved	Nodes	Time	Wins	Solved	Nodes
FSB	5.85	0	100	6	127.56	0	100	72
RPB	3.89	0	100	**11**	25.31	0	100	696
TUNEDRPB	3.72	0	100	**11**	24.70	0	100	**591**
GNN	**2.10**	**82**	100	71	**13.15**	**58**	100	693
GNN-PAT (ours)	2.18	18	100	72	13.25	42	100	654
Combinatorial Auction								
FSB	26.17	0	100	17	531.47	0	75	117
RPB	13.41	0	100	54	91.33	0	100	1119
TUNEDRPB	13.63	0	100	**48**	93.60	0	100	1131
GNN	9.37	5	100	136	64.81	1	100	1030
GNN-PAT (ours)	**9.03**	**95**	100	134	**58.48**	**99**	100	**997**
Set Covering								
FSB	41.61	3	100	14	264.67	3	98	73
RPB	36.58	3	100	22	206.28	1	100	147
TUNEDRPB	37.56	1	100	**21**	211.77	2	100	**140**
GNN	**27.20**	**78**	100	113	**146.41**	**64**	100	320
GNN-PAT (ours)	29.46	15	100	112	159.95	30	100	329
Capacitated Facility Location								
FSB	626.33	0	93	54	1634.40	0	60	46
RPB	58.03	0	100	702	144.92	0	100	770
TUNEDRPB	56.99	1	100	697	143.88	2	100	**748**
GNN	35.04	28	100	1000	76.53	36	100	795
GNN-PAT (ours)	**31.96**	**71**	100	**455**	**72.00**	**62**	100	789
Maximum Independent Set								

B.13 Post-hoc analysis of the lookback property

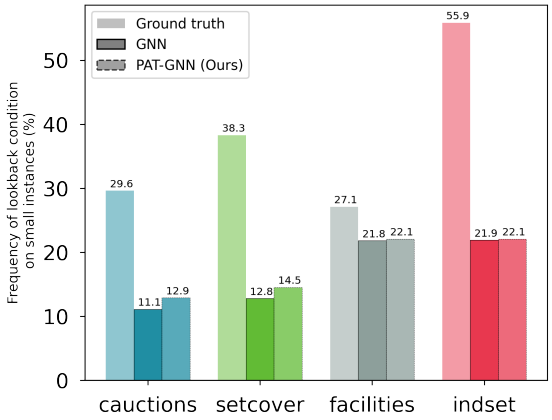

Figure B.2: Statistics of the lookback condition as observed when the problems are solved using the strong branching oracle (Ground truth). We also show how many times traditional GNNs (GNN) and our proposed GNNs (PAT-GNN) respect the lookback condition on the same dataset. Note that the displayed statistics are on the offline dataset and do not reflect the final inference-time performance of the models. We find that the small improvements shown here results in large gains in the inference time performance (see Section 2.5)

B.14 Post-hoc depth-decile analysis of the lookback property

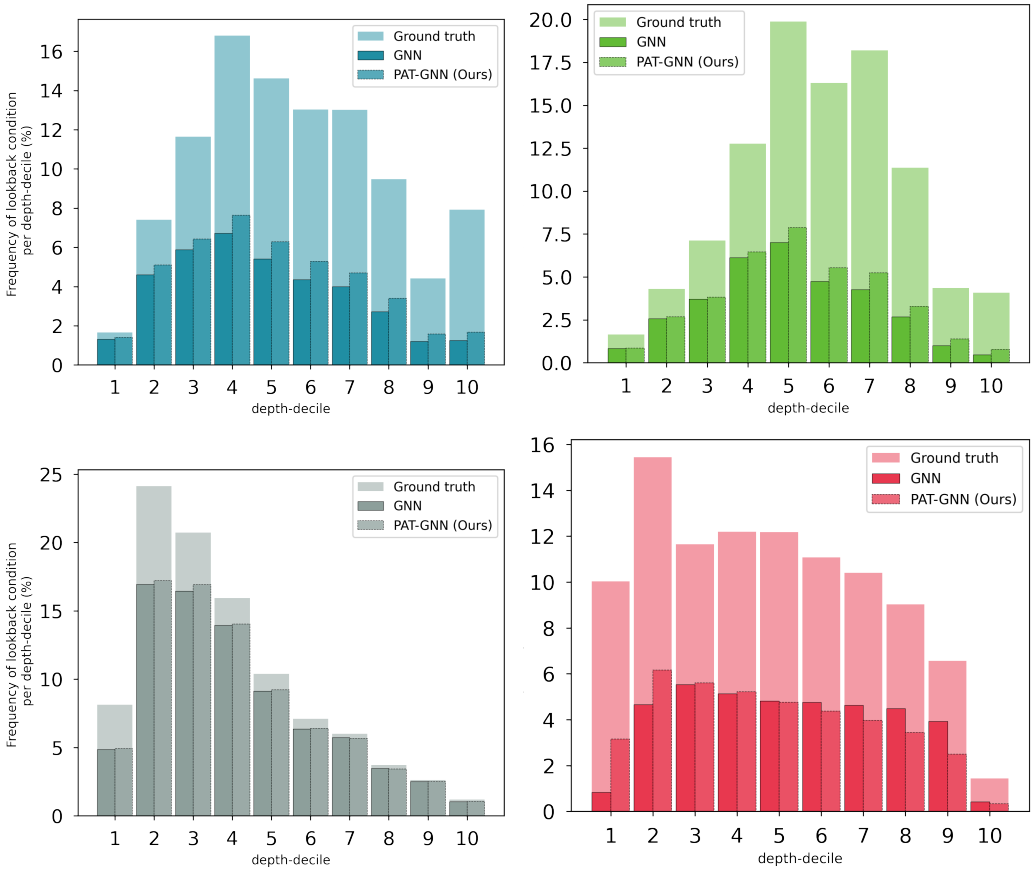

Figure B.3: The gap between the frequency of the lookback condition per *depth-decile* for the strong branching oracle (Ground truth) and the traditionally trained GNNs (GNN) presents an opportunity to improve the GNN models (PAT-GNN). See Section 3.4 for the dataset collection procedure.

B.15 Same strong branching decisions at the sibling nodes

Any MILP solving procedure results in a tree of sub-problems, where each node shares some characteristics with its parent. This line of resemblance can eventually be traced all the way back to the original MILP. For example, the bipartite graph structure remains the same throughout the tree, a fact exploited by Gupta et al. (2020a) to design CPU-efficient GNN-based models for learning to branch.

We investigate more of such dependencies among the B&B tree nodes of problem instances from the benchmark problem families (Gasse et al., 2019) (see Appendix B.4 for more details), which we label as cauctions (Combinatorial Auctions), setcover (Minimum Set Cover), facilities (Capacitated Facility Location),

and indset (Maximum Independent Set).

Specifically, we investigate the frequency with which sibling nodes in the B&B tree have the same strong branching decision. Figure B.4 shows that this condition happens between 3-7% of the times on the small instances that were used to collect approximately 30K observations. The state-of-the-art GNNs (Gasse et al., 2019) are not able to capture this dependency as well. However, given that this condition is fairly less frequent relative to the *lookback* condition, we do not explore this condition in our current work.

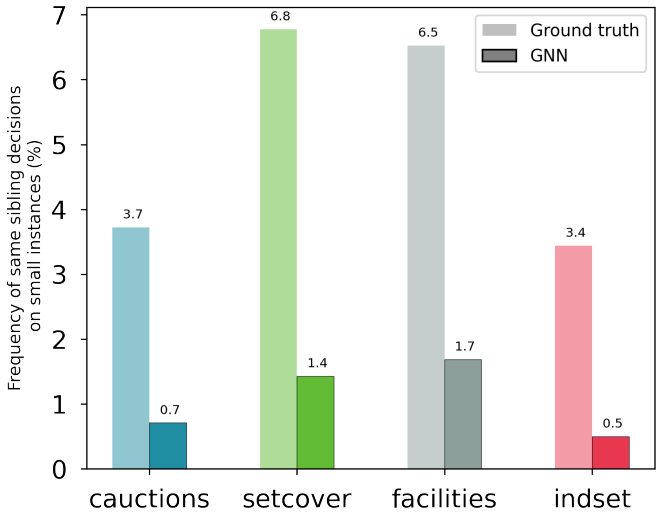

Figure B.4: Frequency with which sibling nodes in a B&B tree have the same strong branching decisions (Ground truth), and the fraction of times this condition is satisfied by GNNs trained as per Gasse et al. (2019).

B.16 Performance comparison with default SCIP

In this section, we provide comparison of our branching strategies with the default SCIP and the tuned version of SCIP following the benchmarking procedure of Nair et al. (2020). But it is important to note that SCIP comes loaded with several heuristics, e.g., cut selection, primal heuristics, restarts, etc. To study the effect of branching strategies, the standard practice in the community is to switch off these heuristics. The results in the main paper are based on this choice. Thus, comparing the branching strategies with default SCIP isn't an apple to apple comparison.

However, it is still useful to see where the proposed branching strategy stand relative to SCIP and some problem-specific tuned version of SCIP. Specifically, we ran a grid search on 3 parameters of SCIP: (a) Presolve, (b) Heuristics, (c) Separating. Each of these parameters can take four values: (a) OFF - do not use it, (b) DEFAULT: keep it at the default level (SCIP without tuning will use this setting), (c) AGGRESSIVE: use it aggressively, and (d) FAST: use it, but do not spend too much time on it. Based on the Eq. 3.9 (T=30mins), we found the following parameters suitable for the problem families: (a) cauctions - (AGGRESSIVE, DEFAULT, OFF), (b) setcover: (DEFAULT, DEFAULT, OFF), (c) facilities: (DEFAULT, DEFAULT, FAST), (d) indset: (FAST, OFF, FAST). Table B.11 shows the performance comparison with SCIP and TunedSCIP. Specifically, we find that the proposed GNN-PAT retains its performance superiority over the default SCIP as well as the tuned version of SCIP.

Table B.11: Performance of *branching strategies* compared with SCIP and TunedSCIP. The best performing numbers are in **bold**. Refer to Section 2.5 for metrics.

Model	Time	Time (c)	Wins	Solved	Nodes (c)
FSB*	n/a	n/a	n/a	n/a	n/a
RPB	626.81	434.92	0	80	17 979
TUNEDRPB	644.20	450.06	0	80	18 104
SCIP	582.47	396.90	4	81	17 979
TUNEDSCIP	581.58	396.98	2	82	19 937
GNN	507.06	333.59	14	80	17 145
GNN-PAT (ours)	**477.26**	**310.22**	**64**	**84**	**16 388**
		Combinatorial Auction (Bigger)			
FSB	2700	n/a	0	0	n/a
RPB	1883.32	1213.57	0	47	58 766
TUNEDRPB	1851.83	1168.58	0	48	58 155
SCIP	1755.15	1048.76	1	54	61 546
TUNEDSCIP	1683.81	963.77	8	52	60 496
GNN	1708.99	991.07	3	54	39 535
GNN-PAT (ours)	**1601.28**	**892.85**	**48**	**59**	**38 385**
		Set Covering			
FSB	917.39	758.19	7	85	50
RPB	737.66	607.19	1	92	104
TUNEDRPB	751.27	619.27	1	91	97
SCIP	677.37	565.91	10	**95**	**86**
TUNEDSCIP	656.72	538.61	18	94	88
GNN	646.03	525.80	14	92	293
GNN-PAT (ours)	**581.91**	**471.20**	**44**	**95**	304
		Capacitated Facility Location			
FSB	2700	n/a	0	0	n/a
RPB	1984.91	865.90	0	32	11 886
TUNEDRPB	2016.85	927.68	0	33	12 486
SCIP	1920.46	784.62	0	37	11 886
TUNEDSCIP	1578.01	443.89	2	48	10 392
GNN	1207.62	262.61	14	65	7233
GNN-PAT (ours)	**1035.32**	**223.81**	**54**	**70**	**5722**
		Maximum Independent Set			

B.17 Results with 10,100-shifted geometric mean of time and nodes

We followed the evaluation protocol proposed by Gasse et al. (2019)(see Evaluation paragraph in Section 5 of Gasse et al. (2019)). Therefore, our results in Tables 3.2 are based on 1-shifted geometric means of nodes and time. However, since Achterberg (2007) studied a much more diverse set of instances with widely varying solving behavior, they use a 100,10-shifted geometric means respectively. We are not in this setting as our instances come from a specific distribution. Nonetheless, in Table B.12 we show the results with 10, and 100-shifted geometric means. Our conclusions will still remain same.

Table B.12: Performance of *branching strategies* on *Big* evaluation instances with 10,100-shifted geometric mean for time and nodes respectively. The best performing numbers are in **bold**. Refer to Section 2.5 for metrics. Since all of the *Big* Combinatorial Auctions instances were solved to optimality, we extend the evaluation to *Bigger* instances (*since only 10 instances were solved using FSB, we omit it here; See Appendix B.11 for the full results including those on *Big* instances.)

Model	Time	Time (c)	Wins	Solved	Nodes (c)
FSB*	n/a	n/a	n/a	n/a	n/a
RPB	633.20	438.94	1	80	18 052
TUNEDRPB	650.24	453.86	0	80	18 169
GNN	515.04	338.33	14	80	17 217
GNN-PAT (ours)	**484.45**	**314.17**	**69**	**84**	**16 448**
Combinatorial Auction (Bigger)					
FSB	2700	n/a	0	0	n/a
RPB	1884.49	1214.20	1	47	58 781
TUNEDRPB	1853.08	1169.20	0	48	58 169
GNN	1710.74	991.84	5	54	39 548
GNN-PAT (ours)	**1603.20**	**893.58**	**54**	**59**	**38 399**
Set Covering					
FSB	917.39	758.19	8	85	84
RPB	740.16	608.88	3	92	157
TUNEDRPB	753.67	620.88	2	91	**149**
GNN	648.72	527.57	16	92	330
GNN-PAT (ours)	**584.60**	**472.93**	**66**	**95**	339
Capacitated Facility Location					
FSB	2700	n/a	0	0	n/a
RPB	1987.27	889.92	0	32	12 443
TUNEDRPB	2019.12	954.35	0	33	12 976
GNN	1215.48	281.93	14	65	8009
GNN-PAT (ours)	**1043.12**	**235.78**	**56**	**70**	**6186**
Maximum Independent Set					

C

Proactive Contact Tracing

Contents

C.1 Decentralized and Centralized versions of PCT

From a privacy perspective, we can view the architecture of the PCT smartphone application as three components:

- **On-device execution of the risk estimation model**: A model (e.g., a set of rules) takes in various features as inputs and outputs risk estimates for that user. The features are of two types - (a) locally available on the smartphone, e.g., symptoms, pre-existing conditions, test results, etc., that are used only with the user's consent. This information never leaves the smartphone, mitigating the risk of leaking personally identifiable information. (b) risk messages from past encounters - the smartphone app manages to send and receive these risk messages according to the network protocol. We discuss this in the third point below.

- **Transmission of the risk estimation model to the device**: The model used for risk estimation is broadcasted to the users at a predetermined frequency. For example, it may be desirable to update the rules every two weeks, in which case a centralised server would broadcast these rules to all the users. This distribution of risk estimation model resembles a decentralised application software in which the application software resides on the user's smartphone. This is in contrast to a centralised software application in which the user sends the query to a centralised server, thereby increasing the risks of information leaks.

- **The communication protocol used to exchange risk messages**: Network protocols lie on a spectrum of centralisation and decentralisation, with privacy-preserving decentralised protocols being preferable for preserving individual rights for contact tracing applications. However, centralisation can be useful for healthcare authorities to monitor the spread of disease and allocate resources.

  The choice of the **communication protocol** used to send risk messages has many implications for user privacy. For a detailed discussion of the pros

and cons of various choices, please see section 2 of Alsdurf et al. (2020). Here, we will provide a simplified description of the communication protocol relying on two components. First, we must establish a **contact protocol** that enables two smartphones, when brought into close proximity, to allow a future exchange of risk messages. Second, an **update protocol** that enables the secure transmission of updated risk estimates between these devices. We now present a centralised and decentralised design for each protocol.

The **contact protocol** can be implemented in two ways -

- **A more decentralised contact protocol**, such as that used in GAEN, has two smartphone application users exchange an identifier when they come in close proximity. This is enabled via Bluetooth Low-Energy (BLE) devices. These identifiers are cycled frequently (on the order of 15 minutes) and are only transmitted to nearby devices through Bluetooth.

- **A more centralised contact protocol** may rely on each application to record the GPS coordinates at every fixed interval (e.g., 5 minutes). These coordinates are sent to a central server at a predetermined interval (e.g., 6 hours). The central server runs a matching algorithm that takes in as input the GPS coordinates of all the users and determines which were in close proximity.

The ultimate aim of the **contact protocol** is to generate an identifier or a key (e.g., hash) that can enable future communication between these matched smartphones. The **update protocol** leverages information exchanged during the contact protocol to inform the users of new risk estimates based on their past encounters. The **update protocol** can be more or less centralised. For example -

- **Some more decentralised update protocols** distribute a database of identifiers to all user devices. Each device searches this database and extracts risk messages using identifiers corresponding to past encounters exchanged during the **contact protocol**. Examples of this protocol are Google/Apple Exposure

Notification system (GAEN) and Temporary Contact Number (TCN). Please refer to sections 2.3.1 and 2.3.2 of Alsdurf et al. (2020).

- **A more centralised update protocol** to send risk messages may rely on the user to send the identifiers received during the contact protocol to the central server. This server returns the risk messages associated with these identifiers, allowing a centralised authority to gather statistics about the distribution of risks and contacts. An example of this protocol is NHS Bluetooth + mix nets, although the software code supports a decentralised version as well. Please refer to section 2.3.3 of Alsdurf et al. (2020).

We leave the discussion around the privacy implications of the above choices to Alsdurf et al. (2020) (please see section 2).

C.2 Agent-based model

Demographics. We sample agent demographics corresponding to the region of Montréal using the census data from Statistics Canada[1]. Specifically, we consider age distribution, statistics on house sizes ranging from 1 to 5, distribution of individual age living alone, and for house sizes ranging from 2 to 5 distribution of the following dwelling characteristics - (a) couple with x kids, (b) single parent with x kids, and (c) random allocation, where x represents number of kids required to complete the house size[2]. Finally, we also consider senior residencies where a proportion of agents above age 65 live. We inform this proportion from the census data as well.

The prevalence of selected medical conditions considered as risk factors for COVID-19 disease and severe disease progression were derived from nationally representative surveys and medical surveillance programs in Canada. The following conditions were considered, each with corresponding data sources: heart

[1]Census Profile (2016), Montréal, Québec, Canada. Statistics Canada. Available from: `https://www12.statcan.gc.ca/census-recensement/2016/dp-pd/prof/details/page.cfm?Lang=E&Geo1=CMACA&Code1=462&Geo2=PR&Code2=01&Data=Count&SearchText=montreal&SearchType=Begins&SearchPR=01&B1=All&TABID=1`. Accessed: 2020-06-02.

[2]We refer the reader to `https://github.com/mila-iqia/COVI-AgentSim/blob/master/src/covid19sim/configs/simulation/region/montreal.yaml` for a fully referenced source of the above parameters.

disease (Petrilli et al., 2020; Williamson et al., 2020; Killerby et al., 2020)[3], stroke (Williamson et al., 2020), asthma (Petrilli et al., 2020; Williamson et al., 2020), chronic obstructive pulmonary disease (COPD) (Petrilli et al., 2020; Williamson et al., 2020; Killerby et al., 2020), cancer (Petrilli et al., 2020; Williamson et al., 2020; Robilotti et al., 2020), diabetes (Petrilli et al., 2020; Williamson et al., 2020; Killerby et al., 2020)[4], obesity (Petrilli et al., 2020; Williamson et al., 2020; Killerby et al., 2020; Bello-Chavolla et al., 2020)[5], chronic kidney disease (CKD) (Petrilli et al., 2020; Williamson et al., 2020; Killerby et al., 2020; Arora et al., 2013), immuno-suppressed conditions (Williamson et al., 2020) and smoking[6][7]. Sex- and age-specific (according to ten year intervals) prevalence estimates were used where data was available. Estimates of age-stratified asymptomatic proportion were obtained from (Davies et al., 2020).

Mobility. Given an agent population with a designated housing and a workplace (schools for younger population), we assign a mobility schedule to each agent which takes them from one location to another. We use miscellaneous locations to emulate random and relatively shorter interactions typical of dynamics at stores, parks, or gyms. Out of the three broad types of locations, agents spend the majority of their time at their house and workplace. The scheduler is designed to respect constraints, such as, agents younger than 15 years of age are never alone at home, or schedules are dynamically altered to cancel an activity in the event of sickness. We used Statistics Canada data to derive the amount of time spent in each activity and

[3]Health data from the Canadian Chronic Disease Surveillance System, 2018. Public Health Agency of Canada. Available from: `https://www.canada.ca/en/public-health/services/publications/diseases-conditions/report-heart-disease-Canada-2018.html`. Accessed: 2020-08-18.

[4]Diabetes in Canada, 2017. Public Health Agency of Canada. Available from: `https://www.canada.ca/en/public-health/services/publications/diseases-conditions/diabetes-canada-highlights-chronic-disease-surveillance-system.html`. Accessed: 2020-08-18.

[5]Overweight and obesity based on measured body mass index, by age group and sex. Statistics Canada. Available from: `https://www150.statcan.gc.ca/t1/tbl1/en/tv.action?pid=1310037301`. Accessed: 2020-09-30

[6]Smoking, 2018. Statistics Canada. Available from: `https://www150.statcan.gc.ca/n1/pub/82-625-x/2019001/article/00006-eng.htm`. Accessed: 2020-09-30

[7]Smoking and COVID-19, 2020. World Health Organization. Available from: `https://www.who.int/news-room/commentaries/detail/smoking-and-COVID-19`.

the frequency at which they occur (e.g., every three days). Note that due to the complexities involved in modeling the public transport systems (e.g., large-scale routing and scheduling), our current model lacks such sophistication and, therefore, only accounts for infections happening at locations rather than in the transits.

While the presence of agents at house, workplace, and school is deterministic, to schedule activities like going to cafes, grocery stores, or gyms (collectively termed as random locations) we resorted to probabilistic sampling. Specifically, we used the data for Canada-wide population from Statista and Statscan to inform number of days in which humans are likely to repeat their visit to these random locations. This resulted in mean daily contacts per age group at various locations shown in Figure C.1.

Contact Patterns. We implement age-stratified contact sampling. Specifically, for each agent we draw number of contacts as per the location-specific age-stratified number of contacts obtained from the contact matrices. We consider the age groups of 5 years. The contact matrices are derived by projecting empirically derived matrix for Canada (Prem et al., 2017) on the region of Montréal using demographic standardization (Arregui et al., 2018) and further adjustment to the reported regional mean contacts E_l (see Table C.1). Thus, a cell i, j in the matrix denote mean daily contacts made by the agents in age group i with agents in age group j. We use a negative binomial distribution to draw the number of contacts with mean as inferred from the contact matrix and the probability of success as 0.5. Further, we use these matrices to infer probability of interaction with other agents in each age group, thereby, implementing location dependent assortativity in interactions.

The simulated contact patterns with no infections are shown in Figure C.2 for overall contacts aggregated across all types of locations. These were obtained by aggregating contacts across 6 simulations with no initial infections, thereby resulting in pre-pandemic contact patterns.

Similarly, Figure C.3 shows simulated contact patterns for contacts at houses. As a result of housing allocation discussed above, we make two observations (a)

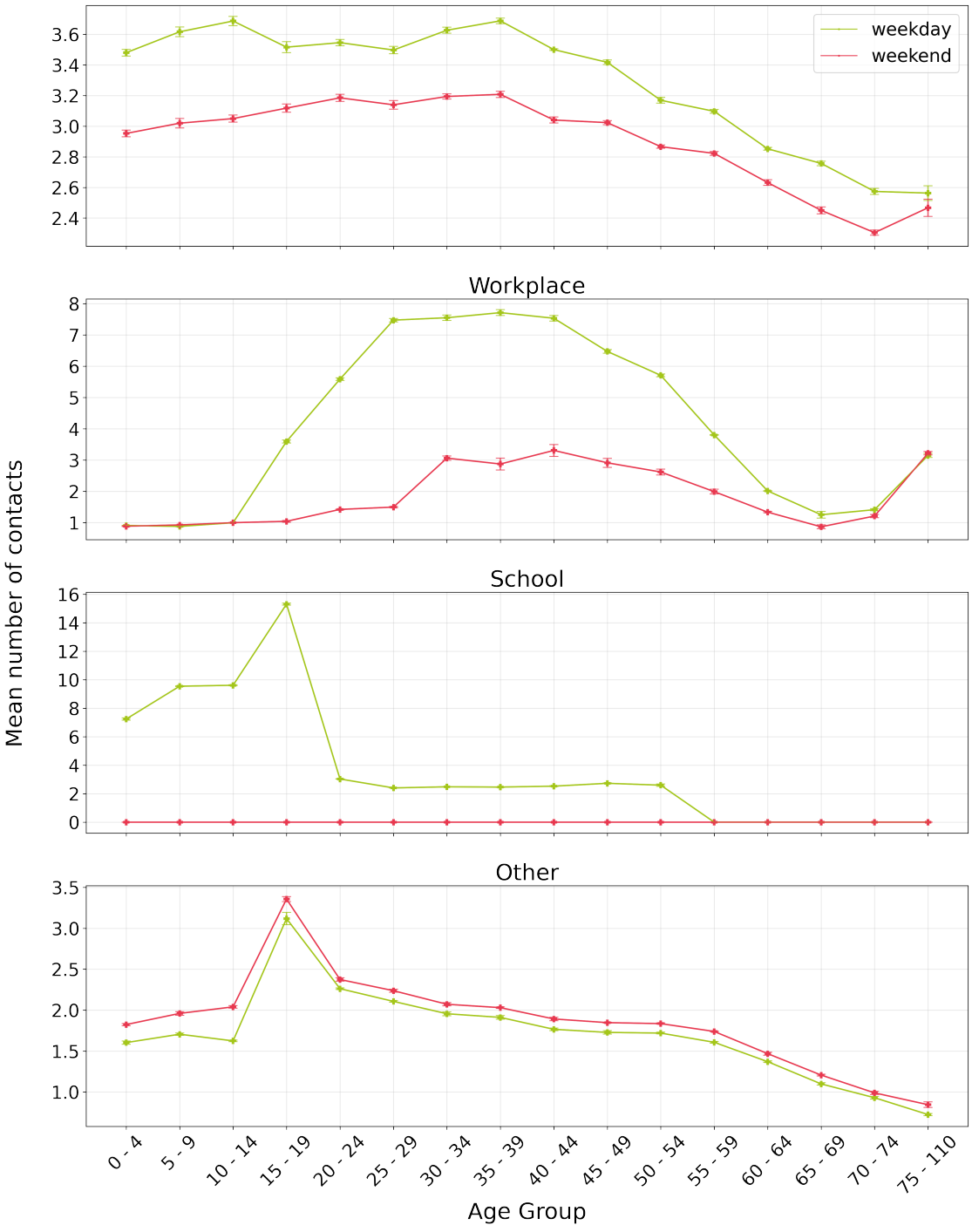

Figure C.1: Simulated mean daily contacts on weekdays and weekends broken down by age groups. Agent activities are scheduled such that the mean number of contacts on work and non-work days follow surveyed data as reported in (Klepac et al., 2020; Mossong et al., 2008)

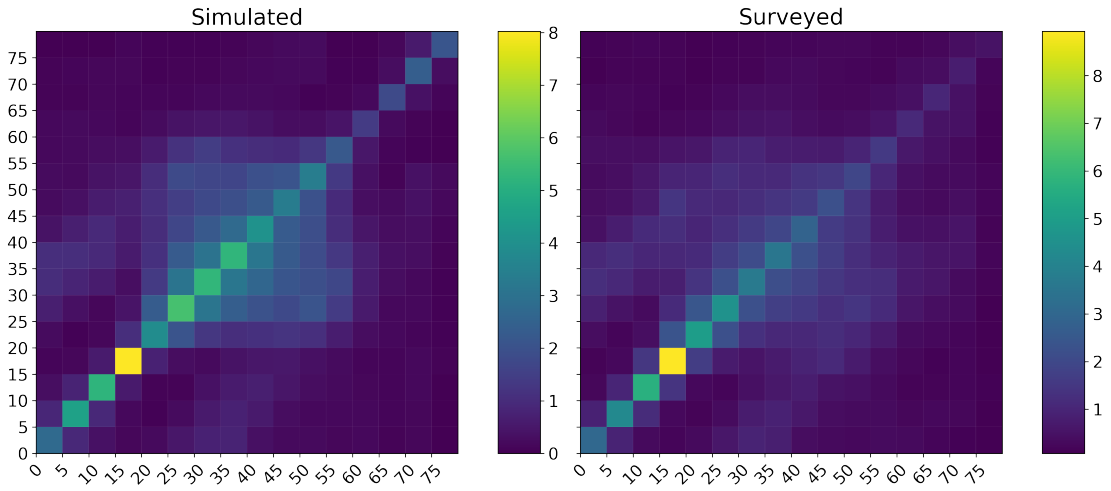

Figure C.2: Contact Matrices for all locations: Overall simulated contact pattern (left) yield a similar pattern to empirically derived matrix (right).

there is an oversampling of contacts towards the older age groups: It is because older agents grouped in collectives like senior residencies are modelled explicitly. This choice was motivated from Zimmerman et al. (2020) which suggests inclusion of collectives in proper response to the COVID-19 pandemic, (b) a slight discrepancy we observe in the intensity of the main diagonal is due to insufficient social gatherings at households.

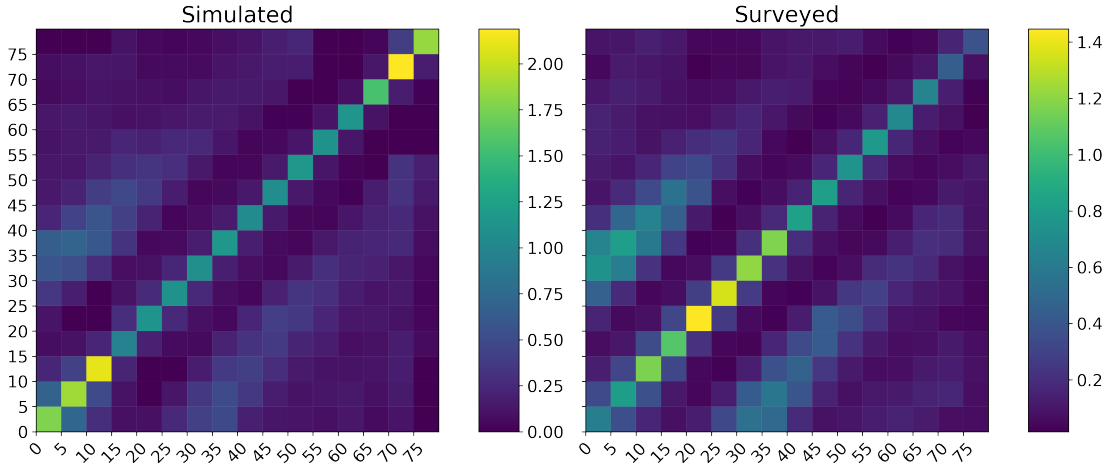

Figure C.3: Contact Matrices for House: Housing allocation of agents is calibrated to yield a contact pattern (left) similar to the empirically derived household contact matrices (right). We explicitly model older adults living in assisted care resulting in oversampling of contacts in that age group.

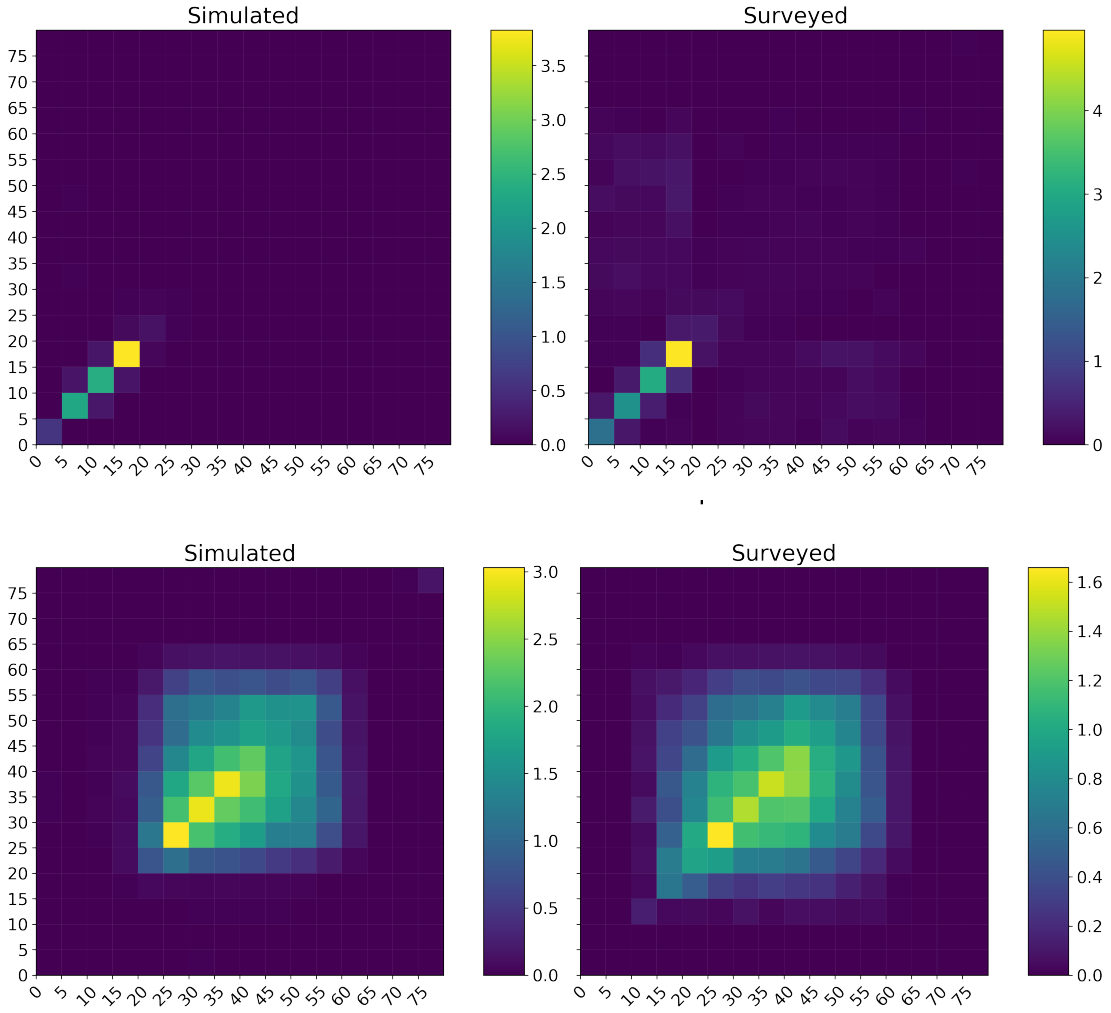

Figure C.4: Contact Matrices for School (top) and Workplaces (bottom): Simulated contact pattern at schools (left) yield a similar pattern to empirically derived matrix (right).

Further, Figure C.4 shows contact patterns for contacts at schools and workplaces, while the Figure C.5 shows contact patterns for contacts at miscellaneous locations. Finally, Figure C.6 shows the distribution of mean daily contacts across a population of 10000 agents in a typical simulation without any infection. As expected, we see the presence of agents who sample disproportionately more number of contacts than the average population.

Behavior modification in intermediate behavior levels Table C.1 summarizes the location dependent reduction factor α_l for various levels. For the sake of

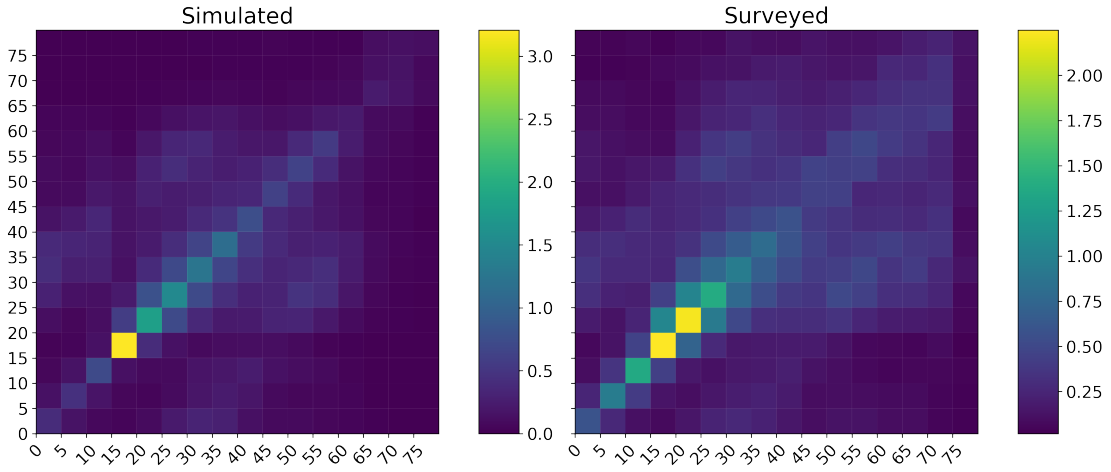

Figure C.5: Contact Matrices for “other" locations: Simulated contact pattern at "Other" locations (left) yield a similar pattern to empirically derived matrix (right). Agents spend relatively shorter duration of time at these locations as compared to house or workplaces. Examples of such locations include parks, restaurants, and grocery stores. We observe a slight oversampling of such contacts due to improper estimates of time spent at such activities.

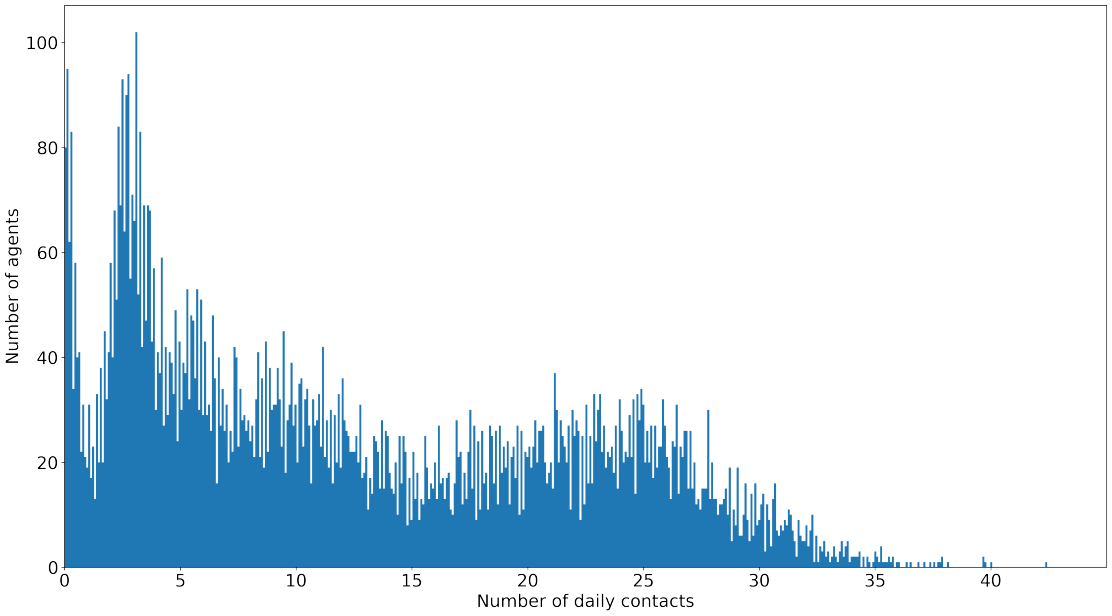

Figure C.6: Distribution of daily contacts: The simulation was run with 10000 agents over a period of 30 days with no infection. As an emerging behavior, we observe the agents which sample too many contacts as compared to the average number of contacts (12.62).

simplicity, we restrict agent behavior levels to one of the top four levels at any point in time. Our choice of using four recommendation levels is motivated by the four-tier measures adopted by Quebec's government[8], though the method is compatible with any number of recommendation levels.

Table C.1: Daily pre-confinement mean contacts E_l, and reduction factor α_l per location type.

Location	E_l	Level 1 α_l	Level 2 α_l	Level 3 α_l	Level 4 α_l
Household	2.7	0.0675	0.135	0.27	1
Workplace	10	0.18	0.36	0.72	1
School	6	0.18	0.36	0.72	1
Other	3.1	0.1125	0.225	0.45	1

Table C.2: This table shows for each location l the pre-confinement mean number of daily contacts E_l and the reduction factor for the number of contacts used in the simulator. Note that the factor for level 3 is based on data collected in the Region of Montréal (Brisson et al., 2020)

We scale the age-stratified contact matrices with the reduction factor corresponding to the agent's behavior level. Note that in non-PCT scenarios, agents are either in level 1 or level 4. PCT, on the other hand, puts each agent in one of the four levels, based on their estimated risk of infectiousness, mimicking the effect of cautionary behavior reinforcement via recommendations as experimented by Ayres et al. (2020). The resulting pre-pandemic contact patterns are compared to the empirically derived matrices in the supplement. The mean number of contacts, sampled by age group, location type, and day of the week, demonstrate similar trends as reported in the BBC Pandemic Project (Klepac et al., 2020), a recent POLYMOD matrix-based study conducted using smartphone apps.

User compliance to the recommendations is unarguably the most important factor determining the effectiveness of any contact tracing method. For the sake of simplicity, in our simulations, varying levels of behavior restrictions are modeled by a fractional reduction relative to the pre-pandemic number of contacts. However, in its real-life implementation, the app would recommend personalized behaviors (e.g. "avoid public transportation"). These need to be deemed acceptable by app

[8] `https://www.quebec.ca/en/health/health-issues/a-z/2019-coronavirus/`

users to achieve high adherence and actualize such reduction in contacts. We direct the readers to the guidelines in Alsdurf et al. (2020) for designing such recommendations in close collaboration with user-behavior researchers.

COVID-19 Transmission Model The transmission probability is modeled according to Ferretti et al. (2020), in which the likelihood of transmission is proportional to age-dependent susceptibility (S_a) of the susceptible agent at age a. The probability of transmission also depends on a location-dependent multiplicative factor B_n (for location n), a symptom status (asymptomatic, mild, severe) dependent ratio (A_s) of the infectious agent, and Effective Viral Load, a surrogate for the cumulative viral load (EVL), transmitted from the infectious agent for the duration of exposure time (δt). A multiplicative factor (r) is finally used to calibrate the reproductive number of the infection spread. The formula for the transmission model is presented in Equation C.1 and Equation C.2 below.

$$\lambda(\delta t, S_a, A_s, n) = \frac{r S_a A_s B_n}{\bar{I}} \int_{\delta t} EVL, \tag{C.1}$$

$$P(\delta t, S_a, A_s, n) = 1 - e^{-\lambda(\delta t, S_a, A_s, n)}, \tag{C.2}$$

where $P(\delta t, S_a, A_s, n)$ is the probability of the contagion event. The values for the above constants are directly used from the open source code[9] of Ferretti et al. (2020) .

Effective Viral Load We sample parameters for a piece-wise linear model of what we call **effective viral load** (EVL)[10]. The attributes of EVL are sampled for each individual separately. Figure C.7 shows a typical shape of EVL motivated from the results in Goyal et al. (2020), and the attributes' mean and standard deviation follow from He et al. (2020); To et al. (2020); Lauer et al. (2020).

[9] `https://github.com/BDI-pathogens/OpenABM-Covid19/blob/master/documentation/parameters/infection_parameters.md`

[10] Viral load is the number of actual viral RNA in a person; we model a number between 0-1 which could be converted to an actual viral load via multiplying by the maximum amount of viral RNA detectable by a given test.

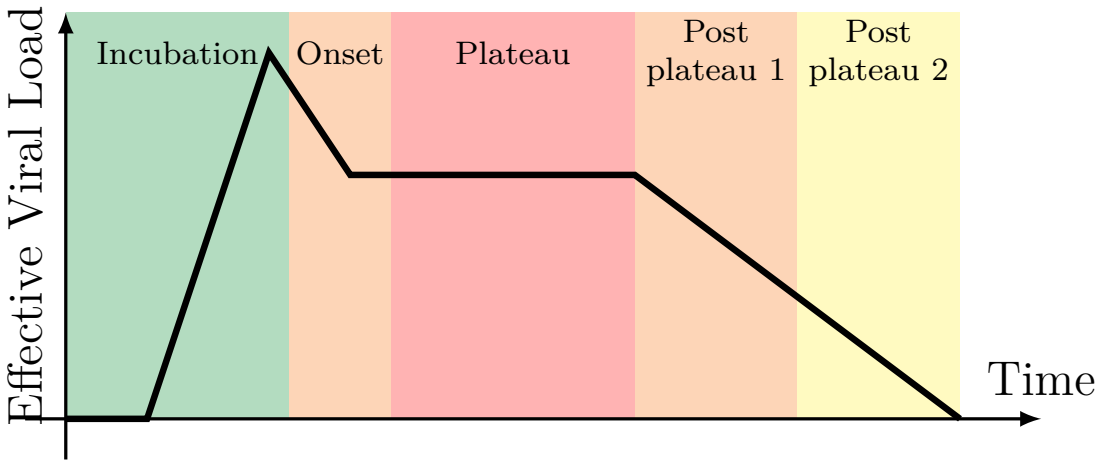

Figure C.7: Schematic showing the viral load curve, and associated phases of symptoms with severity indicators: infectiousness onset occurs on average 2.5 days after exposure, viral load peaks 0.7 days before symptom onset, which occurs an average of 5 **incubation days** after exposure (He et al., 2020). Symptoms are most severe after viral load peak and symptom onset, when the virus has had time to infect many cells. Recovery takes on average 14 days from symptom onset. We use a truncated normal distribution to sample the parameters with the above specified mean values.

Figure C.8 is the mean of sampled effective viral load curve. We integrate over individual viral load curve to determine the likelihood of transmission $\lambda(\delta t, S_a, A_s, n)$ as described in Equation C.1.

Real-time Polymerase Chain Reaction (RT-PCR) Testing We model the number of available tests as a scarce resource with a fixed number of tests available on any given day, with a delay of two days between testing and when the agent receives the result. These choices are motivated by the challenges in obtaining COVID-19 diagnosis at the pandemic's onset in March 2020. We use a triaging mechanism for tests; an agent experiencing severe symptoms is prioritized over the agents estimated to be in level 4 by a tracing app. Because of diagnostic testing's known performance challenges, a SARS-CoV-2 infection phase-dependent false-negative rate is applied for the RT-PCR results (Li et al., 2020).

When an agent gets an RT-PCR test, the agent, as well as other household members, are put into level 4 either (1) until the time of the result (two days) for a negative test result; or (2) for 14 days from the day of the test result for a positive result. An agent with an app can choose to report their test result

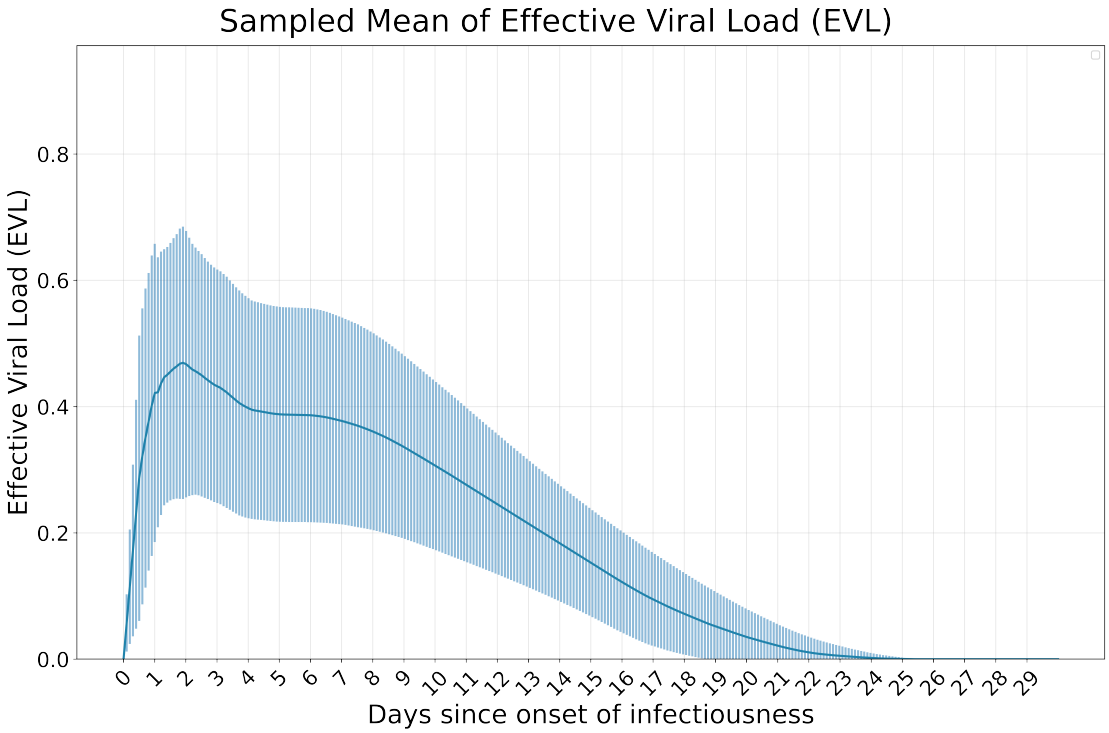

Figure C.8: Mean ± 95% C.I of sampled effective viral load of 40 agents.

triggering a notification to their digital contacts. In BCT, such a traced contact and its household members are automatically quarantined for 14 days unless they test negative. To achieve a similar effect of household quarantining in PCT, we require agents to assume the residents' maximum behavior level.

Imperfect agent behavior We use the dropout probabilities as used by Ferretti et al. (2020), such that an agent recommended to quarantine will be 0.01 likely to dropout to the level 1 and sample contacts as per that level. Further, the simulator does not assume all personal information (i.e., age, sex, and pre-existing conditions) will be entered or logged correctly when downloading an app. Instead, the simulator assumes a 70% likelihood that the agent will log their characteristics correctly. Also, an agent is assumed to be 80% likely to report their RT-PCR test results if they receive a positive test. Imperfect self-reporting of daily symptoms is modeled via two parameters (i) drop-in - the probability of falsely reporting symptoms (to account for malicious behavior), and (ii) dropout - the probability of not reporting a symptom.

App adoption To distribute apps throughout our simulated population, we use an age-dependent distribution across smartphone owners. We use an age-based breakdown of smartphone users as in Ferretti et al. (2020), and use an $UPTAKE$ parameter to vary the population-level adoption rate.

Digital Communication PCT sends risk messages assuming the COVI network communication protocol (Alsdurf et al., 2020). This protocol is designed to conform to strict privacy constraints in the following manner: (a) to prevent identification of users, a unique key is exchanged every time two users are within 2 meters of each other for at least 15 minutes; the unique key is then used as a communication channel to exchange risk messages in the future, and (b) risk messages can only carry N bits, constraining their values to one of 2^N values, dramatically reducing the likelihood of de-anonymization. In our experiments, we use $N = 4$ meaning that apps can send risk messages containing a non-negative integer less than 16. As the risk messages form both the input as well as the output of the predictor, we note that the proposed tracing framework is implicitly recursive (i.e. infectiousness of a user is indirectly influenced by all other users in the network).

C.3 Technical Details of Rule-Based PCT

The Rule-Based PCT algorithm takes as input several types of features such as symptoms, test results, and contacts with risky individuals. Based on these features, the algorithm produces an estimate of the user's infectiousness on that day and over the previous 13 days. More abstractly, RB-PCT is a heuristic function which maps a two-dimensional integer array representing 14 days of features related to the likelihood of having or contracting COVID into a single-dimensional array of quantized risk levels.

Useful Notation To provide a formal description of the RB-PCT algorithm, we now define some terms. In this work, we denote the set of days $\{d_1, d_2, ...d_{max}\}$ as

D, where d_{max} is 14. We denote agent i's behavior level on day d by ζ_d^i such that, $\zeta_d^i \in \{1, 2, 3, 4\}$.

Determination of this recommendation level is done via a risk estimator that uses a rich suite of features to evaluate agent i's risk history on day t. That risk history is denoted by $\mathbf{r}_t^i$, where we constrain risk levels to be non-negative integers with a maximum value of r_{MAX}. Thus, $\mathbf{r}_t^i \in \{0, 1, 2, ...r_{\text{MAX}}\}^{d_{max}}$ We use $r_{t,d}^i$ to denote estimated risk of agent i for day d such that $t - d_{max} < d < t$. If an agent i had an encounter with an agent j on day d such that $t - d < d_{max}$, $r_{t,d}^i$ is sent to the past contacts as a risk message $M_{i,j}^d(t) = r_{t,d}^i$ if j was a contact of i on day d. This determines what information is available to agent's contacts such that they may modify their own behavior or to propagate risk further.

We use $\mathcal{N}(i)$ to denote a set of agent indices that had at least one digitally recorded contact with agent i in the past d_{max} days. Further, we use the colon symbol (:) to iterate through all possibilities for the variable at that position. For example, $M_{i,:}^d(:)$ represents risk messages from agent i to all agents $j \in \mathcal{N}(i)$ for encounter on day d, if there was one. Similarly, $M_{i,:}^i(:)$ represents risk messages sent by agent i to the agents in $\mathcal{N}(i)$ within the past d_{max} days.

We denote agent i's test result on day d by $\text{test}_d^i \in \{+1, 0, -1\}$, where $+1$ denotes a positive test result, -1 denotes a negative test result, and 0 denotes no test was taken. As a simplifying assumption, we assume that a positive test is retained for d_{max} days while a negative test is retained for d_{min} days (we use $d_{min} = 2$ in our experiments). Thus, an agent i getting a negative test result on day d i.e. $\text{test}_d^i = -1$ will have this variable set to -1 for the next two days i.e. $test_{d+1}^i = -1$, and $test_{d+2}^i = -1$, after which, in the absence of any test $test_{d+3}^i = 0$. We further denote a history of test results for d_{max} days in the past by a vector $\mathbf{T}_d^i \in \{+1, 0, -1\}^{d_{max}}$. We refer to the test result on day d' (where $d - d_{max} < d' \leq d$) as $\mathbf{T}_{d,d'}^i = \text{test}_{d'}^i$ as known on day d.

We similarly denote agent i's symptoms for the past d_{max} days by a matrix $\mathbf{S}_d^i$, such that, with $N_{symptoms}$ categories of symptoms, $\mathbf{S}_d^i \in \{0, 1\}^{N_{symptoms} \times d_{max}}$.

Additionally, agent i's symptoms on day $d^{'}$, where $d - d_{max} < d^{'} \leq d$, are represented via $\mathbf{S}_{d,d'}$.

The users are shown multiple options in the app to indicate their daily symptoms. These options are designed and worded by the behavior and epidemiology experts. After discussing with these experts and several iteration of the heuristic, we categorized the symptoms into three categories according to the amount of information that they carry for the diagnosis of COVID-19 infection. The three categories are as follows,

- Highly informative symptoms, S_{HIGH}: severe, extremely severe, chills, unusual, headache, confused, lost consciousness, trouble breathing, sore throat, severe chest paing, loss of taste, light trouble breathing, moderate trouble breathing, heavy trouble breathing
- Moderately informative symptoms, $S_{MODERATE}$: diarrhoea, nausea vomiting, cough, hard time waking up
- Least informative symptoms, S_{LOW}: mild, moderate, fever, gastro, sneezing, runny node, ache, fatigue

Algorithm The *Rule-based PCT* is implemented as Algorithm 1. Here, we describe each function of the *Rule-based PCT* [11].

- **Compute Risk** calls each of the below functions in order to generate a risk history over the last d_{max} days and a current recommendation level. It takes in symptom, test, and risk message data, then takes the element-wise maximum over the risk histories associated with these inputs. *Apply Negative Test* and *Handle Recovery* modify the merged risk history or return an alternative (respectively) to this risk history.
- **Test Results Compute Risk** covers two cases. A positive test result within the last d_{max} days sets the risk over the past d_{max} days to r_{MAX} and ζ_d^i to 4.

[11] Please look at our publicly available code (`http://bit.ly/3xidc9z`), for a more detailed technical description of *Rule-based PCT* algorithm.

- **Symptoms Compute Risk** determines the risk and recommendation levels due to symptoms. We group symptoms by how informative they are of COVID-19. Highly informative symptoms experienced within the last $d - d_{max}/2$ days sets high risk levels into $\mathbf{r}^i_{d:d-d_{max}/2}$, and ζ^i_d to 4. Moderately informative symptoms yield moderate risk levels, and so on.

- **Risk Messages Compute Risk** converts received risk messages into a risk history and recommendation level. We group risk messages into high, medium, and low risk groups, assigning corresponding risk levels to the i's risk history between the day of receipt of that class of risk message and the current day.

- **Handle Recovery** returns the recent risk history to 0 in the absence of recent evidence of COVID-19. If there are no symptoms within the last 7 days, no recent risk messages, and no positive test results, then we set risk for the past $d_{max/2}$ days as 0.

- **Apply Negative Test** overwrites the element-wise maximum over the risk histories r^i_d by assigning a 0 risk to the agent around the days when a negative test was reported. Precisely, we denote by W a time window, $W \in \mathcal{N}$ and $W < d_{max}$, such that we set a 0 risk for the agent for $W/2$ days around the day on which negative test was reported ($W = 7$ in our experiments).

Algorithm 1A *Rule-based PCT*

function TestResultsComputeRisk($\mathbf{T}_d^i$)
  $\mathbf{r} \leftarrow \{0\}^{d_{max}}$
  $\zeta \leftarrow 0$
  if $\sum_{d' \in D} \mathbb{1}_{\{\mathbf{T}^i_{d,d'}=+1\}} \geq 1$ **then**
    $\mathbf{r}_{d:d-d_{max}} \leftarrow r_{\text{MAX}}$
    $\zeta \leftarrow 4$
  end if
  return $\mathbf{r}$, ζ
end function

function SymptomsComputeRisk($\mathbf{S}_d^i$)
  $\mathbf{r} \leftarrow \{0\}^{d_{max}}$
  $\zeta \leftarrow 0$
  if $\sum_{d' \in D,\, j \in S_{\text{HIGH}}} \mathbf{S}^i_{d,\{j,d'\}} \geq 1$ **then**
    $\mathbf{r}_{d:d-d_{max}/2} \leftarrow r_{\text{HIGH}}$
    $\zeta \leftarrow 4$
  else if $\sum_{d' \in D,\, j \in S_{\text{MOD}}} \mathbf{S}^i_{d,\{j,d'\}} \geq 1$ **then**
    $\mathbf{r}_{d:d-d_{max}/2} \leftarrow r_{\text{MOD}}$
    $\zeta \leftarrow 3$
  else if $\sum_{d' \in D,\, j \in S_{\text{MILD}}} \mathbf{S}^i_{d,\{j,d'\}} \geq 1$ **then**
    $\mathbf{r}_{d:d-d_{max}/2} \leftarrow r_{\text{MILD}}$
    $\zeta \leftarrow 2$
  end if
  return $\mathbf{r}$, ζ
end function

function RiskMessagesComputeRisk($M^i_{i,:}(:)$)
  $\mathbf{r} \leftarrow \{0\}^{d_{max}}$
  $\zeta \leftarrow 0$
  if $\sum_{j \in \mathcal{N}(i), d' \in D, d'' \in D} \mathbb{1}_{\{M^{d'}_{i,j}(d'')=r_{\text{MAX}}\}} \geq 1$ **then**
    $\hat{d} \leftarrow$ Earliest day of receiving r_{MAX}
    $\mathbf{r}_{d:\hat{d}} \leftarrow r_{\text{MODERATE}}$
    $\zeta \leftarrow 2$
  else if $\sum_{j \in \mathcal{N}(i), d' \in D, d'' \in D} \mathbb{1}_{\{M^{d'}_{i,j}(d'')=r_{\text{HIGH}}\}} \geq 1$ **then**
    $\hat{d} \leftarrow$ Earliest day of receiving r_{HIGH}
    $\mathbf{r}_{d:\hat{d}} \leftarrow r_{\text{MILD}}$
    $\zeta \leftarrow 1$
  else if $\sum_{j \in \mathcal{N}(i), d' \in D, d'' \in D} \mathbb{1}_{\{M^{d'}_{i,j}(d'')=r_{\text{MOD}}\}} \geq 1$ **then**
    $\hat{d} \leftarrow$ Earliest day of receiving r_{MOD}
    $\mathbf{r}_{d:\hat{d}} \leftarrow r_{\text{MILD}}$
    $\zeta \leftarrow 1$
  end if
  return $\mathbf{r}$, ζ
end function

Algorithm 1B *Rule-based PCT*

function HANDLERECOVERY($\mathbf{S}^i_d$, $\mathbf{T}^i_d$, $M^i_{i,:}(:), \mathbf{r}^i_d$)
  Rx $\leftarrow 1$
  $D_7 \leftarrow \{d, d-1, ...d-7\}$
  $D_4 \leftarrow \{d, d-1, ...d-4\}$
  $D_1 \leftarrow \{d, d-1\}$
  if $\sum \mathbf{S}^i_{d,\{:,d:d-d_{max}/2\}} \geq 1$ **then**
    Rx $\leftarrow 0$
  end if
  if $\sum_{d' \in D} \mathbb{1}_{\{\mathbf{T}^i_{d,d'}=+1\}} \geq 1$ **then**
    Rx $\leftarrow 0$
  end if
  if $\sum_{j \in \mathcal{N}(i),\, d' \in D,\, d'' \in D_7} \mathbb{1}_{\{M^{d'}_{i,j}(d'')=r_{\text{HIGH}}\}} \geq 1$ **then**
    Rx $\leftarrow 0$
  else if $\sum_{j \in \mathcal{N}(i),\, d' \in D,\, d'' \in D_4} \mathbb{1}_{\{M^{d'}_{i,j}(d'')=r_{\text{MOD}}\}} \geq 1$ **then**
    Rx $\leftarrow 0$
  else if $\sum_{j \in \mathcal{N}(i),\, d' \in D,\, d'' \in D_1} \mathbb{1}_{\{M^{d'}_{i,j}(d'')=r_{\text{MILD}}\}} \geq 1$ **then**
    Rx $\leftarrow 0$
  end if
  if Rx $= 1$ **then**
    $\mathbf{r}^i_{d,d:d-d_{max}/2} \leftarrow 0$
  end if
  return $\mathbf{r}^i_d$, Rx
end function

function APPLYNEGATIVETEST($\zeta^i_d, \mathbf{r}^i_d, \mathbf{T}^i_d, W$)
  $d_n \leftarrow$ day of the latest negative test
  $\mathbf{r}^i_{d,\, d_n - W/2\, :\, d_n + W/2} \leftarrow 0$
  if $\mathbf{r}^i_{d,d} = 0$ **then**
    $\zeta^i_d = 0$
  end if
  return $\mathbf{r}^i_d$, ζ^i_d
end function

Algorithm 1C *Rule-based PCT*

function ComputeRisk($\mathbf{T}_d^i, \mathbf{S}_d^i, M_{i,:}^{:}(:), \mathbf{X}_i, \mathbf{r}_{d-1}^i$)
  $W \leftarrow 8$
  $\mathbf{r}_t^i, \zeta_t^i \leftarrow$ TestResultsComputeRisk($\mathbf{T}_d^i$)
  $\mathbf{r}_s^i, \zeta_s^i \leftarrow$ SymptomsComputeRisk($\mathbf{S}_d^i$)
  $\mathbf{r}_m^i, \zeta_m^i \leftarrow$ RiskMessagesComputeRisk($M_{i,:}^{:}(:)$)
  $\mathbf{r}_r, \mathrm{Rx} \leftarrow$ HandleRecovery($\mathbf{S}_d^i$, $\mathbf{T}_d^i$, $M_{i,:}^{:}(:), \mathbf{r}_{d-1}^i$)
  if $\mathrm{Rx} = 1$ **then**
    return $\mathbf{r}_r$, 0
  end if
  $\mathbf{r}_d \leftarrow \max(\mathbf{r}_t, \mathbf{r}_s, \mathbf{r}_m, \mathbf{r}_{d-1})$ ▷ element-wise max
  $\zeta_d^i \leftarrow \max(\zeta_t, \zeta_s, \zeta_m)$
  if $\sum_{d' \in D} \mathbb{1}_{\{\mathbf{T}_{d,d'} = -1\}} \geq 1$ **then**
    $\mathbf{r}_d^i, \zeta_d^i \leftarrow$ ApplyNegativeTest($\zeta_d^i, \mathbf{r}_d^i, \mathbf{T}_d^i, W$)
  end if
  return $\mathbf{r}_d^i$, ζ_d^i
end function

C.4 Other simulation metrics

In this section we present same plots as in the main section, but for the individual metrics, namely, cumulative incidence, $\hat{H}$, effective contacts, $\hat{E}$, and reproduction number R.

C.4.1 Cumulative Incidence $\hat{H}$

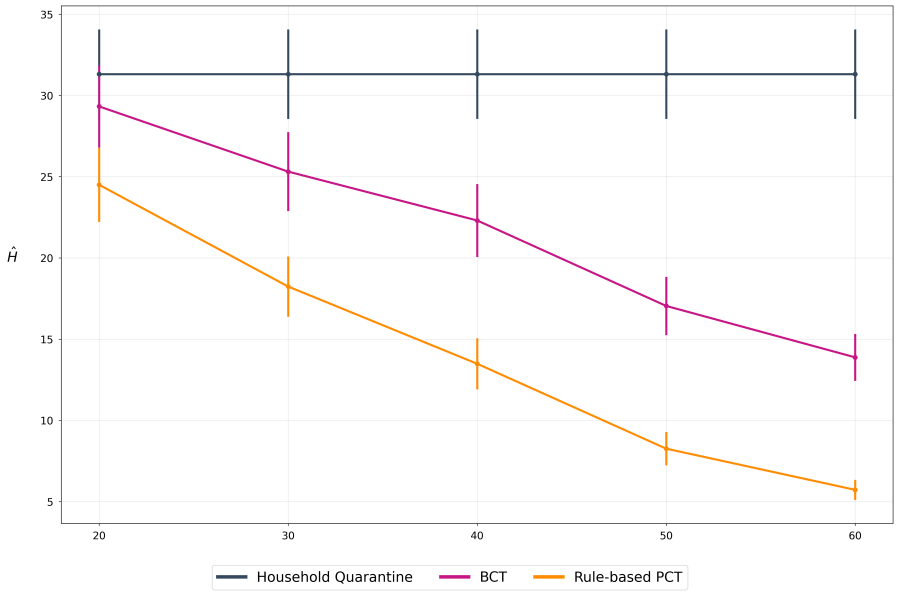

Figure C.9: Adoption rate comparison. We compare all methods for adoption rates between 0% (HQ) and 60% of both BCT and PCT.

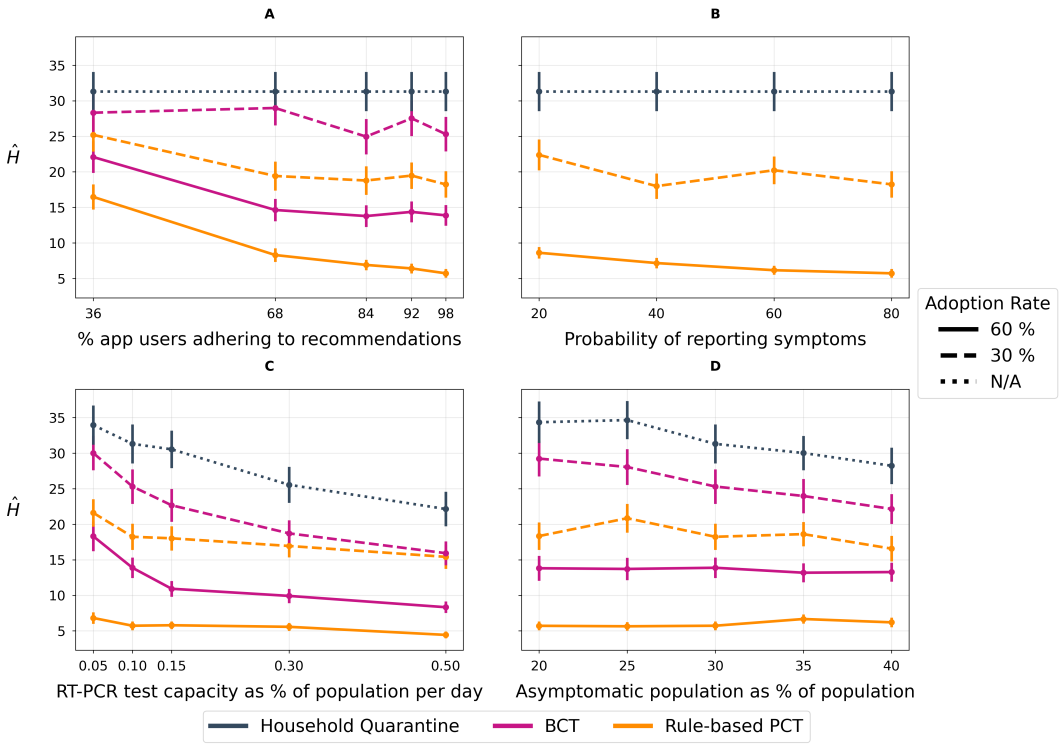

Figure C.10: Sensitivity Analyses. We use N/A to represent irrelevance of adoption rate in the baseline scenario as no DCT app is deployed. **(A) Recommendation Adherence.** Illustrates the impact of varying recommendation adherence (e.g. the daily likelihood of getting a test, quarantining, reducing contacts given an in-app notification is received). **(B) Symptom reporting.** Illustrates the impact of varying the daily rate of symptom reporting. Note: the plot omits BCT because BCT doesn't incorporate symptoms in its inputs. **(C) RT-PCR Testing Capacity.** Illustrates the impact of varying the percentage of the population that can receive an RT-PCR test on any given day, ranging from the observed provincial testing capacity of 0.1% to a highly optimistic value of 0.5% of the population. **(D) Infectiousness and symptoms.** Illustrates the impact of varying the proportion of cases that will not develop symptoms.

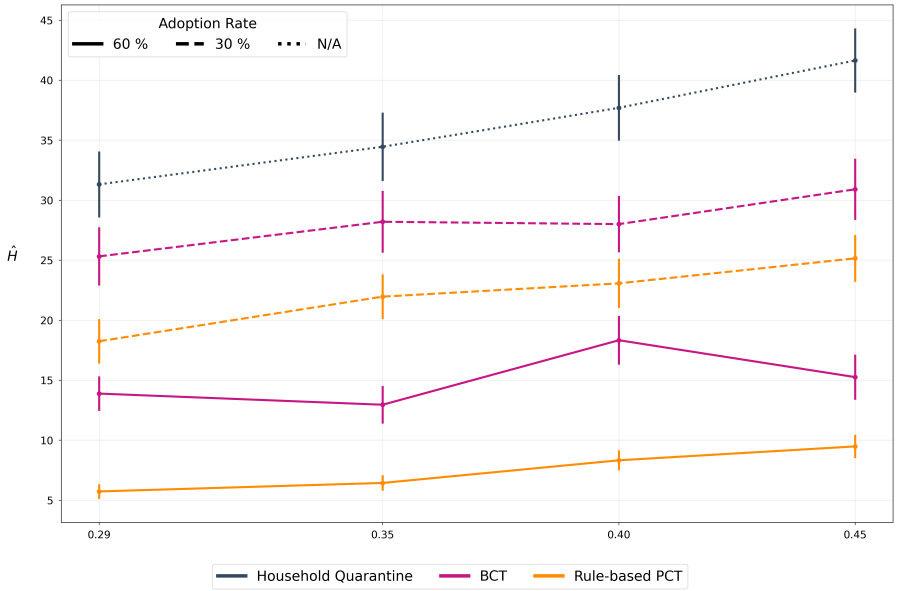

Figure C.11: Asymptomatic infection ratio. We vary the relative infectiousness of asymptomatic cases. A value of f implies that the asymptomatic case can potentially infect f times as many people as compared to a symptomatic case. A value of 0.29 is the chosen minimum as described in the epidemiological literature while a higher value of 0.45 is a hypothetical situation describing a more infectious variant of the virus.

C.4.2 Effective Contacts $\hat{E}$

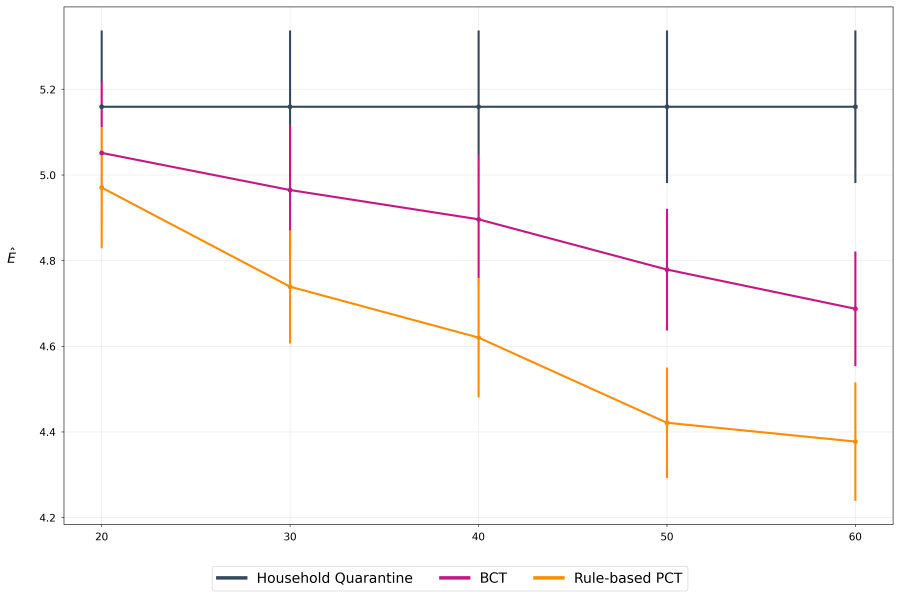

Figure C.12: Adoption rate comparison. We compare all methods for adoption rates between 0% (HQ) and 60% of both BCT and PCT.

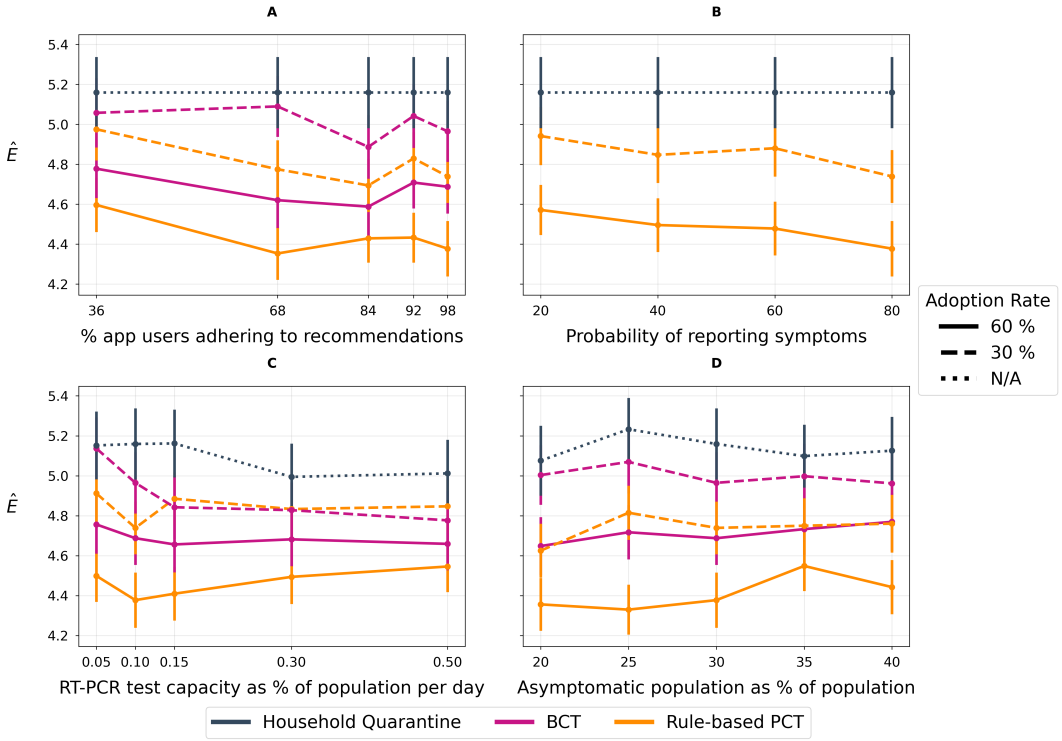

Figure C.13: Sensitivity Analyses. We use N/A to represent irrelevance of adoption rate in the baseline scenario as no DCT app is deployed. **(A) Recommendation Adherence.** Illustrates the impact of varying recommendation adherence (e.g. the daily likelihood of getting a test, quarantining, reducing contacts given an in-app notification is received). **(B) Symptom reporting.** Illustrates the impact of varying the daily rate of symptom reporting. Note: the plot omits BCT because BCT doesn't incorporate symptoms in its inputs. **(C) RT-PCR Testing Capacity.** Illustrates the impact of varying the percentage of the population that can receive an RT-PCR test on any given day, ranging from the observed provincial testing capacity of 0.1% to a highly optimistic value of 0.5% of the population. **(D) Infectiousness and symptoms.** Illustrates the impact of varying the proportion of cases that will not develop symptoms.

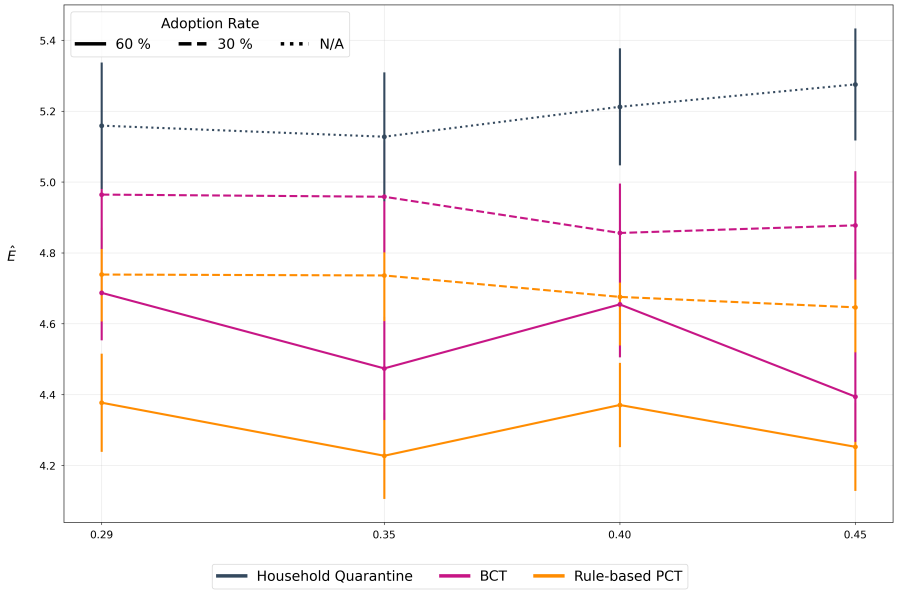

Figure C.14: Asymptomatic infection ratio. We vary the relative infectiousness of asymptomatic cases. A value of f implies that the asymptomatic case can potentially infect f times as many people as compared to a symptomatic case. A value of 0.29 is the chosen minimum as described in the epidemiological literature while a higher value of 0.45 is a hypothetical situation describing a more infectious variant of the virus.

C.4.3 Reproduction Number R

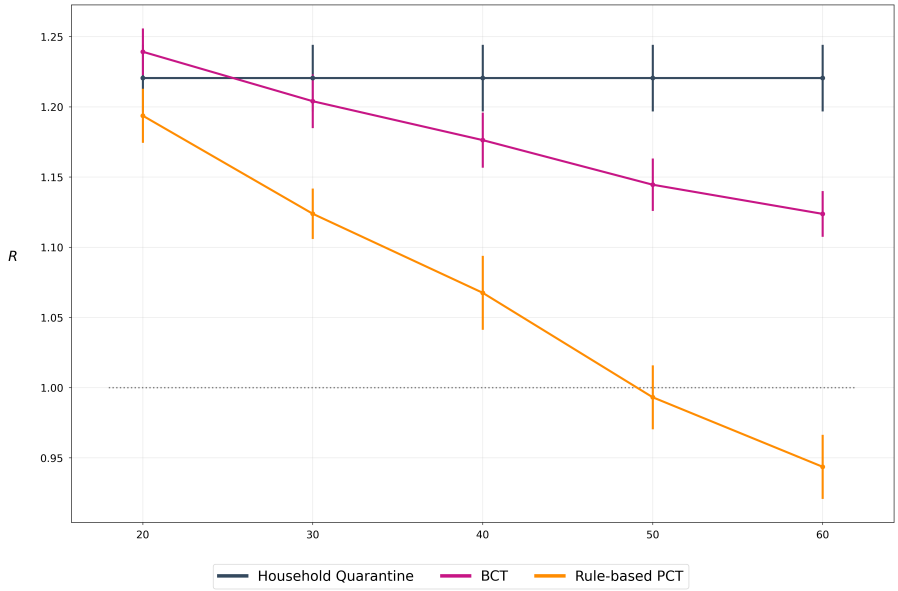

Figure C.15: Adoption rate comparison. We compare all methods for adoption rates between 0% (HQ) and 60% of both BCT and PCT.

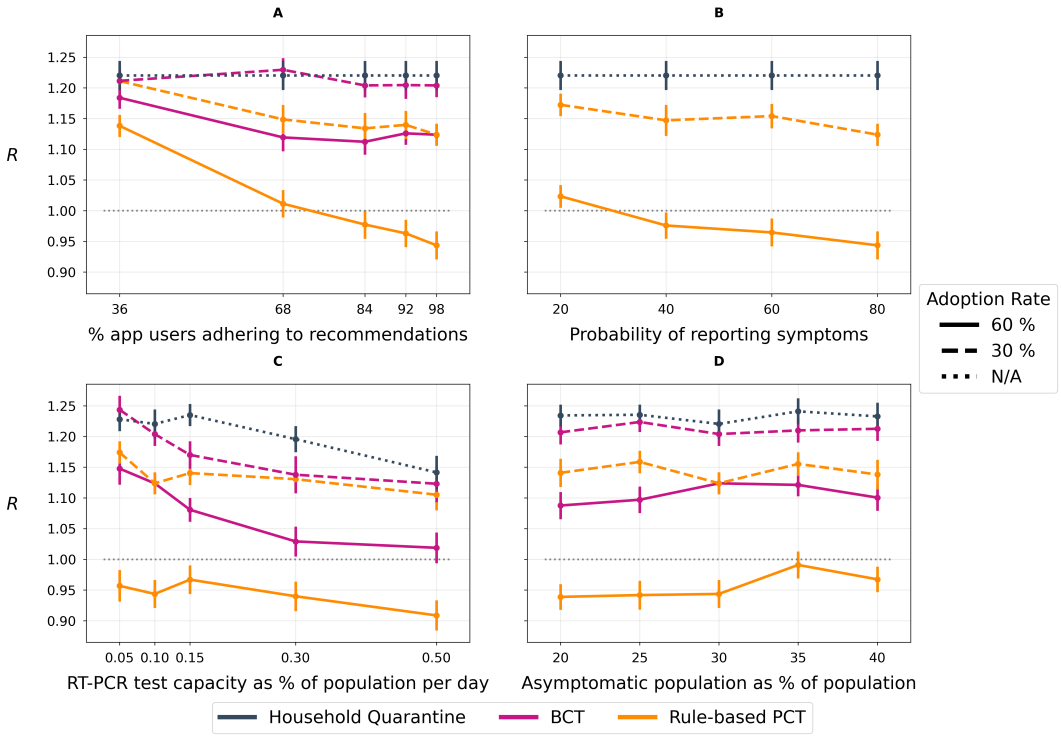

Figure C.16: Sensitivity Analyses. We use N/A to represent irrelevance of adoption rate in the baseline scenario as no DCT app is deployed. **(A) Recommendation Adherence.** Illustrates the impact of varying recommendation adherence (e.g. the daily likelihood of getting a test, quarantining, reducing contacts given an in-app notification is received). **(B) Symptom reporting.** Illustrates the impact of varying the daily rate of symptom reporting. Note: the plot omits BCT because BCT doesn't incorporate symptoms in its inputs. **(C) RT-PCR Testing Capacity.** Illustrates the impact of varying the percentage of the population that can receive an RT-PCR test on any given day, ranging from the observed provincial testing capacity of 0.1% to a highly optimistic value of 0.5% of the population. **(D) Infectiousness and symptoms.** Illustrates the impact of varying the proportion of cases that will not develop symptoms.

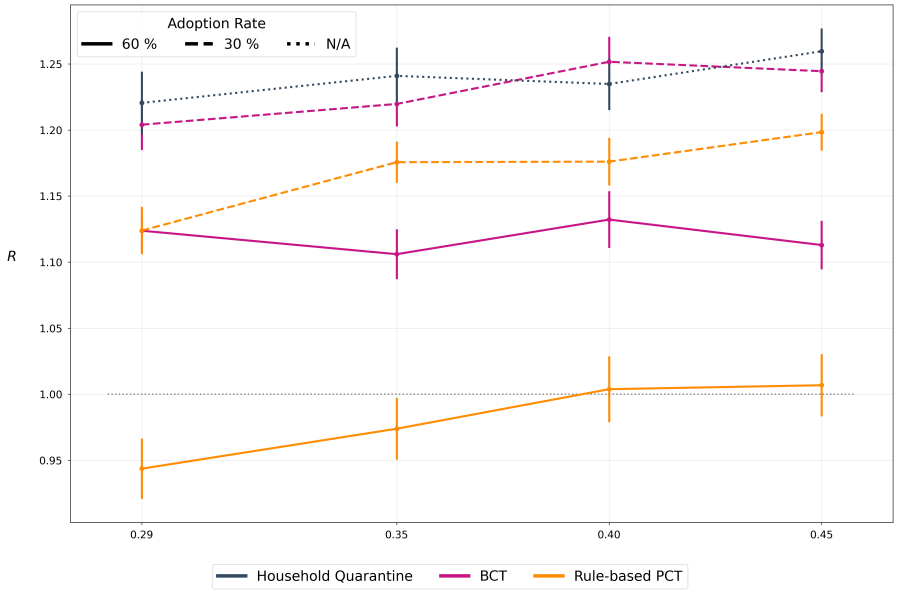

Figure C.17: Asymptomatic infection ratio. We vary the relative infectiousness of asymptomatic cases. A value of f implies that the asymptomatic case can potentially infect f times as many people as compared to a symptomatic case. A value of 0.29 is the chosen minimum as described in the epidemiological literature while a higher value of 0.45 is a hypothetical situation describing a more infectious variant of the virus.

C.5 Cost-effectiveness Analysis

DALYs Disability-Adjusted Life Years (DALYs) are a summary measure of the public health burden associated with premature mortality and morbidity due to having a specific disease or health condition. Notably, DALYs are not a direct metric of public health, but rather of the public health *burden*: a decrease in DALYs results in an increase in overall public health. To calculate DALYs, we individually compute the years of life lost due to premature mortality (YLLs) (Roth et al., 2018) for agents that died during the simulation (i.e. early death), as well as the years of life lost due to disability (YLDs) (James and al, 2018) for agents that were infected and symptomatic. Disability weights (DW) are retrieved from the 2017 Global Burden of Disease Study (Global Burden of Disease Collaborative Network, 2018) and represent health preferences such that: DW of 0 is perfect health, and DW of 1 is equivalent to death. DALYs were calculated without discounting or age-weighting, following the methodology set out by the WHO (World Health Organization, 2018). Life expectancies are retrieved from Statistics Canada[12].

Symptomatic agent status	Disability weight	Equivalent to
Not hospitalized	0.051	Moderate lower respiratory infection
Hospitalized	0.133	Severe lower respiratory infection
Critical care	0.408	Severe COPD without heart failure

Table C.3: Disability weights were drawn from the 2017 Global Burden of Disease Study (Global Burden of Disease Collaborative Network, 2018). Since there are no published weights for COVID-19, weights for similar health conditions are selected. As can be seen, higher weights are associated with worse health states. COVID-19 is a novel disease, and there are no published disability weights for different levels of severity of the disease. Therefore, we use similar conditions as proxies for different health states of the agents. Agent hospitalization status is used as a proxy for actual health status, and can be divided into three categories: agents that are symptomatic and not hospitalized, agents that are hospitalized but not in critical care, and agents that are in critical care.

Thus, defining L_i as life expectancy for an agent i,

$$YLL_i = \mathbb{1}_{\text{i died}} \times L_i$$

[12]Table 13-10-0114-01 Life expectancy and other elements of the life table, Canada, all provinces except Prince Edward Island. Statistics Canada. Available from: `https://www150.statcan.gc.ca/t1/tbl1/en/tv.action?pid=1310011401`.

$$YLD_i = \sum_{s}^{S} DW_s \times l_{i,s}$$
$$DALY_i = YLL_i + YLD_i$$

where:

$$L_i := \text{Life expectancy of agent } i$$
$$DW_s := \text{Disability weight of state } s$$
$$l_{i,s} := \text{duration of state } s \text{ for agent } i$$
$$\mathbb{1}_{\text{i died}} := \begin{cases} 1 & \text{if agent } i \text{ died during the simulation} \\ 0 & \text{otherwise} \end{cases}$$

TPL Temporary Productivity Loss (TPL) is the loss in productivity due to absenteeism from work. To calculate TPL, we extract from the simulator the number of work hours that agents aged 25 to 65 years had to forego due to: (1) quarantine; (2) taking care of a dependent family member (supervision) or; (3) illness that prohibits someone from working. We then multiply this quantity of foregone work hours by the 2019 average hourly wage in Montréal[13] to obtain the total TPL. In computing TPL, we assume that total foregone work hours due to quarantine are scaled by a factor of 0.49(Gallacher and Hossain, 2020) to account for the proportion of agents able to work from home. We follow the methodology for calculating TPL presented in Nurchis et al. (2020).

Aggregated across all agents,

$$n_{25:65} = n_{25:65}^{quarantine} + n_{25:65}^{supervision} + n_{25:65}^{illness}$$
$$TPL = n_{25:65} \times w_{average},$$

where $n_{25:65}^{a}$ is the number of hours foregone due to a, and $w_{average}$ is the average hourly wage for the Montreal workforce.

[13]Weekly and hourly earnings of employees by sex, Montréal and all of Québec, 2015-2019. Institut de la Statistique du Québec. Available from: `https://www.stat.gouv.qc.ca/statistiques/profils/profil06/societe/marche_trav/indicat/tra_remuneration06_an.htm`.

This approach to evaluating the cost of a healthcare intervention is referred to as the Human Capital Approach (HCA) (Nurchis et al., 2020). The HCA can be used to evaluate the TPL, as well as another form of productivity loss known as the permanent productivity loss due to premature death (PPL)(Nurchis et al., 2020). Given the limited number of agents in our simulator, we avoid calculating the PPL due to the high uncertainty surrounding deaths from COVID-19 within the age range of the workforce (25 to 65). Calculating only the TPL instead of both the TPL and the PPL is a more conservative approach: it likely underestimates the impact of our intervention.

ICER

Consider a counterfactual simulation with no pandemic: no DALYs are incurred due to COVID-19 and there is no associated TPL. We evaluate different scenarios (HQ, BCT, PCT) by calculating the differences in DALYs and TPL compared to this counterfactual. We then calculate the ICER between each set of two scenarios according to the following formula:

$$ICER = \frac{C_1 - C_0}{E_1 - E_0}$$

where C_i and E_i respectively represent the economic cost (TPL) and health outcomes (avoided DALYs) of intervention i (HQ, BCT or PCT). C_1 and E_1 represent the outcomes of a new intervention that is being considered for adoption, whereas C_0 and E_0 are associated to a *reference intervention*, which is typically a baseline intervention or standard of care (Drummond et al., 2015).

Let us consider how to calculate $E_1 - E_0$: given that we evaluate incurred DALYs compared to a no-pandemic counterfactual with no DALYs lost due to COVID-19, the improvement in health outcomes of adopting a new intervention instead of the reference intervention is the *reduction* in DALYs of the new intervention over the reference intervention:

$$E_1 - E_0 = -(DALYs_1 - DALYs_0) = DALYs_0 - DALYs_1$$

Hence, the ICER can be calculated with the following formula:

$$ICER = \frac{TPL_1 - TPL_0}{DALYs_0 - DALYs_1}$$

for a reference intervention indexed by 0 and an alternative intervention indexed by 1.

To calculate the impact of both DCT methods (BCT, PCT) over HQ, we calculate the ICER of each method at various adoption rates with respect to HQ:

$$\frac{TPL_{BCT} - TPL_{\text{HQ}}}{DALYs_{\text{HQ}} - DALYs_{BCT}} \qquad \frac{TPL_{PCT} - TPL_{\text{HQ}}}{DALYs_{\text{HQ}} - DALYs_{PCT}}$$

To further compare between BCT and PCT, we calculate the ICER of PCT with BCT as a reference intervention:

$$\frac{TPL_{PCT} - TPL_{BCT}}{DALYs_{BCT} - DALYs_{PCT}}$$

If the ICER of the new intervention is negative with respect to the reference intervention, then the new intervention is described as “cost-saving" under the assumption that the new intervention results in better health outcomes than the reference intervention ($E_1 > E_0$).

Willingness-to-pay threshold The common willingness-to-pay threshold in Canada of \$20k per DALY threshold is from 1992 (Laupacis et al., 1992), so we have to adjust it for inflation to reflect the real threshold in 2020. To make this adjustment, we consult the Canadian Consumer Price Index[14] and adjust the \$20k threshold to reflect inflation since 1992:

$$WTP_{2020} = \frac{CPI_{2020}}{CPI_{1992}} \cdot WTP_{1992} = \frac{137}{84} \cdot 20000 = \$32619.048 \approx \$33\text{K}.$$

[14]Table 18-10-0005-01 Consumer Price Index, annual average, not seasonally adjusted, 2022. Consumer Price Index, Statistics Canada. Available from: `https://www150.statcan.gc.ca/t1/tbl1/en/tv.action?pid=1810001101`

D

Predicting Infectiousness for Proactive Contact Tracing

Contents

D.1 Glossary of bolded terms

Contact tracing Finding the people who have been in contact with an infected person, typically using information such as test results and phone surveys, and recommending that they quarantine themselves to prevent further spread of the disease.

Manual contact tracing Method for contact tracing using trained professionals who interview diagnosed individuals to identify people that have come into contact with an infected person and recommend a change in their behavior (or further testing).

Digital contact tracing (DCT) Predominantly smartphone-based methods of identifying individuals at high risk of contracting an infectious disease based on their interactions, mobility patterns, and available medical information such as test results.

Binary contact tracing (BCT) Methods which use binary information (e.g. tested positive or negative) to perform contact tracing, thus putting users in binary risk categories (at-risk or not-at-risk).

Proactive contact tracing (PCT) DCT methods which produce graded risk information to mitigate spread of disease, thus putting users in graded risk categories (more-at-risk or less-at-risk), potentially using non-binary clues like reported symptoms which are available earlier than test results.

Agent-based epidemiological models (ABMs) Also called individual-based, this type of simulation defines rules of behaviour for individual agents, and by stepping forward in time, contact patterns between agents are generated in a "bottom up" fashion. Contrast with **Population-level epidemiological models**.

Risk of infection Expected infectiousness, or probability of infection for a susceptible agent by an infectious agent given a qualifying contact (15+ minutes at under 2 meters).

Contact An encounter between 2 agents which lasts at least 15 minutes with a distance under 2 meters[1].

Risk Level A version of the risk of infection quantized into 16 bins (for reducing the number of bits being exchanged, for better privacy protection). The thresholds are selected by running the domain randomization with separate seeds and grouping risk messages such that there are an approximately uniform number in each bin.

Risk mapping A table which maps floating point risk scalars into one of 16 discrete risk levels.

Recommendation mapping A table of which recommendation level should be associated with a given risk level. Each recommendation level comes with a series of specific recommendations (e.g. "Limit contact with others", "wash hands frequently", "wear a mask when near others", "avoid public transportation", etc) which should be shown for a given risk level. Reminders of these recommendations may be sent via push notification more or less often depending on the recommendation level.

Population-level epidemiological models This type of model fits global statistics of a population with a mathematical model, typically sets of ordinary differential equations. The equations track some statistics over time, typically counts of people in each of a few "compartments", e.g. the number of Infected and Recovered, and the parameters of the equations are often tuned to match statistics collected from real data. It does not give any information about the individual-level contact patterns which give rise to these statistics.

[1]Public Health Management of cases and contacts associated with COVID-19, 2020. Government of Canada. Available from: `https://www.canada.ca/en/public-health/services/diseases/2019-novel-coronavirus-infection/health-professionals/interim-guidance-cases-contacts.html`. Accessed: 2020-06-04.

Compartment model (SEIR) A compartment model tracks the counts of agents in each of several mutually-exclusive categories called "compartments". In an SEIR model, the 4 compartments are **Susceptible, Exposed, Infectious, and Recovered** (see entries for each of these words).

Susceptible At risk of catching disease, but not infected.

Exposed or infected Infected with the disease, (i.e. potentially carries some **viral load**).

Infectious Carries sufficient viral load to transmit the disease to others. In real life, this typically means that the virus has multiplied sufficiently to overwhelm the immune system.

Recovered No longer carries measurable viral load, after having been infected. In real life this is typically measured by two successive negative **lab tests**.

Viral load Viral load is the number of actual viral RNA in a person as measured by a **lab test**.

Effective viral load A term we introduce representing a number between 0 and 1 which we use as a proxy for viral load. It could be converted to an actual viral load via multiplying by the maximum amount of viral RNA detectable by a lab test.

Infectiousness Degree to which an agent is able to transmit viruses to another agent. A factor in determining whether an encounter between a susceptible and an infectious agent results in the susceptible agent being exposed (it is a property of the infector agent; this does not consider environmental factors or the other agent's susceptibility; see **Transmission probability**).

Attack rate The proportion of individuals in a population who are infected over some time period.

Contamination duration How long a location remains infectious after an infected person has visited it and shed viruses there.

Transmission probability The chance that one agent infects another during an encounter. **Infectiousness** of the infected agent, **susceptibility** of the other agent, both their behavior (e.g. usage of masks) and other environmental factors all play a role in determining this probability.

PCR Test Also, colloquially referred to as a lab test. In this context, this means a Polymerase Chain Reaction (PCR) test used to amplify viral DNA samples, typically obtained via a nasopharangeal swab.

Asymptomatic When an infected person shows no symptoms. For COVID-19, many people remain asymptomatic for the entire course of their disease.

Incubation days The number of days from exposure (infection) until an agent shows symptoms. In real life this is the average amount of time it takes the viral RNA to multiply sufficiently in / burst out of host cells and cause an immune response, which is what produces the symptoms.

Reproduction number (R) The average number of other agents an agent infects, measured over a certain window of time. We follow Gupta et al. (2020b) and approximate R by computing the infection tree and taking the ratio of the number of infected children divided by the number of parents who are recovered infectors.

Serial interval The average number of days between successive symptom onsets (i.e. between the **incubation** time of the infector and the infectee).

Generation time The number of days between successive infections (i.e. between the exposure of the infector and the infectee). Generation time is difficult to measure directly so one can estimate it from the **serial interval**.

Domain randomization A method of generating a training dataset by sampling from several distributions, each corresponding to a different setting of parameters of a simulator. This allows to generate a dataset which covers a potentially wide range of settings, improving generalization to environments which may not be well covered by the simulator in any one particular setting chosen a priori.

Oracle Predictor Baseline for the **Set Transformer** which uses ground-truth infectiousness levels as "predictions". Provides an expected upper-bound performance for the predictors and can be used to provide risk level inputs when pre-training a predictors.

Multi-Layer Perceptron (MLP) A risk prediction model which concatenates all inputs (after heuristically aggregating to a fixed size vector the set-valued inputs), feeds these through several fully-connected neural network layers and is trained to predict infectiousness. See Section D.3.1.

Set Transformer (ST) Model A risk prediction model which uses a modified set transformer to process and attend to inputs, and is trained to predict infectiousness. See Section D.3.1.

Deep Set (DS) Model Similar structure as the Set Transformer, but using pooling instead of self-attention. This allows it to use less memory and compute as compared to the Set Transformer.

Adoption rate The proportion of the total population which uses a digital contact tracing application.

Domain randomization The technique of varying the parameters of a simulation environment to create a broad distribution of training data.

Re-identification The process of matching anonymous data with publicly available data so as to determine which person is the owner.

Big brother attacks "Big brother" is a reference to the 1984 book by George Orwell where a highly regulated government surveilled and controlled its citizenry. The term "big brother attacks" references the idea that an organized group (e.g., state, federal, academic or financial institution) may try to gain control of your data.

Little brother attacks Little brother attacks references the idea that potential security threats are also posed by smaller actors like individuals, criminals, and data brokers.

Vigilante attacks Extra-judicial violence by an individual. In the context of this paper, specifically as a result of a recommendation shown to the user or gained through other means.

Degree of Restriction Ratio between the number of people given a recommendation in the highest level of restriction (i.e. quarantine) relative to other categories.

Global mobility scaling factor A simulator parameter enabling us to vary the amount of contacts which may lead to infection in the simulator. It allows us to scale the mobility to simulate pre-lockdown or post-lockdown environments.

Auto-induced distribution shift A change in distribution of data observed by an agent or algorithm as a result of the agent or algorithm's actions.

Iterative re-training The process of generating data using some method in a simulator, training a model on this data in a supervised manner, then evaluating the model in the simulator to produce more data.

Pareto frontier The Pareto frontier is a way of evaluating a tradeoffs in a set of policies and environments with multi-dimensional outputs (e.g., viral spread and mobility).

D.2 Comparison of approach to related work

Epidemiological Modeling and Simulations. Given that modeling contact-tracing requires capturing past interactions, it is mathematically complicated to consider the dynamic nature of network interactions. An agent-based model (simulator), on the other hand, gives us maximum flexibility to incorporate real data and/or assumptions easily and emulate the effect of personalized policies (such as resulting from the proposed app). Models with binary contagion and random-walk mobility are ubiquitous (Stevens, 2020). The appeal of such models is their simplicity; they are easy to code, fast to run, and can give a general picture of some aspects of disease spread. There are other models with increasing complexity of either the mobility model, contagion model, agent demographics, or some combination of these, e.g. (Verity et al., 2020; Wood et al., 2022; Lorch et al., 2022; Hinch et al., 2020).

GLEAM (Global Epidemic And Mobility model)[2] is an off-the-shelf simulation platform for epidemics which offers mobility patterns and demographic information, and uses a generic format for defining how a disease depends on these two things. Similarly, FRED (Framework for Reconstructing Epidemiological Dynamics) provides an open-source, agent-based model with realistic social networks and US demographics. Grefenstette et al. (2013), and (Wood et al., 2022) build an intervention-planning tool on top of this simulator. These works are comparable to our simulator only; they do not do any contact tracing or individual risk prediction.

The work presented in (Lorch et al., 2022) and (Ferretti et al., 2020) is closely related to ours. A detailed mobility model for interactions is presented in (Lorch et al., 2022), but the epidemiological model of the disease is much simpler there. First, the contact graph is built based on their mobility model. The next step is an implementation of various policy interventions by health authorities, which includes contact tracing. This is done instead of building a contact graph that takes into account policy interventions and contact tracing apps at an individual level. Next, most similarly to our work, (Ferretti et al., 2020) proposes an app-based tracing and recommendations. However, their epidemiological model is

[2]GLEAM Project. Available from: `https://www.gleamproject.org/`.

a very simple differential-equation model, and interventions for controlling the disease are computed analytically.

Risk estimation approaches. While the applications deployed so far are based primarily on binary contact tracing as discussed above, some probabilistic risk estimation approaches, similar to ours, have been developed for other diseases or applications, and have begun to be applied to COVID-19. For example, (Baker et al., 2021) uses the susceptible-infected-recovered (SIR) model and describes the dynamical process of infection propagation using the dynamical message passing equations from (Lokhov et al., 2014); the probability of each node (person) to be in a specific state (S,I or R) is estimated via the dynamic message-passing (DMP) algorithm, which belongs to the family of local message-passing methods similar to belief propagation (BP) algorithm (Yedidia et al., 2000) for estimating marginal probability distributions over the network nodes; despite BP being only an approximate inference method, not guaranteed to converge to the correct marginals when the underlying graphical model has (undirected) cycles, it demonstrated remarkable performance in various applications. A similar approach based on BP was also discussed in (Murphy et al., 2021). Furthermore, there are recent extensions of belief propagation approach graph neural nets (Satorras and Welling, 2021). Also, another recent work uses Gibbs sampling and SEIR model for their test-trace-isolate approach (Herbrich et al., 2020). However, such approaches typically rely on the knowledge of the social interaction graph, which is not available in our case due to privacy and security constraints.

To the best of our knowledge, our work is the first to use an approach based on detailed agent-based epidemiological model together with a model of phone app messaging to generate simulated data for training an ML-based predictor of individual-level risk.

Digital contact tracing for COVID-19 (Hinch et al., 2020; Hellewell et al., 2020; Aleta et al., 2020; Grantz et al., 2021) study, either via simulations or mathematical models, the conditions under which BCT can be effective. Toward addressing

the issues with BCT, (Hinch et al., 2020) show in simulation that using self-reported symptoms in addition to test results can greatly help control an outbreak. Probabilistic (non-binary) approaches to the problem of contact tracing (e.g. Baker et al. (2021); Satorras and Welling (2021); Briers et al. (2020)) typically assume full access to location histories and contact graph, an unacceptable violation of privacy in most places in the world. As a result, these methods are most often used for predicting overall patterns of disease spread.

Agent-Based Models as a generative process Our use of an agent-based simulator (Gupta et al., 2020b) as a generative model allows us to generate fine-grained (continuous) values for expected infectiousness with realistic contact patterns. Most works performing probabilistic inference for disease modeling use a simple differential equation generative model, which does not characterize individual behavior but rather the dynamics of transition between each of several mutually exclusive disease states. Such models make many simplifying assumptions, such as contact patterns based on random walks, which make them unsuitable for individual-level prediction of infectiousness; they are typically used instead to infer latent variables such as infectiousness that would be consistent with the population-level statistics generated by the differential equation model, and/or to predict statistics of spread of the disease in a population, e.g. (ref, ref). While agent-based models are widely used in epidemiological literature to model the spread of disease (see e.g. (ref) for review), to our knowledge we are the first to use an ABM as a generative model for training a deep learning-based infectiousness predictor.

Distributed inference and belief propagation Belief propagation in graphical models is often used for disease spread modeling, e.g. Fan et al. (2016). Some recent works have applied this to COVID-19; for example, Baker et al. (2021) use the susceptible-infected-recovered (SIR) model and describe the process of infection propagation using the dynamical message passing equations from Lokhov et al. (2014). A work concurrent with ours follows a similar justification for modeling a latent parameter of expected infectiousness, using an SEIR model with inference via (Herbrich et al., 2020). However, these approaches rely on a centralized social

graph or a large number of bits exchanged between nodes, which is challenging both in terms of privacy and bandwidth. This challenge motivated our particular form of distributed inference where we pretrain the predictor and do not assume that the messages exchanged are probability distributions, but instead just informative input to the node-level predictor.

D.3 Experimental details

D.3.1 ML architectures and baseline details:

Binary contact tracing quarantines app-users who had high-risk encounters with an app-user who receives a positive PCR test. Under BCT, if Alice gets a positive test result, then every user who encountered Alice within 14 days of her receiving the positive test result is sent a message which places them in app-recommendation level 3 (quarantine) for 14 days.

Set Transformer Recall that in Section 5.3 we proposed two parameterizations of the model (DS and ST in Figure 5.2). In the first proposal, we use a set transformer (ST) to model interactions between the elements in the set $\mathbb{D}_i^d \cup \mathbb{E}_i^d$. We now describe the precise architecture of the model used.

The model comprises 5 embedding modules, namely: the health status embedding ϕ_{hs}, health profile embedding ϕ_{hp}, day offset embedding ϕ_{do}, risk message embedding $\phi_e^{(r)}$ and an embedding $\phi_e^{(n)}$ of the number of repeated encounters.

The model was trained for 160 epochs on a domain randomized dataset (see below) comprising $\sim 10^7$ samples. We used a batch-size of 1024, resulting in $\sim 80k$ training steps. The learning rate schedule is such that the first $2.5k$ steps are used for linear learning-rate warmup, wherein the learning rate is linearly increased from 0 to 2×10^{-4}, followed by a cosine annealing schedule that decays the learning rate from 2×10^{-4} to 8×10^{-6} in $50k$ steps.

D.3.2 Domain randomization

Inspired by research in hyper-parameter search (Bergstra and Bengio, 2012) and recent advances in deep reinforcement learning (Tobin et al., 2017) we created the transformer's training data by sampling uniformly in the following ranges:

1. Adoption rate $\in [30 - 60]$
2. Carefulness $\in [0.5 - 0.8]$
3. Initial proportion of exposed people $\in [0.002, 0.006]$
4. Oracle additive noise $\in [0.05 - 0.15]$
5. Oracle multiplicative noise $\in [0.2 - 0.8]$
6. Global mobility scaling factor $\in [0.3 - 0.9]$
7. Symptoms dropout: Likelihood of not reporting some symptoms $\in [0.1, 0.6]$
8. Symptoms drop-in: Likelihood of falsely reporting symptoms $\in [0.0001, 0.001]$
9. Quarantine dropout (test) $\in [0.01, 0.03]$: likelihood of not quarantining when recommended to quarantine due to a positive test
10. Quarantine dropout (household) $\in [0.02, 0.05]$: likelihood of not quarantining when recommended to quarantine because a household member got a positive test
11. All-levels dropout $\in [0.01, 0.05]$: likelihood of not following app-recommended behavior and instead exhibit pre-pandemic behavior

D.3.3 Training time

Our ML experiments use approximately 250 days training time on GPUs while simulations required approximately 41 days of CPU time. All CPU time was run on compute using renewable resources.

E

Anticipating Worst-Case Viruses

The PCT framework with warning signals strives to caution individuals earlier. The usefulness of early warning signals will be more noticeable for a virus that exhibits dormant characteristics, i.e., the host is infectious without definite signs of being infected.

The following changes to the *viral profile* will increase its dormant characteristics,

- **Weaker tests & delayed results**: In the absence of conclusive diagnostic tests, we will be likely to detect an infectious person. SARS-CoV-2's most conclusive test is through Real-Time Polymerase Chain Reaction Test (RT-PCR). The test's false-negative rate varies depending on the individual's viral load, with a minimum of 33% at the peak viral load (Li et al., 2020). The test results take at least a day to two to return. Thus, a worst-case virus will likely have a higher false-negative rate with a longer turnover time. As an aside, the mega-diagnostic platforms[1], proposed in Bill and Melinda Gates, 2021 Annual letter focused on better preparedness to future pandemic, serve the purpose of faster turnover.

- **Inconclusive symptoms & harsher health-impact**: If an infected individual shares a symptom profile with other common diseases, it is likely to be ignored by an individual. Familiar stories have come up during the COVID-19 pandemic. Thus, a worst-case virus will likely have more of such inconclusive signs for some people and have a harsher health impact on others.

- **Longer incubation period**: If an infectious individual is unaware of the infection, they are likely to spread the infection. Recall that the incubation period is the time since infection before any symptoms appear. SARS-CoV-2 has a mean incubation period of five days(Lauer et al., 2020), so a longer incubation period will likely to result in more infections per individual (i.e. higher reproductive number).

[1]Available from: `https://www.gatesnotes.com/2021-Annual-Letter#ALChapter3`

- **More asymptomatic population & their infectiousness**: An estimated 20-40% (He et al., 2020; Luo et al., 2020) of the population does not show any symptoms throughout the time they are infected. However, their infectiousness (likelihood to infect others) is also roughly 30% of those who exhibit symptoms (Luo et al., 2020). Thus, a worst-case virus will likely have a larger proportion (less than 100%) of asymptomatic population with higher infectiousness.

Bibliography

Abeler, J., Bäcker, M., Buermeyer, U., and Zillessen, H. (2020). COVID-19 contact tracing and data protection can go together. *JMIR mHealth uHealth*, 8(4):e19359.

Abrache, J., Crainic, T. G., Gendreau, M., and Rekik, M. (2007). Combinatorial auctions. *Annals of Operations Research*, 153(1):131–164.

Abueg, M., Hinch, R., Wu, N., Liu, L., Probert, W. J. M., Wu, A., Eastham, P., Shafi, Y., Rosencrantz, M., Dikovsky, M., Cheng, Z., Nurtay, A., Abeler-Dörner, L., Bonsall, D. G., McConnell, M. V., O'Banion, S., and Fraser, C. (2020). Modeling the combined effect of digital exposure notification and non-pharmaceutical interventions on the covid-19 epidemic in washington state. *medRxiv*.

Achterberg, T. (2007). *Constraint Integer Programming*. Doctoral thesis, Technische Universität Berlin, Fakultät II - Mathematik und Naturwissenschaften, Berlin.

Achterberg, T. and Berthold, T. (2009). Hybrid branching. In *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*.

Achterberg, T., Koch, T., and Martin, A. (2005). Branching rules revisited. *Operations Research Letters*, 33(1):42 – 54.

Aleta, A., Martín-Corral, D., y Piontti, A. P., Ajelli, M., Litvinova, M., Chinazzi, M., Dean, N. E., Halloran, M. E., Longini Jr, I. M., Merler, S., et al. (2020). Modelling the impact of testing, contact tracing and household quarantine on second waves of covid-19. *Nature Human Behaviour*, 4(9):964–971.

Alsdurf, H., Belliveau, E., Bengio, Y., Deleu, T., Gupta, P., Ippolito, D., Janda, R., Jarvie, M., Kolody, T., Krastev, S., Maharaj, T., Obryk, R., Pilat, D., Pisano, V., Prud'homme, B., Qu, M., Rahaman, N., Rish, I., Rousseau, J.-F., Sharma, A., Struck, B., Tang, J., Weiss, M., and Yu, Y. W. (2020). Covi white paper.

Alvarez, A. M., Louveaux, Q., and Wehenkel, L. (2017). A machine learning-based approximation of strong branching. *Institute for Operations Research and the Management Sciences Journal on Computing*, 29(1):185–195.

Andrychowicz, M., Baker, B., Chociej, M., Józefowicz, R., McGrew, B., Pachocki, J., Petron, A., Plappert, M., Powell, G., Ray, A., Schneider, J., Sidor, S., Tobin, J., Welinder, P., Weng, L., and Zaremba, W. (2020). Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 39(1):3–20.

Applegate, D., Bixby, R., Chvátal, V., and Cook, W. (1995). Finding cuts in the TSP. Technical report, DIMACS.

Arora, P., Vasa, P., Brenner, D., Iglar, K., McFarlane, P., Morrison, H., and Badawi, A. (2013). Prevalence estimates of chronic kidney disease in canada: results of a nationally representative survey. *Canadian Medical Association Journal*, 185(9):E417–E423.

Arregui, S., Aleta, A., Sanz, J., and Moreno, Y. (2018). Projecting social contact matrices to different demographic structures. *PLoS computational biology*, 14(12):e1006638.

Ayres, I., Romano, A., and Sotis, C. (2020). How to make covid-19 contact tracing apps work: Insights from behavioral economics.

Baker, A., Biazzo, I., Braunstein, A., Catania, G., Dall'Asta, L., Ingrosso, A., Krzakala, F., Mazza, F., Mézard, M., Muntoni, A. P., et al. (2021). Epidemic mitigation by statistical inference from contact tracing data. *Proceedings of the National Academy of Sciences*, 118(32):e2106548118.

Balas, E. and Ho, A. (1980). Set covering algorithms using cutting planes, heuristics, and subgradient optimization: a computational study. In *Combinatorial Optimization*, pages 37–60. Springer.

Bello-Chavolla, O. Y., Bahena-López, J. P., Antonio-Villa, N. E., Vargas-Vázquez, A., González-Díaz, A., Márquez-Salinas, A., Fermín-Martínez, C. A., Naveja, J. J., and Aguilar-Salinas, C. A. (2020). Predicting Mortality Due to SARS-CoV-2: A Mechanistic Score Relating Obesity and Diabetes to COVID-19 Outcomes in Mexico. *The Journal of Clinical Endocrinology & Metabolism*, 105(8):2752–2761.

Bengio, Y., Gupta, P., Maharaj, T., Rahaman, N., Weiss, M., Deleu, T., Müller, E. B., Qu, M., Schmidt, V., St-Charles, P., Alsdurf, H., Bilaniuk, O., Buckeridge, D. L., Marceau-Caron, G., Carrier, P. L., Ghosn, J., Ortiz-Gagne, S., Pal, C. J., Rish, I., Schölkopf, B., Sharma, A., Tang, J., and Williams, A. (2021). Predicting infectiousness for proactive contact tracing. In *9th International Conference on Learning Representations 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Bengio, Y., Gupta, P., Radovic, D., Scholl, M., Williams, A., de Witt, C. S., Zhang, T., and Zhang, Y. (2022). (private)-retroactive carbon pricing [(p) recap]: A market-based approach for climate finance and risk assessment. *arXiv preprint arXiv:2205.00666*.

Bengio, Y., Janda, R., Yu, Y. W., Ippolito, D., Jarvie, M., Pilat, D., Struck, B., Krastev, S., and Sharma, A. (2020). The need for privacy with public digital contact tracing during the COVID-19 pandemic. *The Lancet Digital Health*.

Bengio, Y., Lodi, A., and Prouvost, A. (2018). Machine learning for combinatorial optimization: a methodological tour d'horizon. *arXiv:1811.06128*.

Bergstra, J. and Bengio, Y. (2012). Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(10):281–305.

Bianconi, G., Sun, H., Rapisardi, G., and Arenas, A. (2021). Message-passing approach to epidemic tracing and mitigation with apps. *Physical Review Research*, 3(1):L012014.

Bojarski, M., Del Testa, D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., Jackel, L. D., Monfort, M., Muller, U., Zhang, J., et al. (2016). End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*.

Bonaccorsi, G., Pierri, F., Cinelli, M., Flori, A., Galeazzi, A., Porcelli, F., Schmidt, A. L., Valensise, C. M., Scala, A., Quattrociocchi, W., and Pammolli, F. (2020). Economic and social consequences of human mobility restrictions under COVID-19. *Proceedings of the National Academy of Sciences*, 117(27):15530–15535.

Briers, M., Charalambides, M., and Holmes, C. (2020). Risk scoring calculation for the current NHSx contact tracing app. *arXiv preprint arXiv:2005.11057*.

Briscese, G., Lacetera, N., Macis, M., and Tonin, M. (2020). Expectations, reference points, and compliance with covid-19 social distancing measures. Working Paper 26916, National Bureau of Economic Research.

Brisson, M., Gingras, G., Drolet, M., Laprise, J.-F., Mondor, M., Martin, D., Blanchette, C., and Demers, E. (2020). Épidémiologie et modélisation de l'évolution de la covid-19 au québec. Technical report, National Institute of Public Health of Quebec.

Buckman, J., Roy, A., Raffel, C., and Goodfellow, I. (2018). Thermometer encoding: One hot way to resist adversarial examples. In *International Conference on Learning Representations*.

Buitrago-Garcia, D., Egli-Gany, D., Counotte, M. J., Hossmann, S., Imeri, H., Ipekci, A. M., Salanti, G., and Low, N. (2020). Occurrence and transmission potential of asymptomatic and presymptomatic sars-cov-2 infections: A living systematic review and meta-analysis. *PLoS medicine*, 17(9):e1003346.

Burges, C. J. (2010). Technical report msr-tr-2010-82: From ranknet to lambdarank to lambdamart: An overview. Technical Report 23-581, Microsoft Research.

Cele, S., Jackson, L., Khoury, D. S., Khan, K., Moyo-Gwete, T., Tegally, H., San, J. E., Cromer, D., Scheepers, C., Amoako, D. G., et al. (2021). Omicron extensively but incompletely escapes pfizer bnt162b2 neutralization. *Nature*, pages 1–5.

Cencetti, G., Santin, G., Longa, A., Pigani, E., Barrat, A., Cattuto, C., Lehmann, S., Salathe, M., and Lepri, B. (2021). Digital proximity tracing on empirical contact networks for pandemic control. *Nature communications*, 12(1):1655.

Chebotar, Y., Handa, A., Makoviychuk, V., Macklin, M., Issac, J., Ratliff, N., and Fox, D. (2019). Closing the sim-to-real loop: Adapting simulation randomization with real world experience. In *2019 International Conference on Robotics and Automation*, pages 8973–8979. IEEE.

Cornuéjols, G., Sridharan, R., and Thizy, J.-M. (1991). A comparison of heuristics and relaxations for the capacitated plant location problem. *European Journal of Operational Research*, 50(3):280–297.

Coughlin, S. S. (1990). Recall bias in epidemiologic studies. *J. Clin. Epidemiol.*, 34(1):87–91.

Czypionka, T., Iftekhar, E. N., Prainsack, B., Priesemann, V., Bauer, S., Valdez, A. C., Cuschieri, S., Glaab, E., Grill, E., Krutzinna, J., et al. (2022). The benefits, costs and feasibility of a low incidence covid-19 strategy. *The Lancet Regional Health-Europe*, 13:100294.

Davies, N. G., Klepac, P., Liu, Y., Prem, K., Jit, M., and Eggo, R. M. (2020). Age-dependent effects in the transmission and control of covid-19 epidemics. *Nature medicine*, 26(8):1205–1211.

Detsky, A. S. and Naglie, I. G. (1990). A clinician's guide to cost-effectiveness analysis. *Annals of internal medicine*, 113(2):147–154.

Dhingra, B., Liu, H., Yang, Z., Cohen, W., and Salakhutdinov, R. (2017). Gated-attention readers for text comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1832–1846.

Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271.

Dilkina, B., Houtman, R., Gomes, C. P., Montgomery, C. A., McKelvey, K. S., Kendall, K., Graves, T. A., Bernstein, R., and Schwartz, M. K. (2017). Trade-offs and efficiencies in optimal budget-constrained multispecies corridor networks. *Conservation Biology*, 31(1):192–202.

Drummond, M. F., Sculpher, M. J., Claxton, K., Stoddart, G. L., and Torrance, G. W. (2015). *Methods for the economic evaluation of health care programmes*. Oxford university press.

Dumoulin, V., Perez, E., Schucher, N., Strub, F., Vries, H. d., Courville, A., and Bengio, Y. (2018). Feature-wise transformations. *Distill*, 3(7):e11.

Dumoulin, V., Shlens, J., and Kudlur, M. (2017). A learned representation for artistic style. In *5th International Conference on Learning Representations 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

El Emam, K., Jonker, E., Arbuckle, L., and Malin, B. (2011). A systematic review of re-identification attacks on health data. *PloS one*, 6(12):e28071.

Elson, D. (1972). Site location via mixed-integer programming. *Journal of the Operational Research Society*, 23(1):31–43.

Etheve, M., Alès, Z., Bissuel, C., Juan, O., and Kedad-Sidhoum, S. (2020). Reinforcement learning for variable selection in a branch and bound algorithm. In *International Conference on Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, pages 176–185. Springer.

Fan, K., Li, C., and Heller, K. (2016). A unifying variational inference framework for hierarchical graph-coupled hmm with an application to influenza infection. In *Proceedings of the Thirtieth Association for the Advancement of Artificial Intelligence Conference on Artificial Intelligence*, page 3828–3834.

Fayazi, S. A. and Vahidi, A. (2018). Mixed-integer linear programming for optimal scheduling of autonomous vehicle intersection crossing. *IEEE Transactions on Intelligent Vehicles*, 3(3):287–299.

Ferretti, L., Wymant, C., Kendall, M., Zhao, L., Nurtay, A., Abeler-Dörner, L., Parker, M., Bonsall, D., and Fraser, C. (2020). Quantifying sars-cov-2 transmission suggests epidemic control with digital contact tracing. *Science*, 368(6491).

Finn, F. (2005). Integer programming, linear programming and capital budgeting. *Abacus*, 9:180 – 192.

Floudas, C. A. and Lin, X. (2005). Mixed integer linear programming in process scheduling: Modeling, algorithms, and applications. *Annals of Operations Research*, 139(1):131–162.

Foster, B. A. and Ryan, D. M. (1976). An integer programming approach to the vehicle scheduling problem. *Journal of the Operational Research Society*, 27(2):367–384.

Gallacher, G. and Hossain, I. (2020). Remote work and employment dynamics under covid-19: Evidence from canada. *Canadian Public Policy*, pages Accepted–version.

Gandhi, M., Yokoe, D. S., and Havlir, D. V. (2020). Asymptomatic transmission, the achilles' heel of current strategies to control COVID-19. *New England Journal of Medicine*, 382(22):2158–2160.

Gasse, M., Chételat, D., Ferroni, N., Charlin, L., and Lodi, A. (2019). Exact combinatorial optimization with graph convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 15554–15566.

Gaythorpe, K., Imai, N., Cuomo-Dannenburg, G., Baguelin, M., Bhatia, S., Boonyasiri, A., and Cori, A. (2020). Report 8: Symptom progression of covid-19. Technical report, Imperial College London.

Geurts, P., Ernst, D., and Wehenkel, L. (2006). Extremely randomized trees. *Machine learning*, 63(1):3–42.

Gilpin, L. H., Bau, D., Yuan, B. Z., Bajwa, A., Specter, M., and Kagal, L. (2018). Explaining explanations: An overview of interpretability of machine learning. In *2018 IEEE 5th International Conference on data science and advanced analytics*, pages 80–89. IEEE.

Glasmachers, T. (2017). Limits of end-to-end learning. In *Asian conference on machine learning*, pages 17–32. Proceedings of Machine Learning Research.

Gleixner, A., Bastubbe, M., Eifler, L., Gally, T., Gamrath, G., Gottwald, R. L., Hendel, G., Hojny, C., Koch, T., Lübbecke, M. E., Maher, S. J., Miltenberger, M., Müller, B., Pfetsch, M. E., Puchert, C., Rehfeldt, D., Schlösser, F., Schubert, C., Serrano, F., Shinano, Y., Viernickel, J. M., Walter, M., Wegscheider, F., Witt, J. T., and Witzig, J. (2018). The SCIP Optimization Suite 6.0. Technical report, Optimization Online.

Global Burden of Disease Collaborative Network (2018). Global burden of disease study 2017 (gbd 2017) disability weights. Technical report, Seattle, United States of America: Institute for Health Metrics and Evaluation.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. `http://www.deeplearningbook.org`.

Goyal, A., Cardozo-Ojeda, E. F., and Schiffer, J. T. (2020). Potency and timing of antiviral therapy as determinants of duration of sars-cov-2 shedding and intensity of inflammatory response. *Science advances*, 6(47):eabc7112.

Grantz, K. H., Lee, E. C., D'Agostino McGowan, L., Lee, K. H., Metcalf, C. J. E., Gurley, E. S., and Lessler, J. (2021). Maximizing and evaluating the impact of test-trace-isolate programs: A modeling study. *PLoS medicine*, 18(4):e1003585.

Grefenstette, J. J., Brown, S. T., Rosenfeld, R., DePasse, J., Stone, N. T., Cooley, P. C., Wheaton, W. D., Fyshe, A., Galloway, D. D., Sriram, A., et al. (2013). Fred (a framework for reconstructing epidemic dynamics): an open-source software system for modeling infectious diseases and control strategies using census-based populations. *BMC public health*, 13(1):1–14.

Gupta, P., Gasse, M., Khalil, E., Mudigonda, P., Lodi, A., and Bengio, Y. (2020a). Hybrid models for learning to branch. *Advances in neural information processing systems*, 33:18087–18097.

Gupta, P., Khalil, E. B., Chetélat, D., Gasse, M., Bengio, Y., Lodi, A., and Kumar, M. P. (2022). Lookback for learning to branch. *Transactions of Machine Learning Research*.

Gupta, P., Maharaj, T., Rahaman, N., Weiss, M., et al. (2023). Proactive contact tracing. *PLOS Digit Health*.

Gupta, P., Maharaj, T., Weiss, M., Rahaman, N., Alsdurf, H., Sharma, A., Minoyan, N., Harnois-Leblanc, S., Schmidt, V., Charles, P.-L. S., Deleu, T., Williams, A., Patel, A., Qu, M., Bilaniuk, O., Caron, G. M., Carrier, P.-L., Ortiz-Gagne, S., Rousseau, M.-A., Buckeridge, D., Ghosn, J., Zhang, Y., Schölkopf, B., Tang, J., Rish, I., Pal, C., Merckx, J., Muller, E. B., and Bengio, Y. (2020b). COVI-agentsim: an agent-based model for evaluating methods of digital contact tracing.

Ha, D., Dai, A., and Le, Q. V. (2017). Hypernetworks. *International Conference on Learning Representations*.

He, H., Daume III, H., and Eisner, J. M. (2014a). Learning to search in branch and bound algorithms. *Advances in neural information processing systems*, 27.

He, H., Daumé, H. I., and Eisner, J. (2014b). Learning to search in branch-and-bound algorithms. In *Advances in Neural Information Processing Systems 27*, pages 3293–3301.

He, X., Lau, E. H. Y., Wu, P., Deng, X., Wang, J., Hao, X., Lau, Y. C., Wong, J. Y., Guan, Y., Tan, X., Mo, X., Chen, Y., Liao, B., Chen, W., Hu, F., Zhang, Q., Zhong, M., Wu, Y., Zhao, L., Zhang, F., Cowling, B. J., Li, F., and Leung, G. M. (2020). Temporal dynamics in viral shedding and transmissibility of COVID-19. *Nature Medicine*, 26(5):672–675.

Hearst, M. A. (1998). Support vector machines. *IEEE Intelligent Systems*, 13(4):18–28.

Hellewell, J., Abbott, S., Gimma, A., Bosse, N. I., Jarvis, C. I., Russell, T. W., Munday, J. D., Kucharski, A. J., Edmunds, W. J., Sun, F., et al. (2020). Feasibility of controlling covid-19 outbreaks by isolation of cases and contacts. *The Lancet Global Health*.

Heneghan, C., Brassey, J., and Jefferson, T. (2020). Sars-cov-2 viral load and the severity of covid-19. `https://www.cebm.net/covid-19/sars-cov-2-viral-load-and-the-severity-of-covid-19/`. Accessed: 2020-06-02.

Herbrich, R., Rastogi, R., and Vollgraf, R. (2020). Crisp: A probabilistic model for individual-level COVID-19 infection risk estimation based on contact data.

Hinch, R., Probert, W., Nurtay, A., Kendall, M., Wymant, C., Hall, M., and Fraser, C. (2020). Effective configurations of a digital contact tracing app: A report to NHSx. `https://cdn.theconversation.com/static_files/files/1009/Report_-_Effective_App_Configurations.pdf`.

Hinton, G., Vinyals, O., and Dean, J. (2015). Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Hinton, G. E., Dayan, P., Frey, B. J., and Neal, R. M. (1995). The" wake-sleep" algorithm for unsupervised neural networks. *Science*, 268(5214):1158–1161.

Huang, Z., Wang, K., Liu, F., Zhen, H.-L., Zhang, W., Yuan, M., Hao, J., Yu, Y., and Wang, J. (2022). Learning to select cuts for efficient mixed-integer programming. *Pattern Recognition*, 123:108353.

Hutter, F., Hoos, H. H., and Leyton-Brown, K. (2010). Automated configuration of mixed integer programming solvers. In *International Conference on Integration of Artificial Intelligence (AI) and Operations Research (OR) Techniques in Constraint Programming*, pages 186–202. Springer.

Ivers, L. and Weitzner, D. (2020). Can digital contact tracing make up for lost time? *The Lancet Public Health*, 5.

James, S. L. and al (2018). Global, regional, and national incidence, prevalence, and years lived with disability for 354 diseases and injuries for 195 countries and territories, 1990–2017: a systematic analysis for the global burden of disease study 2017. *The Lancet*, 392(10159):1789 – 1858.

Kantor, I., Robineau, J.-L., Bütün, H., and Maréchal, F. (2020). A mixed-integer linear programming formulation for optimizing multi-scale material and energy integration. *Frontiers in Energy Research*, 8:49.

Keeling, M. J., Hollingsworth, T. D., and Read, J. M. (2020). Efficacy of contact tracing for the containment of the 2019 novel coronavirus (covid-19). *J. Epidemiol. Community Health*, 74:861–866.

Khalil, E. B., Bodic, P. L., Song, L., Nemhauser, G., and Dilkina, B. (2016). Learning to branch in mixed integer programming. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pages 724–731.

Khalil, E. B., Morris, C., and Lodi, A. (2022). MIP-GNN: A Data-Driven Framework for Guiding Combinatorial Solvers. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(9):10219–10227.

Killerby, M. E., Link-Gelles, R., Haight, S. C., Schrodt, C. A., England, L., Gomes, D. J., Shamout, M., Pettrone, K., O'Laughlin, K., Kimball, A., et al. (2020). Characteristics associated with hospitalization among patients with covid-19—metropolitan atlanta, georgia, march–april 2020. *Morbidity and Mortality Weekly Report*, 69(25):790.

Kim, T., Song, I., and Bengio, Y. (2017). Dynamic layer normalization for adaptive neural acoustic modeling in speech recognition. *Proc. Interspeech 2017*, pages 2411–2415.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kingma, D. P. and Welling, M. (2013). Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Kirlik, G. and Sayın, S. (2014). A new algorithm for generating all nondominated solutions of multiobjective discrete optimization problems. *European Journal of Operational Research*, 232(3):479–488.

Kleinman, R. A. and Merkel, C. (2020). Digital contact tracing for covid-19. *CMAJ*, 192(24):E653–E656.

Klepac, P., Kucharski, A. J., Conlan, A. J., Kissler, S., Tang, M., Fry, H., and Gog, J. R. (2020). Contacts in context: large-scale setting-specific social mixing matrices from the bbc pandemic project.

Kretzschmar, M. E., Rozhnova, G., Bootsma, M. C., van Boven, M., van de Wijgert, J. H., and Bonten, M. J. (2020). Impact of delays on effectiveness of contact tracing strategies for covid-19: a modelling study. *The Lancet Public Health*, 5(8):e452–e459.

Krueger, D., Maharaj, T., and Leike, J. (2020). Hidden incentives for auto-induced distributional shift. *arXiv preprint arXiv:2009.09153*.

Kullback, S. and Leibler, R. A. (1951). On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86.

Kyu, H. H., Abate, D., Abate, K. H., Abay, S. M., Abbafati, C., Abbasi, N., Abbastabar, H., Abd-Allah, F., Abdela, J., Abdelalim, A., et al. (2018). Global, regional, and national disability-adjusted life-years (dalys) for 359 diseases and injuries and healthy life expectancy (hale) for 195 countries and territories, 1990–2017: a systematic analysis for the global burden of disease study 2017. *The Lancet*, 392(10159):1859–1922.

Land, A. H. and Doig, A. G. (1960). An automatic method of solving discrete programming problems. *Econometrica*, 28(3):pp. 497–520.

Lau, M. S., Grenfell, B., Thomas, M., Bryan, M., Nelson, K., and Lopman, B. (2020). Characterizing superspreading events and age-specific infectiousness of SARS-CoV-2 transmission in Georgia, USA. *Proceedings of the National Academy of Sciences*, 117(36):22430–22435.

Lauer, S. A., Grantz, K. H., Bi, Q., Jones, F. K., Zheng, Q., Meredith, H. R., Azman, A. S., Reich, N. G., and Lessler, J. (2020). The incubation period of coronavirus disease 2019 (COVID-19) from publicly reported confirmed cases: estimation and application. *Annals of internal medicine*, 172(9):577–582.

Laupacis, A., Feeny, D., Detsky, A. S., and Tugwell, P. X. (1992). How attractive does a new technology have to be to warrant adoption and utilization? tentative guidelines for using clinical and economic evaluations. *Canadian Medical Association Journal*, 146(4):473.

Lee, J., Lee, Y., Kim, J., Kosiorek, A., Choi, S., and Teh, Y. W. (2019). Set transformer: A framework for attention-based permutation-invariant neural networks. In *International conference on machine learning*, pages 3744–3753. PMLR.

Levine, S., Kumar, A., Tucker, G., and Fu, J. (2020). Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*.

Leyton-Brown, K., Pearson, M., and Shoham, Y. (2000). Towards a universal test suite for combinatorial auction algorithms. In *Proceedings of the 2nd ACM conference on Electronic commerce*, pages 66–76.

Li, D., Wang, D., Dong, J., Wang, N., Huang, H., Xu, H., and Xia, C. (2020). False-negative results of real-time reverse-transcriptase polymerase chain reaction for severe acute respiratory syndrome coronavirus 2: role of deep-learning-based ct diagnosis and insights from two cases. *Korean journal of radiology*, 21(4):505–508.

Liebel, L. and Körner, M. (2018). Auxiliary tasks in multi-task learning. *arXiv preprint arXiv:1805.06334*.

Linderoth, J. and Savelsbergh, M. (1999). A computational study of search strategies for mixed integer programming. *INFORMS Journal on Computing*, 11:173–187.

Liu, W., Lin, R., Liu, Z., Liu, L., Yu, Z., Dai, B., and Song, L. (2018). Learning towards minimum hyperspherical energy. In *Advances in neural information processing systems*, pages 6222–6233.

Lodi, A. and Zarpellon, G. (2017). On learning and branching: a survey. *TOP*, 25:207–236.

Lokhov, A. Y., Mézard, M., Ohta, H., and Zdeborová, L. (2014). Inferring the origin of an epidemic with a dynamic message-passing algorithm. *Physical Review E*, 90(1).

Lorch, L., Kremer, H., Trouleau, W., Tsirtsis, S., Szanto, A., Schölkopf, B., and Gomez-Rodriguez, M. (2022). Quantifying the effects of contact tracing, testing, and containment measures in the presence of infection hotspots. *ACM Trans. Spatial Algorithms Syst.*, 8(4).

Luo, L., Liu, D., Liao, X.-L., Wu, X.-B., Jing, Q.-L., Zheng, J.-Z., and Fang-Hua Liu, e. a. (2020). Modes of contact and risk of transmission in covid-19 among close contacts. *medRxiv*.

Mandel, A. and Veetil, V. (2020). The economic cost of covid lockdowns: An out-of-equilibrium analysis. *Economics of Disasters and Climate Change*.

Martin, T., Karopoulos, G., Hernández-Ramos, J. L., Kambourakis, G., and Nai Fovino, I. (2020). Demystifying covid-19 digital contact tracing: A survey on frameworks and mobile apps. *Wireless Communications and Mobile Computing*, 2020.

Maziarka, Ł., Danel, T., Mucha, S., Rataj, K., Tabor, J., and Jastrzębski, S. (2020). Molecule attention transformer. *arXiv preprint arXiv:2002.08264*.

Mildenhall, B., Srinivasan, P. P., Tancik, M., Barron, J. T., Ramamoorthi, R., and Ng, R. (2020). Nerf: Representing scenes as neural radiance fields for view synthesis. *arXiv preprint arXiv:2003.08934*.

Minka, T. P. (2001). Expectation propagation for approximate bayesian inference. In *Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence*, UAI'01, page 362–369, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Mossong, J., Hens, N., Jit, M., Beutels, P., Auranen, K., Mikolajczyk, R., Massari, M., Salmaso, S., Tomba, G. S., Wallinga, J., Heijne, J., Sadkowska-Todys, M., Rosinska, M., and Edmunds, W. J. (2008). Social contacts and mixing patterns relevant to the spread of infectious diseases. *PLOS Medicine*, 5(3):1.

Munzert, S., Selb, P., Gohdes, A., Stoetzer, L. F., and Lowe, W. (2021). Tracking and promoting the usage of a covid-19 contact tracing app. *Nature Human Behaviour*, 5(2):247–255.

Murdoch, W. J., Singh, C., Kumbier, K., Abbasi-Asl, R., and Yu, B. (2019). Interpretable machine learning: definitions, methods, and applications. *arXiv preprint arXiv:1901.04592*.

Murphy, K., Kumar, A., and Serghiou, S. (2021). Risk score learning for covid-19 contact tracing apps. In *Machine Learning for Healthcare Conference*, pages 373–390. PMLR.

Murphy, K. P., Weiss, Y., and Jordan, M. I. (1999). Loopy belief propagation for approximate inference: an empirical study. In *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*, pages 467–475.

Murray, C. J. (2022). Covid-19 will continue but the end of the pandemic is near. *The Lancet*.

Nair, V., Bartunov, S., Gimeno, F., von Glehn, I., Lichocki, P., Lobov, I., O'Donoghue, B., Sonnerat, N., Tjandraatmadja, C., Wang, P., et al. (2020). Solving mixed integer programs using neural networks. *arXiv preprint arXiv:2012.13349*.

Ng, A. Y. (2004). Feature selection, l1 vs. l2 regularization, and rotational invariance. In *Proceedings of the twenty-first international conference on Machine learning*, page 78.

Nowatzki, T., Ferris, M., Sankaralingam, K., Estan, C., Vaish, N., and Wood, D. (2013). Optimization and mathematical modeling in computer architecture. *Synthesis Lectures on Computer Architecture*, 8(4):1–144.

Nurchis, M. C., Pascucci, D., Sapienza, M., Villani, L., D'Ambrosio, F., Castrini, F., Specchia, M. L., Laurenti, P., and Damiani, G. (2020). Impact of the burden of covid-19 in italy: Results of disability-adjusted life years (dalys) and productivity loss. *International Journal of Environmental Research and Public Health*, 17(12):4233.

Papadimitriou, C. H. and Steiglitz, K. (1982). *Combinatorial Optimization: Algorithms and Complexity*. Prentice-Hall, Inc., USA.

Papadimitriou, C. H. and Steiglitz, K. (1998). *Combinatorial optimization: algorithms and complexity*. Courier Corporation.

Parsonson, C. W., Laterre, A., and Barrett, T. D. (2022). Reinforcement learning for branch-and-bound optimisation using retrospective trajectories. *arXiv preprint arXiv:2205.14345*.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. (2017). Automatic differentiation in pytorch. *Advances in neural information processing systems*.

Peng, C. and Liao, B. (2022). Heavy-head sampling for fast imitation learning of machine learning based combinatorial auction solver. *Neural Processing Letters*, pages 1–14.

Perez, E., Strub, F., De Vries, H., Dumoulin, V., and Courville, A. (2018). FiLM: Visual reasoning with a general conditioning layer. In *Thirty-Second Association for the Advancement of Artificial Intelligence Conference on Artificial Intelligence*.

Petrilli, C. M., Jones, S. A., Yang, J., Rajagopalan, H., O'Donnell, L., Chernyak, Y., Tobin, K. A., Cerfolio, R. J., Francois, F., and Horwitz, L. I. (2020). Factors associated with hospital admission and critical illness among 5279 people with coronavirus disease 2019 in new york city: prospective cohort study. *BMJ*, 369.

Pochet, Y. and Wolsey, L. A. (2006). *Production planning by mixed integer programming*. Springer Science & Business Media.

Prem, K., Cook, A., and Jit, M. (2017). Projecting social contact matrices in 152 countries using contact surveys and demographic data. *PLoS Computational Biology*, 13.

Rahaman, N., Baratin, A., Arpit, D., Draxler, F., Lin, M., Hamprecht, F., Bengio, Y., and Courville, A. (2019). On the spectral bias of neural networks. In *International Conference on Machine Learning*, pages 5301–5310. Proceedings of Machine Learning Research.

Robilotti, E. V., Babady, N. E., Mead, P. A., Rolling, T., Perez-Johnston, R., Bernardes, M., Bogler, Y., Caldararo, M., Figueroa, C. J., Glickman, M. S., et al. (2020). Determinants of covid-19 disease severity in patients with cancer. *Nature medicine*, 26(8):1218–1223.

Rodríguez, P., Graña, S., Alvarez-León, E. E., Battaglini, M., Darias, F. J., Hernán, M. A., López, R., Llaneza, P., Martín, M. C., Ramirez-Rubio, O., et al. (2021). A population-based controlled experiment assessing the epidemiological impact of digital contact tracing. *Nature communications*, 12(1):1–6.

Rossi, R., Socci, V., Talevi, D., Mensi, S., Niolu, C., Pacitti, F., Di Marco, A., Rossi, A., Siracusano, A., and Di Lorenzo, G. (2020). Covid-19 pandemic and lockdown measures impact on mental health among the general population in italy. *Frontiers in Psychiatry*, 11:790.

Roth, G. A., Abate, D., Abate, K. H., Abay, S. M., Abbafati, C., Abbasi, N., Abbastabar, H., Abd-Allah, F., Abdela, J., Abdelalim, A., et al. (2018). Global, regional, and national age-sex-specific mortality for 282 causes of death in 195 countries and territories, 1980–2017: a systematic analysis for the global burden of disease study 2017. *The Lancet*, 392(10159):1736–1788.

Roth, K., Milbich, T., Sinha, S., Gupta, P., Ommer, B., and Cohen, J. P. (2020). Revisiting training strategies and generalization performance in deep metric learning. In *International Conference on Machine Learning*, pages 8242–8252. PMLR.

Rudin, C. (2019). Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature machine intelligence*, 1(5):206–215.

Russell, A. (2020). The rise of coronavirus hate crimes. *The New Yorker*.

Sadeghi, F. and Levine, S. (2016). Cad2rl: Real single-image flight without a single real image. *arXiv preprint arXiv:1611.04201*.

Satorras, V. G. and Welling, M. (2021). Neural enhanced belief propagation on factor graphs. *Proceedings of the 24th International Conference on Artificial Intelligence and Statistics (AISTATS)*.

Scavuzzo, L., Chen, F. Y., Chételat, D., Gasse, M., Lodi, A., Yorke-Smith, N., and Aardal, K. (2022). Learning to branch with tree mdps. *arXiv preprint arXiv:2205.11107*.

Shaw, P., Uszkoreit, J., and Vaswani, A. (2018). Self-attention with relative position representations. *arXiv preprint arXiv:1803.02155*.

Squazzoni, F., Polhill, J. G., Edmonds, B., Ahrweiler, P., Antosz, P., Scholz, G., Chappin, E., Borit, M., Verhagen, H., Giardini, F., et al. (2020). Computational models that matter during a global pandemic outbreak: A call to action. *JASSS-The Journal of Artificial Societies and Social Simulation*, 23(2):10.

Srivastava, R. K., Greff, K., and Schmidhuber, J. (2015). Highway networks. *arXiv preprint arXiv:1505.00387*.

Stevens, H. (2020). Why outbreaks like coronavirus spread exponentially, and how to "flatten the curve".

Sun, H., Chen, W., Li, H., and Song, L. (2020). Improving learning to branch via reinforcement learning. *Workshop on Learning Meets Combinatorial Algorithms*.

Sweeney, L. (2002). K-anonymity: A model for protecting privacy. *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, 10(5):557–570.

Thomson, J. (1904). XXIV. on the structure of the atom: an investigation of the stability and periods of oscillation of a number of corpuscles arranged at equal intervals around the circumference of a circle; with application of the results to the theory of atomic structure. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 7(39):237–265.

To, K. K.-W., Tsang, O. T.-Y., Leung, W.-S., Tam, A. R., Wu, T.-C., Lung, D. C., Yip, C. C.-Y., Cai, J.-P., Chan, J. M.-C., Chik, T. S.-H., and al. (2020). Temporal profiles of viral load in posterior oropharyngeal saliva samples and serum antibody responses during infection by sars-cov-2: an observational cohort study. *The Lancet Infectious Diseases*.

Tobin, J., Fong, R., Ray, A., Schneider, J., Zaremba, W., and Abbeel, P. (2017). Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 23–30. IEEE.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Verity, R., Okell, L. C., Dorigatti, I., Winskill, P., Whittaker, C., Imai, N., Cuomo-Dannenburg, G., Thompson, H., Walker, P. G., Fu, H., et al. (2020). Estimates of the severity of coronavirus disease 2019: a model-based analysis. *The Lancet infectious diseases*.

Vinceti, M., Filippini, T., Rothman, K. J., Ferrari, F., Goffi, A., Maffeis, G., and Orsini, N. (2020). Lockdown timing and efficacy in controlling covid-19 using mobile phone tracking. *The Lancet Clinical Medicine*.

Watson, C., Cicero, A., Blumenstock, J., and Fraser, M. (2020). A national plan to enable comprehensive COVID-19 case finding and contact tracing in the us. Technical report, John Hopkins - Bloomberg school of public health - Center for health security.

Williamson, E. J., Walker, A. J., Bhaskaran, K., Bacon, S., Bates, C., Morton, C. E., Curtis, H. J., Mehrkar, A., Evans, D., Inglesby, P., et al. (2020). Factors associated with covid-19-related death using opensafely. *Nature*, 584(7821):430–436.

Winn, J. and Bishop, C. M. (2005). Variational message passing. *Journal of Machine Learning Research*, 6(Apr):661–694.

Wolsey, L. A. (1988). *Integer Programming*. Wiley-Blackwell.

Wood, F., Warrington, A., Naderiparizi, S., Weilbach, C., Masrani, V., Harvey, W., Ścibior, A., Beronov, B., Grefenstette, J., Campbell, D., et al. (2022). Planning as inference in epidemiological dynamics models. *Frontiers in Artificial Intelligence*, 4:550603.

Woodhams, S. (2020). Covid-19 digital rights tracker - contact tracing apps. Technical report, Top10VPN.com. Accessed: 2020-06-11.

World Health Organization (2018). WHO methods and data sources for global burden of disease estimates 2000–2016. `http://www.who.int/healthinfo/global_burden_disease/GlobalDALY_method_2000_2016.pdf`.

Wymant, C., Ferretti, L., Tsallis, D., Charalambides, M., Abeler-Dörner, L., Bonsall, D., Hinch, R., Kendall, M., Milsom, L., Ayres, M., et al. (2021). The epidemiological impact of the nhs covid-19 app. *Nature*, 594(7863):408–412.

Yedidia, J. S., Freeman, W. T., and Weiss, Y. (2000). Generalized belief propagation. In *Proceedings of the 13th International Conference on Neural Information Processing Systems*, page 668–674.

Zaheer, M., Kottur, S., Ravanbakhsh, S., Poczos, B., Salakhutdinov, R. R., and Smola, A. J. (2017). Deep sets. In *Advances in neural information processing systems*, pages 3391–3401.

Zarpellon, G., Jo, J., Lodi, A., and Bengio, Y. (2021). Parameterizing branch-and-bound search trees to learn branching policies. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 3931–3939.

Zhang, T., Banitalebi-Dehkordi, A., and Zhang, Y. (2022a). Deep reinforcement learning for exact combinatorial optimization: Learning to branch. In *2022 26th International Conference on Pattern Recognition (ICPR)*, pages 3105–3111, Los Alamitos, CA, USA. IEEE Computer Society.

Zhang, T., Williams, A., Phade, S., Srinivasa, S., Zhang, Y., Gupta, P., Bengio, Y., and Zheng, S. (2022b). AI for Global Climate Cooperation: Modeling Global Climate Negotiations, Agreements, and Long-Term Cooperation in RICE-N. In *AAAI 2022 Fall Symposium: The Role of AI in Responding to Climate Challenges*.

Zimmerman, S., Sloane, P. D., Katz, P. R., Kunze, M., O'Neil, K., and Resnick, B. (2020). The need to include assisted living in responding to the covid-19 pandemic. *Journal of the American Medical Directors Association*, 21(5):572–575.