MIE1666H: Machine Learning for Mathematical Optimization

Fall 2021

Instructor:	Prof. Elias B. Khalil, khalil@mie.utoronto.ca
Office hours:	Q&A after every lecture
Text:	Various suggested readings (URLs to be shared on Quercus)
Official Prereqs.:	MIE335H or MIE1603H or MIE1619H or MIE1516 or CSC311H or equiv.
Lectures:	Mondays 3:00-6:00pm, Bahen 2139, starting September 13, 2021
Slack:	https://slack.com/xxx
Piazza:	https://piazza.com/xxx

Course description

Mathematical optimization algorithms are used to solve a wide variety of decision-making tasks. The design of optimization algorithms often requires substantial theoretical insights or algorithmic engineering, both of which are manual, tedious tasks. This course introduces automated machine learning approaches for improving optimization algorithms in the presence of a historical dataset or a generator of problem instances from a domain of interest. Topics include automated algorithm configuration, modeling iterative heuristics in the reinforcement learning framework, deep neural networks for modeling combinatorial optimization problems, guiding exact solvers with learned search strategies, learning-theoretic guarantees, and benchmarking/computational considerations. The focus will be on discrete optimization in the integer programming framework, both exact and heuristic. Along with programming, knowledge in at least two of the following is required: algorithm design, integer programming, combinatorial optimization, machine learning; knowledge of deep learning is helpful but not required.

This is an advanced graduate-level course intended for research-stream students. *MEng students may register for the course but should expect that it will be more research-oriented than their typical courses.*

By the end of this course, the student is expected to have improved at general academic skills such as writing papers, reviewing peers', and delivering high-quality oral presentations. The main evaluation will be a group project where students are expected to either apply techniques discussed in the course to their own research interests, or develop new models/algorithms that improve over existing ones from the literature. A goal of this course is that these projects will be publishable in a peer-reviewed forum.

Prerequisites

Registered students need not satisfy all of the prerequisites simultaneously, as basic elements of machine learning and mathematical optimization will be reviewed early on in the course.

- **Algorithm design** and data structures through MIE335H (Algorithms and Numerical Methods) or equivalent, or Computer Science background at the undergraduate level;
- **Mathematical optimization** through one of MIE1603H (Integer Programming) or MIE1619H (Constraint Programming and Hybrid Optimization). Alternatively, advanced understanding of continuous optimization along with basic knowledge of combinatorial algorithms is sufficient;

- **Machine Learning** through MIE1616H (Structured Learning and Inference) or CSC311H (Introduction to Machine Learning) or equivalent;
- **Programming experience** with Python and common open-source machine learning packages;

Course goals

- Students will develop a broad understanding of the different ways in which an optimization algorithm can be automatically tailored to a set of instances using machine learning.
- Students will learn how to identify algorithmic tasks that can benefit the most from machine learning.
- Students will learn about existing methods for improving exact integer programming solvers with machine learning.
- Students will learn to use feature engineering and deep learning models to extract features of integer programming problems.
- Students will learn to use Graph Neural Networks to model combinatorial problems.
- Students will use solid empirical methods to evaluate learning and non-learning approaches for tuning algorithms along various axes (data requirements, training time, prediction time, solver performance metrics, etc.).
- Students will learn to use Reinforcement Learning to represent iterative optimization heuristics and improve them automatically.
- By means of reading assignments, students will develop a deep understanding of the literature relating to a subtopic of relevance to their own research.
- By means of the course project, students will tackle a new research question or reproduce existing research, contributing to their research, writing, and presentation skills.
- By means of the course project, students will learn to integrate machine learning models within an optimization solver or heuristic.

Evaluation

Component		Weight	Notes
Practical Assignments (2)		20	A1 (5), A2 (15)
Peer Review (1)		15	
Project		65	
Proposal	0		
Checkpoint	5		
Initial Submission	15		
Presentation	15		
Final Submission	30		

Practical Assignments (20%)

These two individual assignments will serve as practical introductions to modeling/solving integer programs and to using machine learning to predict solver runtimes, respectively. Submissions will consist of a code repository and a short report summarizing the solution and the results. Typically, Python will be used in conjunction with an integer programming solver (e.g., SCIP, Gurobi, CPLEX) and machine learning libraries (e.g., PyTorch, sklearn, TensorFlow).

1. **Modeling and Solving Integer Programs (5%), Due Early October.** Following introductory lectures on the topic (Weeks 1, 2), you will be asked to use an integer programming solver to model an operational problem inspired by a real application. You will then generate synthetic instances and attempt to solve them using the solver. A report will summarize your findings on how the solver performs using evaluation metrics from Week 3.
2. **Predicting Solver Runtimes (15%), Due Late October.** Runtime prediction is a central task in algorithm configuration, selection, portfolios, and learning-to-optimize more generally. You will be given a set of already-solved integer programming instances from the same family of problems along with the running times of a solver. You will then develop regression models that predict solver running times. A report will summarize the models you used, tricks to improve performance, feature engineering or deep learning architectures, etc.

Peer review (15%)

1. **Review of an initial project submission (15%), Due Mid November.** Each student will be assigned another project's initial submission. You will write a detailed review following best practices/guidelines that are recommended at top machine learning conferences such as NeurIPS and ICLR.

Group Project (65%)

Full project details will be released during Week 2.

The project is **mandatory** for the purpose of completing this course. The goal is for **a group of 2–4 students** to apply newly acquired knowledge in the design of learning-based approaches for improving optimization algorithms. Besides implementation, you will get to read relevant research papers, devise evaluation protocols, and present your work to your classmates. The work you'll do will be similar to how a research paper is developed. You are encouraged to identify project topics that are relevant to common research interests of the group members (e.g., application domains,

emerging deep learning architectures, optimization under uncertainty, etc.). All topics must be approved by the instructor, who will provide guidance during the selection process.

1. **Proposal (0%), Due Oct 11 (Week 5)**: A 1–2 pager describing the goal of the project and solution approaches being considered. The instructor will then approve the projects or request modifications.
2. **Checkpoint (5%), Due Nov 01 (Week 8)**: A short report consisting of a preliminary literature review, initial experiments and hypotheses, and a plan for the rest of the project along with each group member's primary foci.
3. **Initial Submission (15%), Due Nov 15 (Week 10)**: An initial draft written in the format of a research paper with all relevant sections and current partial results. This report will be reviewed by the instructors and 2-3 peers. The initial submission should be updated to answer these reviews and include more complete results, ultimately converging into the Final Submissions.
4. **Presentation (15%), On Nov 29 & Dec 06 (Weeks 12, 13)**: A 20-minute conference-style group presentation followed by a 5-minute Q%A. You should present your work in a way that is accessible to your classmates.
5. **Final Submission (30%), Due end of Finals week, date TBD**: A final paper that is almost ready for submission to a conference. Evaluation will be based on the improvements you've made to the Initial Submission based on the reviews, as well as the actual content and writing.

Assignment submissions (no extensions)

- Assignment submissions will be online through GitHub or Quercus; instructions will be provided in the Assignments themselves.
- Assignments up to 48h late will be given a 30% penalty.

Class Participation

There is no explicit mark for class participation. However, marks for any of the above may be adjusted to account for lack of participation. Students are expected to actively participate in all lectures by asking and answering questions, making suggestions, asking for and providing clarification of important concepts. Students are expected to have read the required readings for each lecture and to have made use of the additional material in areas where they are confused. The level of participation in the class will be used as a guide to the extent that the students have completed and understood the readings.

Questions and discussions

Technical questions on the assignments or project should be posted to Piazza. There is no guarantee that answers to questions will be provided on weekends. The Slack channel will be used for open discussions between members of groups for the project and for sharing relevant resources beyond the material in class.

Schedule of topics

The schedule of topics below is subject to change.

Week	Lecture	Readings	Notes
01 (Sep 13)	Course overview	Smith (1999)	
02 (Sep 20)	Introduction to integer programming	Wolsey, Ch. 1, 2, 7, 13	
03 (Sep 27)	Empirical algorithmics	Hooker (1995), SLS, Ch. 1, 4	**Assignment 1 due**
04 (Oct 04)	Algorithm configuration	Hoos (2012), Rice (1976)	
05 (Oct 11)	**Official holiday – No Lecture**		
06 (Oct 18)	Learning in exact algorithms	Lodi (2013), Bengio et al. (2021)	**Assignment 2 due**
07 (Oct 25)	Learning to branch: a case study	Zarpellon & Lodi (2018), Khalil et al. (2016)	
08 (Nov 01)	Reinforcement learning for algorithms	Mazyavkina et al. (2021), Dai et al. (2017)	
09 (Nov 08)	Graph Neural Networks for comb. opt.	Cappart et al. (2021)	
10 (Nov 15)	Learning-theoretic guarantees	Roughgarden, Ch. 29	
11 (Nov 22)	Other learning settings	Roughgarden, Ch. 25, 30	
12 (Nov 29)	**Project presentations**		
13 (Dec 06)	**Project presentations**		

Primary References

The list below is subject to change.

Books

- Hoos, Holger H., and Thomas Stützle. Stochastic local search: Foundations and applications. Elsevier, 2004. [link]
- Wolsey, Laurence A. Integer programming. John Wiley & Sons, 2020. [link]
- Roughgarden, Tim, ed. Beyond the Worst-Case Analysis of Algorithms. Cambridge University Press, 2020. [link]

Surveys

- Smith, Kate A. "Neural networks for combinatorial optimization: a review of more than a decade of research." INFORMS Journal on Computing 11.1 (1999): 15-34. [link]
- Hoos, Holger H. "Automated Algorithm Configuration and Parameter Tuning." Autonomous Search, Springer Berlin Heidelberg, pp. 37–71. [link]
- Bengio, Yoshua, Andrea Lodi, and Antoine Prouvost. "Machine learning for combinatorial optimization: a methodological tour d'horizon." European Journal of Operational Research 290.2 (2021): 405-421. [link]
- Lodi, Andrea, and Giulia Zarpellon. "On learning and branching: a survey." Top 25.2 (2017): 207-236. [link]
- Mazyavkina, Nina, et al. "Reinforcement learning for combinatorial optimization: A survey." Computers & Operations Research (2021): 105400. [link]
- Cappart, Quentin, et al. "Combinatorial optimization and reasoning with graph neural networks." arXiv preprint arXiv:2102.09544 (2021). [link]
- Hutter, Frank, et al. "Algorithm runtime prediction: Methods & evaluation." Artificial Intelligence 206 (2014): 79-111. [link]

Position papers

- Lodi, Andrea. "The heuristic (dark) side of MIP solvers." Hybrid metaheuristics. Springer, Berlin, Heidelberg, 2013. 273-284. [link]
- Hooker, John N. "Testing heuristics: We have it all wrong." Journal of heuristics 1.1 (1995): 33-42. [link]

Land acknowledgment

I (we) wish to acknowledge this land on which the University of Toronto operates. For thousands of years it has been the traditional land of the Huron-Wendat, the Seneca, and most recently, the Mississaugas of the Credit River. Today, this meeting place is still the home to many Indigenous people from across Turtle Island and we are grateful to have the opportunity to work on this land.

Privacy statement for recordings

(tentative) This course, including your participation, will be recorded on video and will be available to students in the course for viewing remotely and after each session. Course videos and materials belong to your instructor, the University, and/or other source depending on the specific facts of each situation, and are protected by copyright. In this course, you are permitted to download session videos and materials for your own academic use, but you should not copy, share, or use them for any other purpose without the explicit permission of the instructor. For questions about recording and use of videos in which you appear, please contact Prof. Khalil.

Statement on Inclusivity

You belong here. The University of Toronto commits to all students, faculty and staff that you can learn, work and create in a welcoming, respectful and inclusive environment. In this class, we embrace the broadest range of people and encourage their diverse perspectives. This team environment is how we will innovate and improve our collective academic success. You can read the evidence for this approach here.

We expect each of us to take responsibility for the impact that our language, actions and interactions have on others. Engineering denounces discrimination, harassment and unwelcoming behaviour in all its forms. You have rights under the Ontario Human Rights Code. If you experience or witness any form of harassment or discrimination, including but not limited to, acts of racism, sexism, Islamophobia, anti-Semitism, homophobia, transphobia, ableism and ageism, please tell someone so we can intervene. Engineering takes these reports extremely seriously. You can talk to anyone you feel comfortable approaching, including your professor or TA, an academic advisor, our Assistant Dean, Diversity, Inclusion and Professionalism, the Engineering Equity Diversity & Inclusion Action Group, any staff member or a U of T Equity Office.

You are not alone. Here you can find a list of clubs and groups that support people who identify in many diverse ways. Working together, we can all achieve our full potential.

Statement on Accommodations

The University of Toronto supports accommodations for students with diverse learning needs, which may be associated with mental health conditions, learning disabilities, autism spectrum, ADHD, mobility impairments, functional/fine motor impairments, concussion or head injury, blindness and low vision, chronic health conditions, addictions, deafness and hearing loss, communication disorders and/or temporary disabilities, such as fractures and severe sprains, or recovery from an operation.

If you have a learning need requiring an accommodation the University of Toronto recommends that students register as soon as possible with Accessibility Services at this link, 416-978-8060 or accessibility.services@utoronto.ca.

Statement on Mental Health

As a university student, you may experience a range of health and/or mental health challenges that could result in significant barriers to achieving your personal and academic goals. Please note, the University of Toronto and the Faculty of Applied Science & Engineering offer a wide range of free and confidential services that could assist you during these times.

As a U of T Engineering student, you have an Academic Advisor (undergraduate students) or a Graduate Administrator (graduate students) who can support you by advising on personal matters that impact your academics. Other resources that you may find helpful are listed on the U of T Engineering Mental Health & Wellness webpage, and a small selection are also included here:

- Accessibility Services & the On-Location Advisor
- Health & Wellness and the On-Location Health & Wellness Engineering Counsellor
- Inclusion & Transition Advisor
- U of T Engineering Learning Strategist and Academic Success
- My Student Support Program (MySSP)
- Registrar's Office
- SKULE Mental Wellness
- Scholarships & Financial Aid Office & Advisor

  If you find yourself feeling distressed and in need of more immediate support resources, consider reaching out to the counsellors at My Student Support Program (MySSP) or visiting the Feeling Distressed webpage.