BSc project proposals

in

Algorithms, Data Structures and Foundations of Machine Learning

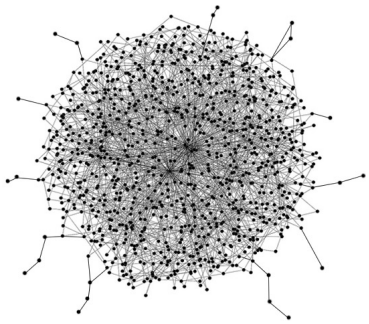
Peyman Afshani
Gerth Stølting Brodal
Kasper Green Larsen
Andrea Paudice
Chris Schwiegelshohn

{peyman, gerth, larsen, apaudice, schwiegelshohn}@cs.au.dk
Nygaard 3^{rd} floor

The following pages contain potential topics for BSc projects, but other projects are also possible. The final topic and direction of the project is settled under guidance by the advisor. In particular the balancing between theory and practical implementations is done on an individual basis, and is often adjusted during the BSc project process. Please do not hesitate to pass by our offices on or send us an e-mail for setting up a meeting to discuss potential BSc projects.

Connectivity in Graph Sketching

Problem *Let $G(V, E)$ be a graph over n nodes, where n is known. The graph is distributed, meaning that every node only knows its own neighbors.*
We assume that there exists a coordinator C. Every node can send a single message to C. After sending this message, C has to decide if the graph is connected or not.
The objective is to find the smallest message size such that C outputs the correct answer with probability at least $2/3$.



Distributed computing is a popular way to model problems arising in communication networks and large data sets. If a graph G is distributed, we typically imagine every node as being its own computing unit. The node only has its own local information, i.e. it knows its own neighborhood but not, for example, if two of its neighbors are also neighbors.

The nodes exchange messages (to each other or to a coordinator) with the intent of solving a problem. The efficiency of such a messaging protocol is measured by bounding the size of the messages in terms of the number bits, as well as the number of communication rounds. A special case of particular interest is to allow only a *single* communication round.

A graph is connected if there exists a path from every node to every other node. We want to understand how large the messages have to be for a coordinator to decide whether the graph is connected, while using only a single round of communication. Intuitively, it sounds like this problem should require message sizes of $\Omega(n)$ as there may exist a single edge (u, v) which determines connectivity, but neither u or v have any way of knowing this when constructing the message and thus will have to send their entire neighborhood. However, it turns out, very surprisingly, that $O(\log^3 n)$ bits per message is enough!

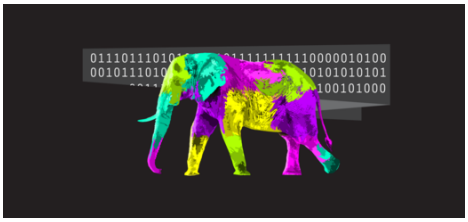
The aim of the project is to understand these messaging protocols, make a clear exposition the proofs, and investigate different message size/communication rounds trade offs.

References

- *Analyzing graph structure via linear measurements.* Kook Jin Ahn, Sudipto Guha, and Andrew McGregor. Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, Kyoto, Japan, January 17-19, 2012, pages 459–467.
  DOI: 10.1137/1.9781611973099.40.

Contact: Chris Schwiegelshohn

Tracking Frequent Items in Data Streams



Simons Institute

Finding frequently occurring items in a dataset is a common task in data mining. In the data streaming literature, this problem is typically referred to as the heavy-hitters problem, which is as follows: a huge stream of items is processed from one end to the other. Each item in the stream can be thought of as an integer and the goal is to report the integers occurring most often while using limited memory. An integer in the stream may represent e.g. a source IP address of a TCP packet. One then may want to find sources with high utilization in a network traffic monitoring application. Another example has each integer representing how many times a concrete web surfer clicked a web ad on a particular site. The goal then becomes to identify the frequent ad clickers in web advertising. Common to these applications is that one wants a solution with tiny memory consumption and processing time.

The aim of the project is to work with both real-life and synthetic data streams, and to implement, experiment with and compare different solutions to the heavy-hitters problem. An important aspect of several of the solutions to the heavy-hitters problem is that they rely on randomization. Thus a central task is to use basic probability theory to give a theoretical analysis of the solutions. The final report should include a theoretical section covering the basic analysis of different algorithms for the problem, a summary of the implemented algorithms, an experimental evaluation of the algorithms and a discussion of the obtained results.

References

- *An Improved Data Stream Summary: The Count-min Sketch and its Applications.* Graham Cormode, S. Muthukrishnan. Journal of Algorithms, 55(1), 58-75, 2005. DOI: 10.1016/j.jalgor.2003.12.001.

- *Finding Repeated Elements.* J. Misra, David Gries. Science of Computer Programming, 2(2), 143-152, 143-152, 1982. DOI: 10.1016/0167-6423(82)90012-0.

Contact: Chris Schwiegelshohn

Closest Points

Problem *Let P be a set of n points in the $\mathbb{R}^d$, given by their coordinates. The problem is to find two points $p, q \in P$ such that the Euclidean distance between them is the smallest.*

This is a classical computational geometry problem. Clearly, we can find the closest pair by going through all the pairs of points, compute the distance between every pair and then find the minimum among them. Unfortunately, this is a rather slow algorithm which runs in $\Theta(n^2)$ time.

The first elegant solution came by Shamos and later published together with Hoey in 1975. Their algorithm was a simple divide and conquer algorithm: partition the point set into two equal sizes, P_ℓ and P_r, e.g., with a vertical line h and then find the closest pair in each subset recursively. Let d be the smallest distance that we find in this way. If the closest pair is completely inside one subset, we are done. Otherwise, one point from P_ℓ and another point from P_r must be the closest points. In particular, we only need to consider the points within distance d of the line h. Now, they observed that only a $O(n)$ pairs need to be considered as each point in P_ℓ or P_r cannot have too many points of distance d next to them. This gives the recursion

$$f(n) = 2f(n/2) + O(n)$$

which solves to $f(n) = O(n \log n)$.

This solution generalizes to a point set in 3D in a more or less straightforward way and it gives an algorithm with $O(n \log^2 n)$ running time. The running time comes from a recursion

$$g(n) = 2g(n/2) + f(n)$$

where $f(n)$ is the time it takes to solve a 2D closest pair problem. If we go with $f(n) = O(n \log n)$, then we get $g(n) = O(n \log^2 n)$. However, a brilliant solution by Shamos and Bentley showed that this can actually be improved to $O(n \log n)$.

The goal of this project is to follow these developments. The tasks are: (1) Implement the $O(n \log n)$ algorithm for finding the closest pair in a set of n points in the plane. (2) Implement an $O(n \log^2 n)$ algorithm for finding the closest pair in a set of n points in 3D. (3) Implement the algorithm by Bentley and Shamos that runs in $O(n \log n)$ time to find the closest pair in a set of n points in 3D.

References

- *Divide-and-Conquer in Multidimensional Space.* Jon Louis Bentley and Michael Ian Shamos. 8th Annual ACM Symposium on Theory of Computing (STOC), 220-230, 1976. DOI: 10.1145/800113.803652.
- *Closest-Point Problems.* Michael Ian Shamos and Dan Hoey. 16th Annual Symposium on Foundations of Computer Science (FOCS), 151-162, 1975. DOI: 10.1109/SFCS.1975.8.

Contact: Peyman Afshani

Smallest Enclosing Ball

Problem *Let P be a set of n points in d-dimensions, given by their coordinates. The problem is to find the smallest ball that contains all the points. Thus, the output is the center of the ball and its radius.*

This is one of the classical problems in clustering. If we assume d is a constant, then the problem can be solved in linear time but often the dependencies are exponential in d. In particular, the best algorithm has the running time of $O(d^2n + 2^{O(\sqrt{\log n})})$ which is exponential in d. As a result, when the dimension is large, settling for approximations is a reasonable choice.

In this project, you will implement and compare two simple solutions.

1. This algorithm achieves a $(1+\varepsilon)$ factor approximation in time $O(nd/\varepsilon + 1/\varepsilon^5)$ (Bâdoiu and Clarkson, 2003).

2. However, there is also a simpler algorithm that achieves a 3/2 factor approximation in linear time, in fact, using only one scan of the input (Zarrabi-Zadeh and Chan, 2006).

The goal of this project is to implement these solutions and to compare them.

Task one. Implement the algorithms. Run some basic tests to make sure that they are correct.

Task two. Run some experiments to measure the running time, and the quality of the solution. Note that one algorithm should be fast but less accurate while the other one should be the opposite.

Task three. In your report, explain briefly how you implemented the algorithms and list any difficulties in doing so. You also need to briefly explain how the algorithms work but you don't have to cover the proof of correctness parts. Were the descriptions of the algorithms sufficient for the implementation or did you have to make some non-obvious choices? Describe how you designed your test cases and how you generated the input. Then, try to justify the results of the experiments.

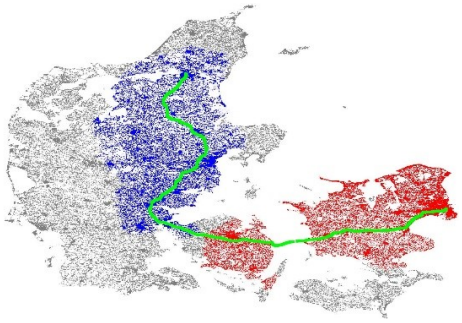
References

- *Smaller Core-sets for Balls.* Mihai Bâdoiu and Kenneth L. Clarkson. Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), 801–802, 2003. https://dl.acm.org/citation.cfm?id=644240

- *A Simple Streaming Algorithm for Minimum Enclosing Balls.* Hamid Zarrabi-Zadeh and Timothy Chan. Proc. 18th Canadian Conference on Computational Geometry (CCCG), August, 2006.
  http://www.cs.queensu.ca/cccg/papers/cccg36.pdf

Contact: Peyman Afshani

Shortest Paths Computations on Open Street Map Data



Martin Jacobsen

Open Street Map data contains a very detailed description of road networks worldwide and is the underlying data use in many map applications. The data is annotated with both geometric features and logically information of e.g. connection of road segments, speed limits, road types, one-way streets e.t.c.

The aim of this project is to be able to work with open street map data, in particular to visualize open street map data, convert open street map data to graph representations, implement algorithms for finding shortest paths in road network graphs, and to evaluate experimentally the performance of various algorithms. An important part of the process will be to read the theory behind selected shortest path algorithm, e.g. Dijkstra's algorithm, bidirectional shortest path algorithms, A*, landmark based algorithms, contraction hierarchies, highway hierarchies, hub labelling, and transit node algorithms. The final report should include a summary of the implemented theory, a description of the implementation, an experimental evaluation of the implemented algorithms, and a discussion of the obtained results — in particular a discussion of space usage versus query time.

The Open Street Map data can also be combined with other data sources, like the Danish height elevation model, to take elevation differences into account for e.g. finding optimal bicycle routes.

References

- *Route Planning in Transportation Networks.* Hannah Bast, Daniel Delling, Andrew Goldberg, Matthias Müller-Hannemann, Thomas Pajor, Peter Sanders, Dorothea Wagner, Renato F. Werneck. Algorithm Engineering 2016: 19-80. DOI: 10.1007/978-3-319-49487-6_2.
- Videos with Andrew Goldberg on the topic: Beyond Worst Case Analysis Workshop 2011, NWU 2013.
- Slides from the Algorithm Engineering course, 2017: route.pdf.

Contact: Gerth Stølting Brodal.

Regularized Linear Regression

Problem: We consider the problem of computing approximate solutions to the following

$$\underset{x\in\mathbb{R}^d}{\text{minimize}} \;\; f(x) = \frac{1}{2n}\sum_{i=1}^{n}(\langle x, z_i\rangle - y_i)^2 + \lambda\cdot\frac{\|x\|_2^2}{2}, \tag{1}$$

where $\lambda > 0$, $(z_i, y_i) \in \mathbb{R}^{d+1}$ for each $i \in [n]$, and $\langle\cdot,\cdot\rangle$ denotes the standard inner product in $\mathbb{R}^d$.

Description: The above is a fundamental problem in machine learning with many practical applications to estimation and data analysis including, but not limited to, house and stock pricing, and biological data analysis. Its exact solution can be computed with $\mathcal{O}(d^3 + nd^2)$ operations, which is not feasible when d or n is large. An alternative is represented by the *Gradient Descent* (GD) algorithm that computes an ε-approximate solution in $\mathcal{O}(dn\log(1/\varepsilon))$ time. For very large n, a faster algorithm is *Stochastic Gradient Descent* (SGD), which requires $\mathcal{O}(d/\varepsilon)$ operations to compute an ϵ-approximate solution, but only in-expectation. Finally, the *Stochastic Averaged Gradient Accelerated* method (SAGA) computes an ε-approximate solution (in-expectation) with $\mathcal{O}(n + d\log(1/\varepsilon))$ operations. Depending on the value of n, SAGA may be faster than SGD, although its implementation is slightly more involved.

The aim of this project is to implement these solutions and compare them on a synthetic or a real-world benchmark. In particular, the following tasks should be completed: implement and test the exact method; implement and test GD and SGD; implement and test SAGA; compare the algorithms in terms of runtime and accuracy of the computed solutions as n and d vary.

References

- *The method of steepest descent for non-linear minimization problems.* H.B. Curry. Quarterly of Applied Mathematics, Vol. 2, No. 3, pp. 258–261, 1944. GD original paper, available here.
- *A Stochastic Approximation Method.* H. Robbins, S. Monro. The Annals of Mathematical Statistics, Vol. 22, No. 3, pp. 400-407, 1951. SGD original paper, available here.
- *Understanding Machine Learning: From Theory to Algorithms.* S. Ben David, S. Shalev-Schwartz. Chapters 13-14. Cambridge University Press. 2014. Modern material on GD and SGD, available here.
- *SAGA: A Fast Incremental Gradient Method With Support for Non-Strongly Convex Composite Objectives.* A. Defazio, F. Bach, S. Lacoste-Julien. NIPS, 2014. SAGA original paper, available here.

Contact: Andrea Paudice

Bachelor Projects 2026

Data Intensive Systems Group

1. Red^2: Redundancy Reduction in Graphs

Advisor: *Davide Mottin (davide@cs.au.dk), Maximilian Egger (maximilian.egger@cs.au.dk)*

Redundancy in data can cause inconsistencies, especially during updates. For example, if your name appears twice in a dataset, both entries must be updated when you change it. Graphs, which represent relationships such as friendships in social networks, often contain even more redundancy than traditional databases.

In this project, we aim to reduce redundancy in graph data by enforcing dependencies between attributes [1], building on our recent research [unpublished, provided if interested]. Students can choose between two directions:

Projects:

- Develop an efficient algorithm to discover data dependencies [1], inspired by classic approaches [2].
- Evaluate our redundancy reduction techniques on applications like Graph Neural Networks [3].

Tasks:

- Study existing dependency discovery algorithms
- Apply our redundancy reduction framework (link available on request)
- Implement an algorithm for one of the two projects
- Run experiments and analyze the results

References:

[1] Fan, Wenfei, Yinghui Wu, and Jingbo Xu. *Functional dependencies for graphs.* SIGMOD 2016.
[2] Papenbrock, Thorsten, et al. *Functional dependency discovery: An experimental evaluation of seven algorithms.* PVLDB 8.10 (2015).
[3] https://distill.pub/2021/gnn-intro/

2. CriticBench Evaluation with New Iterative Methods

Advisor: *Ankita Maity (ankita.maity@cs.au.dk), Akhil Arora (akhil.arora@cs.au.dk)*

Problem: CriticBench [1] is a benchmark for testing LLMs' ability to generate answers, **critique** them, and make **corrections** across various reasoning domains. It covers five domains (math, commonsense, symbolic, coding, algorithmic) and measures generation, critique, and correction performance. This project empirically investigates **iterative reasoning** methods (like self-feedback loops) within CriticBench. We will add iterative methods (e.g. self-refinement) as new "models" in the CriticBench evaluation table. It will clarify how multiple rounds of feedback affect critique vs. correction performance.

Tasks:

- Reproduce a baseline CriticBench evaluation: run a selected LLM (e.g. GPT-4 or an open model) on CriticBench tasks in the standard G/Q/C (generation, critique, correction) pipeline.
- Implement iterative reasoning strategies: for example, use the Self-Refine approach [2] where the model generates an answer, critiques it, refines it, and possibly repeats for several rounds. Another strategy could be to chain multiple critique-correction cycles.
- Evaluate these methods on CriticBench: for each question, allow the model to do 2–3 critique/correction loops instead of just one. Measure the final answer accuracy and critique F1.
- Does this help against baseline methods (Table 1 of the CriticBench paper)?
- Analyze results by task domain: identify which kinds of reasoning problems benefit most from iteration (e.g., do algorithmic vs. commonsense tasks improve differently?).

Expected Outcome:

- Empirical results showing whether iterative self-feedback improves correction accuracy over single-step correction. For example, we may find that multiple critique rounds significantly boost performance on math tasks.
- Reports for baseline vs. iterative-method scores (for generation, critique, and correction).
- Clear documentation of methodology, including prompts used for iteration, and error analysis showing how answers evolve.

Required Skills:

- Familiarity with Python and calling LLMs (via API or open-source models).
- Understanding of prompting and model evaluation.

References:

- Lin et al. (2024), "CriticBench" – introduces the GQC benchmark and shows how different models perform on critique/correction tasks [1].
- Madaan et al. (2023), "Self-Refine: Iterative Refinement with Self-Feedback" – describes a method where an LLM repeatedly critiques and refines its own answers [2].

3. GAG: Graph Algorithms marry GPUs

Advisor: *Davide Mottin (davide@cs.au.dk), Ira Assent (ira@cs.au.dk), Cheng Huang (cheng@cs.au.dk),*

Graphs are expressive representations of connections among objects. Yet, when it comes to run complex graph algorithms such as finding communities [1], speed becomes a problem. For this reason, many people proposed parallel solutions for graph algorithms on multi-core CPUs and GPUs.

However, GPUs have limited memory, shared instructions, and other hardware limitations that hinder their applicability to graphs, where nodes have a large variety of neighbors. As such, we need to redesign the algorithms to work on GPUs.

This project explores the uncharted possibility of parallelizing and making graph algorithms more efficient. As an example, in a previous project we have successfully parallelized for GPUs D-core decomposition on dynamic [2] graphs, with results 100 times faster than the original algorithm using some trick on GPUs [3].

If you are interested in low-level optimizations, this is the project for you!

Requirements: Interest in programming at low level and learn a number of tricks that apply to GPUs.

Tasks:

- Study and run simple programs on GPUs
- Reimplement a simple algorithm in a naïve manner for GPUs
- Optimize the algorithm to beat the fastest parallel algorithm
- Make experiments and report

References:

[1] Li et al. A comprehensive review of community detection in graphs. Neurocomputing, 600, 2024.
[2] Liao, Xuankun, et al. Distributed d-core decomposition over large directed graphs. VLDB, 2022.
[3] https://www.cherryservers.com/blog/introduction-to-gpu-programming-with-cuda-and-python#:~:text=%23What%20is%20GPU%20Programming%3F,(GPGPU%20computing)%20as%20well.

4. CacheSaver: Efficient, Cost-Effective Experimentation with Large Language Models

Advisor: Nearchos Potamitis (nearchos.potamitis@cs.au.dk), *Akhil Arora* (akhil.arora@cs.au.dk)

CacheSaver is a high-efficiency caching tool for experiments using large language models (LLMs), designed to reduce costs, ensure reproducibility, and streamline debugging. By caching multiple responses per query and tracking usage, CacheSaver enables unique sampling across experiments while preserving full statistical integrity. Built on top of Python's `asyncio`, it supports asynchronous execution, batching, prompt deduplication, and race-condition prevention.

We intend to hire **at least three groups** for this project. Each group will focus on **developing and evaluating one novel LLM-based application** using CacheSaver as the backbone for efficient experimentation.

Tasks:

- Familiarize yourself with CacheSaver's code [1], features, and mechanics.
- Select and study one LLM application area (described below). Each group will work on one application.
- Design and implement an experimental setup for the chosen application using CacheSaver for caching, logging, and reproducibility.
- Conduct systematic experiments and ablation studies to demonstrate the efficiency and utility of CacheSaver in the selected application.
- (Optional but ideal) Repeat experiments using multiple LLMs (e.g., LLaMa, GPT-4, Claude) to assess generalizability.

Applications:

1. LLM as a Judge: Large language models can serve as evaluators or "judges" for generated text—rating reasoning quality, factuality, or style. This project will explore how to build and benchmark automated judging pipelines using CacheSaver, ensuring efficient reuse of evaluation calls and consistent scoring across models.

2. Retrieval-Augmented Generation (RAG): RAG systems enhance LLMs by grounding their responses in retrieved external knowledge. In this project, students will implement RAG pipelines with CacheSaver to cache retrievals, generated outputs, and prompts—reducing costs while maintaining experimental reproducibility and scalability.

3. Multi-Agent Systems: Multi-agent setups involve several LLMs collaborating or debating to solve complex problems. This project will leverage CacheSaver to efficiently coordinate and log

multi-agent interactions, enabling scalable experimentation and controlled studies of communication, planning, and emergent reasoning.

Expected outcome:

- Clean code and documentation of the chosen application built on top of CacheSaver.
- Successful integrations of the new application into our CacheSaver [1] framework.
- Experimental reports highlighting cost savings, reproducibility, and efficiency gains enabled by CacheSaver.

Required skills:

- Strong programming skills and knowledge of Python, multi-threading, async execution.
- A creative mind with an inquisitive attitude.
- Experience with LLMs and prompting is a plus.

Related work:

[1] CacheSaver Github repository
[2] Fleet of Agents: Coordinated Problem Solving with Large Language Models using Genetic Particle Filtering
[3] Tree of Thoughts: Deliberate Problem Solving with Large Language Models
[4] WebShop: Towards Scalable Real-World Web Interaction with Grounded Language Agents

5. Explainable AI

Advisor: *Ira Assent (ira@cs.au.dk)*

The primary objective of this project is to develop and evaluate algorithms that enhance the explainability of machine learning or data mining models. As these models become increasingly complex, it is crucial to make their predictions interpretable to non-technical users, domain experts, and regulators to build trust and ensure ethical usage. This project aims to assess algorithms that identify influential features, visualize decision boundaries, and quantify the reliability of model predictions.

Tasks:

- Literature Review of current algorithms and methods in explainable AI, with a focus on evaluation metrics
- Develop/implement algorithms for explainability, such as feature attribution methods (e.g., SHAP [2], LIME[1]) or methods for generating counterfactuals. Possibly design new variations or optimizations
- Evaluate on standard datasets (e.g., MNIST, CIFAR-10) using quantitative metrics, such as fidelity or stability, possibly including new metrics or evaluation approaches
- Document algorithms, metrics, and evaluation process, including insights into how to improve interpretability and a discussion on metric effectiveness.

Expected outcome:

- An implementation of one or more explainability algorithms models or
- A quantitative assessment based on metrics like fidelity, or stability or
- A comparative analysis of any developed methods against existing ones or
- A discussion of notions of explainability and its metrics, with suggestions for future improvements

References:

1. Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). "Why Should I Trust You?": Explaining the Predictions of Any Classifier." Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.
2. Lundberg, S. M., & Lee, S.-I. (2017). "A Unified Approach to Interpreting Model Predictions." Advances in Neural Information Processing Systems (NeurIPS).
3. Vilone, Giulia, and Luca Longo. "Notions of explainability and evaluation approaches for explainable artificial intelligence." Information Fusion 76 (2021): 89-106.
4. Nauta, M., et al. (2023). From anecdotal evidence to quantitative evaluation methods: A systematic review on evaluating explainable ai. ACM Computing Surveys, 55(13s), 1-42.

6. BeG:A Benchmark for Graph Correspondences

Advisor: *Davide Mottin* (*davide@cs.au.dk*)

Graphs are structures where points (nodes) are connected through relationships. For example, a social network links people through friendships, and a protein network connects molecules that interact. Many real-world problems require finding which nodes in two graphs correspond to each other — a task known as *subgraph matching*, *graph matching*, or *graph alignment*, depending on the goal and scale.

This project extends an existing benchmark and library for evaluating state-of-the-art graph alignment algorithms across multiple datasets, with applications in biology, social networks, and recommendation systems. Our initial results are presented in [1].

Tasks:

- Study related work and our previous solutions
- Implement missing algorithms from the literature
- Integrate additional datasets
- Run experiments and analyze results

References:

[1] Experimental evaluation of graph alignment algorithms

7. Can Bandits Count?

Advisor: *Kasper Overgaard Mortensen (km@cs.au.dk), Davide Mottin (davide@cs.au.dk)*

Bandits [3] are predecessors of Reinforcement Learning (RL) in which we have a number of choices each associated with a hidden reward. By playing a game and making our choices we aim to maximize the overall reward. The advantage of bandits with respect to RL is their simplicity and their applicability to a wide range of applications.

For instance, in data storages and databases, a crucial problem is to estimate how many answers a query has without executing it. This problem is called cardinality estimation [1]. Yet, if queries vary a lot, the current methods struggle [2].

In this project we aim to combine Bandits and cardinality estimation when queries are non-stationary, i.e. they vary a lot, by experimenting with some variations of Bandits.

Tasks include:

- Study the related work and get familiar with Multi-armed bandits
- Run previous methods for cardinality estimation
- Generate an adversarial workload
- Implement bandits for cardinality estimation
- Run experiments on available data and queries

Related work:

[1] Estimating Cardinalities with Deep Sketches
[2] Robust Query Driven Cardinality Estimation under Changing Workloads
[3] Introduction to Multi-armed bandits

8. Robust Multilingual News Collection Pipelines

Advisor: *Akhil Arora* (akhil.arora@cs.au.dk)

Most large-scale quotation corpora and related datasets rely on English-language news and are no longer updated. For example, *Quotebank* contains ~178 million quotations from 162 million English news articles (2008–2020) [1], but data collection stopped in April 2020. There are currently no openly available large-scale news datasets covering recent years or multiple languages. Meanwhile, projects such as *QuoteKG* (~1M quotes in 55 languages from Wikiquote) [2] and *Media Cloud* (covering ~20 languages for global news analytics) [3] illustrate the growing importance and feasibility of multilingual data.

This project aims to build and evaluate a robust pipeline for collecting and organizing recent news articles from the web, filling the temporal gap after April 2020 and expanding beyond English coverage. The focus is on building scalable, well-logged, and reproducible news collection pipelines, which could later serve as input for downstream tasks such as quotation extraction.

Tasks:

- Identify and integrate suitable news sources or APIs for multilingual and post-2020 data (e.g., using *Media Cloud* [3] or direct RSS/news APIs).
- Develop a modular Python pipeline to collect, store, and log news articles, including metadata (publication date, source, URL, etc.).
- Run and evaluate the pipeline to collect a multi-month dataset, reporting article counts and coverage statistics across selected languages.
- Document all design choices, methods, and challenges to ensure the system is reproducible and extensible.
- *(Optional stretch goal)*: Implement or adapt a quotation-extraction module to identify and attribute direct quotes from the collected articles.

Group distribution: Multiple student groups may work on **different target languages or regions**, enabling comparative evaluation and ensuring broad linguistic coverage (e.g., Danish, German, French, Spanish, Hindi, etc.).

Expected outcome:

- A working, multilingual news collection pipeline covering post-2020 articles.
- A structured dataset of news articles with metadata, suitable for downstream NLP research.
- Clear documentation of design, setup, and lessons learned to support long-term maintenance and extension.

Required skills:

- Solid Python programming ability.
- Familiarity with web scraping, API integration, and data processing (JSON, CSV).
- Interest in multilingual data and willingness to handle real-world text variability.
- Experience with LLMs and prompting is a plus.

Related work:

[1] Vaucher et al., Quotebank: A Corpus of Quotations from a Decade of News, WSDM'21
[2] Kuculo et al., QuoteKG: A Multilingual Knowledge Graph of Quotes, ESWC'22
[3] Media Cloud (2023) – Media analytics platform (supports ~20 languages for news content collection).

9. Dynamic Query-Aware PathRank Summarization

Advisor: *Ama Bembua Bainson (ama@cs.au.dk) , Davide Mottin (davide@cs.au.dk)*

Knowledge Graphs (KGs) represent facts as entities connected by relationships, enabling reasoning over rich, structured information. However, answering queries efficiently over large KGs can be slow. *PathRank* is a semantic ranking algorithm that identifies the most relevant nodes and paths based on how often they co-occur and interact in the graph.

This project investigates how to make PathRank [1] *online* and *adaptive* — able to update semantic rankings as new queries arrive. Instead of recomputing everything from scratch, the goal is to design an incremental, query-aware PathRank that efficiently updates scores as the query distribution evolves. The resulting dynamic semantic summary can accelerate query answering while focusing on the most relevant parts of the KG.

Interests: Graph algorithms, online learning, or efficient data processing

Requirements: Python programming and basic knowledge of graph algorithms.

Tasks:

- Study and implement the basic PathRank algorithm.
- Generate and simulate a stream of evolving queries over a sample KG (e.g., WikiData or DBpedia).
- Design an online update strategy for dynamic PathRank scores.
- Evaluate the speed and accuracy of the dynamic version against static PathRank

References

[1] Lee, Sangkeun, et al. "Pathrank: A novel node ranking measure on a heterogeneous graph for recommender systems." Proceedings of the 21st ACM international conference on Information and knowledge management. 2012.

BSc project proposals

in

Computational Complexity and Game Theory

Ioannis Caragiannis
Kristoffer Arnsfelt Hansen
Stratis Skoulakis
{iannis, arnsfelt, stratis}@cs.au.dk
Incuba 4^{th} floor

The following pages contain potential topics for BSc projects, but other projects are also possible. The final topic and direction of the project is settled under guidance by the advisor. In particular the balance between theory and practical implementations is done on an individual basis, and is often adjusted during the BSc project process. Please do not hesitate to pass by our offices on or send us an e-mail for setting up a meeting to discuss potential BSc projects.

Linear Complementarity Problems

A linear complementarity problem (LCP) in n variables is given by an $n \times n$ matrix M and a vector $q \in \mathbb{R}^n$ and the task is simply to find a non-negative vector $z \in \mathbb{R}^n$ such that the vector $Mz + q$ is both non-negative and orthogonal to z.

LCPs generalize Linear Programming (LP) problems (a major topic in the course Optimization). In fact, the larger class of convex quadratic programs can be formulated as LCPs. These are both classes of problems that can be solved in polynomial time with interior point algorithms. Several other algorithms exist for LCPs, for instance Lemke's algorithm that is closely related to the Simplex algorithm for Linear Programming.

For other classes of LCPs we have no known polynomial time algorithms. The problem of computing a Nash equilibrium (NE) in 2-player (general sum) games can also be formulated as LCPs. While the task of computing a NE is generally viewed as a computationally hard problem (hard for a complexity class called PPAD), Lemke's algorithm will often do well in practice for concrete games. Several other important problems from algorithmic game theory can likewise be formulated as LCPs and Lemke's algorithm or a variant of Lemke's algorithm can be used to solve these. Moving to general LCPs, it turns out that solving them is actually an NP-hard problem!

The project would typically be a mix of theory and practice, involving the analysis of algorithms and experimentation with concrete implementations and problems.

References

- Cottle, Pang, and Stone: The Linear Complementarity Problem, 2009. DOI: 10.1137/1.9780898719000.
- Murty: Linear Complementarity, Linear and Nonlinear Programming, 1997. internet edition.

Contact: Kristoffer Arnsfelt Hansen

Fair Division Algorithms for Indivisible Items

Fair division is the task of allocating items to agents so that some fairness criterion is satisfied. The problem finds applications in every setting where some kind of allocation of resources is required. In the very popular variations with indivisible items, problem instances consist of a set of n agents and a collection of m items. Agents have valuations for the items: agent i has a valuation $v_i(j)$ for item j. Valuations for sets (or bundles) of items are then additive, with the valuation of agent i for the bundle of items S being $v_i(S) = \sum_{j \in S} v_i(j)$. An allocation of items to agents is simply a partition $A = (A_1, A_2, ..., A_n)$ of the items to the agents so that agent i gets the bundle of items A_i.

An important fairness criterion is envy-freeness. We say that an allocation A is envy-free if for every pair of agents i and j, it holds that $v_i(A_i) \geq v_i(A_j)$. In words, every agent is happy with his/her bundle of items and does not envy the bundle allocated to some other agent. It can be easily seen that envy-freeness is rarely feasible. Indeed, consider an instance with two agents, who have positive valuations for a single item. Then, any allocation that gives the item to one of the agents is not envy-free. For this reason, several relaxations of envy-freeness have been considered in the literature.

Among them, the notion of envy-freeness up to any item (or EFX) is the most appealing. An allocation A is EFX, if for every pair of agents i and j, it holds that $v_i(A_i) \geq v_i(A_j \setminus \{g\})$, for any item $g \in A_j$. In words, an allocation is EFX if the possible envy of an agent for another is eliminated by removing any item from the bundle of items allocated to the latter. Unfortunately, until now, it is not known whether all instances have EFX instances or not. Fortunately, there are polynomial-time algorithms to compute partial EFX allocations (i.e., by discarding some of the items), simultaneously providing additional efficiency guarantees (e.g., they make sure that all agents get a bundle of items, for which they have a high value). The objective of this project is to study the recent literature on fair division of indivisible items, with a particular focus on the proposed algorithms for computing partial EFX allocations. Part of the work will include the implementation of these algorithms (and of variations of them) and their testing on real and synthetic instances.

References

- *A Little Charity Guarantees Almost Envy-Freeness.* Bhaskar Ray Chaudhury, Telikepalli Kavitha, Kurt Mehlhorn, and Alkmini Sgouritsa. In *Proceedings of the 31st ACM-SIAM Annual Symposium on Discrete Algorithms (SODA)*, 2658–2672, 2020. DOI: 10.1137/1.9781611975994.162.
- *Envy-Freeness Up to Any Item with High Nash Welfare: The Virtue of Donating Items.* Ioannis Caragiannis, Nick Gravin, and Xin Huang. In *Proceedings of the 20th ACM Conference on Economics and Computation (EC)*, 527–545, 2019. DOI: 10.1145/3328526.3329574.
- *Spliddit: Unleashing Fair Division Algorithms.* Jonathan R. Goldman and Ariel D. Procaccia. SIGecom Exchanges, 13(2), 41–46, 2014. DOI: 10.1145/2728732.2728738.

Contact: Ioannis Caragiannis

Algorithmic Aspects of Participatory Budgeting

Participatory budgeting is a relatively recent trend used by municipalities and regional governments worldwide to decide the distribution of funding to public projects. The main idea is to let the citizens express their opinion through voting over spending priorities and then implement the outcome of the vote as a public decision.

The details of voting are rather complicated (compared to a typical government election) and include a series of steps and features. We briefly present three components that are interesting from an algorithmic viewpoint. The first is the selection of the available alternatives and their nature. For example, alternatives could be discrete such as the construction of a bridge or continuous such as the length of bicycle paths. Second, how are the actual preferences of the citizens will be structured in the ballots? This step may result in a loss of information. Third, how should the votes be aggregated into a collective decision? This part is the most crucial algorithmically but strongly depends and interplays with the previous two.

The interdisciplinary field of computational social choice (lying at the intersection of social choice theory and theoretical computer science), which traditionally studies the computational complexity of voting rules, has extensively considered participatory budgeting recently, together with other modern trends of participatory democracy. The focus has been on axiomatic properties, and on quantifying the lack of efficiency of the voting process, due to the loss of information when expressing preferences.

The aim of the project is to survey the recent approaches in participatory budgeting, focusing on theoretical developments and analysis, case studies from its application to municipalities around the world, and available online tools. The project will involve the implementation of representative aggregation methods and experiments/simulations with synthetic voting scenarios.

References

- Web: www.participatorybudgeting.org, pbstanford.org.
- *Participatory Budgeting: Models and Approaches.* Haris Aziz and Nisarg Shah. In *Pathways Between Social Science and Computational Social Science: Theories, Methods, and Interpretations*, Springer, 2021. Forthcoming (available from Nisarg Shah's homepage).
- *Interactive Democracy.* Markus Brill. In *Proceedings of the 17th International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, 1183–1187, 2018. dl.acm.org/doi/10.5555/3237383.3237873.

Contact: Ioannis Caragiannis

Financial Networks and Systemic Risk

The last major financial crisis and its aftermath have revealed the systemic risks and hazards for the society that can arise in financial markets. These are mainly due to the complicated structure of these markets, with many different financial institutions (e.g., banks or firms) that are highly interconnected and interact with each other.

Over the last decade, substantial research has been undertaken to analyse, understand, and manage systemic risks in financial markets. This originates from the seminal work of Eisenberg and Noe, who proposed a graph-theoretic model that is considered the standard today. Interconnections between financial institutions are represented by a directed graph $G = (V, E)$. The node set V contains the financial institutions. A weighted directed edge $e \in E$ expresses a debt relation between two institutions. In addition, each institution has non-negative external assets, which capture the value of property rights (such as real estate, gold, business and mortgage loans, etc.) that the firm has acquired from non-financial institutions. Eisenberg and Noe discuss a clearing mechanism for such a market in which every institution v uses its available assets to pay its debt.

This BSc thesis aims to survey recent work on the model of Eisenberg and Noe and its extensions. Two kinds of questions will be considered. First, we will study computational problems related to identifying risks on the graph representing the interactions between financial institutions. Second, we will study simplified strategic games played by the several institutions. The work will mainly focus on mathematical proofs and analysis of algorithms in the graph-theoretic model above, but it may include implementations and simulations of strategic play by the financial institutions and experiments on these dynamics.

References

- *Systemic Risk in Financial Systems.* Larry Eisenberg and Thomas Noe. Management Science, 47(2):236– 249, 2001. Available from: https://www.jstor.org/stable/2661572
- *Strategic Payments in Financial Networks.* Nils Bertschinger, Martin Hoefer, & Daniel Schmand. In *Proceedings of the 11th Innovations in Theoretical Computer Science Conference (ITCS)*, 46:1-16, 2018. Available from: https://drops.dagstuhl.de/opus/volltexte/2020/11731/
- *Forgiving Debt in Financial Network Games.* Panagiotis Kanellopoulos, Maria Kyropoulou, Hao Zhou. In *Proceedings of the 31st International Joint Conference on Artificial Intelligence (IJCAI)*, pages 335-341, 2022. Available from: https://www.ijcai.org/proceedings/2022/48

Contact: Kristoffer Arnsfelt Hansen

Reinforcement Learning in Prisoner's Dilemma

The Prisoner's Dilemma (PD) is a fundamental setting in game theory illustrating how two rational strategic agents may fail to cooperate, even when cooperation would yield a better outcome for both. In PD, two prisoners are separately offered a deal: betray the other for a reduced sentence, or stay silent. If both betray, they both receive moderate sentences; if both remain silent, they receive minimal sentences. However, if one betrays while the other stays silent, the betrayer goes free while the silent prisoner receives a harsh sentence. While mutual silence leads to the best collective outcome, betrayal is known to be the dominant strategy. The Prisoner's Dilemma captures various real-world situations, from pricing strategies in economics to behaviors in animal species, highlighting the tension between individual gain and mutual cooperation.

Classical game theory predicts that mutual cooperation is unlikely to arise in a single, one-shot version of the game. However, recent studies indicate that Machine Learning algorithms can exhibit cooperative behaviors when the game is played repeatedly over time. This project aims to explore whether Reinforcement Learning (RL) algorithms, such as Q-Learning, can foster cooperative behavior in the context of the Prisoner's Dilemma. The project will begin with an experimental phase, evaluating various RL algorithms, followed by a mathematical analysis based on experimental findings.

References

- *Artificial Intelligence, Algorithm Design, and Pricing.* Larry Eisenberg and Thomas Noe. AEA Papers and Proceedings, 452–56, 2022. Available from: https://www.aeaweb.org/articles?id=10.1257/pandp.20221059
- *Reinforcement learning in a prisoner's dilemma.* Arthur Dolgopolov. In *Games and Economic Behavior*, 144:84-103, 2024. Available from: https://www.sciencedirect.com/science/article/pii/S0899825624000058
- *Artificial Intelligence, Algorithmic Pricing, and Collusion.* Emilio Calvano, Giacomo Calzolari, Vincenzo Denicolò, Sergio Pastorello. In *American Economic Review*, 110: 3267-97, 2020. Available from: https://www.aeaweb.org/articles?id=10.1257/aer.20190623

Contact: Stratis Skoulakis

Online Learning and Assembly of Calculus

Online Learning (OL) is a crucial subfield of Machine Learning that addresses decision-making problems. Recently, Online Learning has gained significant importance in game theory, as OL algorithms provide formal optimality guarantees on an agent's payoff regardless of the behavior of other agents over time. Over the years, various OL algorithms have been proposed, and recent studies reveal that the behavior of human agents aligns closely with the behavior of these OL algorithms.

The goal of this project is to investigate whether Online Learning can explain the behavior of real human agents in competitive game-theoretic settings. A key challenge is that current OL algorithms rely on complex mathematical operations that do not align with human and animal brain. However, recent work introduces a computational model called Assembly Calculus, which models fundamental brain operations. In this project, we will explore classical OL settings (e.g., the Expert Problem, Multi-Armed Bandits etc.) and examine whether biological learning processes, such as Hebbian Plasticity, can inspire new OL algorithms compatible with Assembly Calculus. Ultimately, the goal is to develop novel OL algorithms that, while forgoing complex mathematical operations, are compatible with the fundamental operations of the brain.

References

- *Introduction to Online Convex Optimization.* Elad Hazan. Available from: https://arxiv.org/pdf/1909.05207
- *Assemblies of neurons learn to classify well-separated distributions.* Max Dabagia, Santosh S. Vempala, Christos H. Papadimitriou. In *Conference on Learning Theory*, 178:3685-3717, 2022. Available from: https://arxiv.org/abs/2110.03171
- *Random Projection in the Brain and Computation with Assemblies of Neurons.* Christos H. Papadimitriou, Santosh S. Vempala. In *10th Innovations in Theoretical Computer Science Conference*, 57:1-57:19, 2019. Available from: https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.ITCS.2019.57

Contact: Stratis Skoulakis