System and Device Programming Projects

Device Programming Part (Prof. Quer)

A.Y. 2021-2022

Projects will undergo the following rules and restrictions:

- Projects are optional and the exam is passed only if the written test (both parts) is passed.
- Each student can take a project only once in his curricula path. The project must be selected before taking any written examination tests. Each project must be selected by groups of 2 or (at most) 3 students.
- Projects will be assigned to group of students on a first-come-first-served basis.
- All projects taken in 2021-2022 must be completed and delivered before or at the latest during the examination session of February 2023. The beginning of a course in the A.Y. 2022-2023 will automatically erase all pending projects.
- The evaluation of each project will follow a short presentation given by the group of candidates during one examination session. To enroll for the presentation, the project must be uploaded on the portal web page before the written test of that examination section. The student must enroll for the exam in that session (even if it only discussing the project and not taking any written test).
- Marks may differ for different students within the same group, depending on the effort they put into the project and on the final oral presentation. Projects may amend the final mark adding to it a value ranging from -2.0 to +6.0 marks. Project taken (assigned by the instructors) and **not completed** by the group will be evaluated (for all students within the group) with a final value of **-2.0**. Once the final mark for a project has been assigned, there is no time limit to its validity.

Projects will be assigned with the following protocol:

- Phase 1 (selection phase)
    - By **Friday the 13st of May** each group of students willing to take a project must indicate two group works (first and second choice)
        - Before that deadline there will be the opportunity to discuss with the instructors characteristics and details of all projects either in real-time or off-line

- The choices must be stated by uploading on the portal, at the page "Elaborati", a **single** file with **name**, **content** and **description** equal to the following **unique** string

  rn1_rn2_rn3_p1_p2

  where
  - **rn1**, **rn2** and **rn3** are the register numbers of the 2 or (at most) 3 students forming the group. Register numbers must be reported in increasing order.
  - **p1** and **p2** are the works selected by the group (namely, first choice **p1**, and second choice **p2**), i.e., q1, q2, etc., for Quer's projects and c1, c2, etc. for the Cabodi's projects.

  Again, please notice that only one student for each group has to upload this information.
- Two correct examples of this file name, content, and comment to insert on the "Elaborati" web page are the following:
  - 134567_146789_189345_q1_c2 which means that the students of register numbers 134567, 146789 and 189345 selected the Quer's project number 1 as a first choice and the Cabodi's project number 2 as second choice.
  - 176789_191352_c3_q4 which means that the students of register numbers 176789 and 191352 selected the Cabodi's project number 3 as a first choice and the Quer's project number 4 as second choice.
- Notice that only **one student for each group** has to upload this file.

- Phase 2 (tentative assignment phase)
  - By **Friday the 20th of May** the instructors will publish on the portal web page a list of assignments, i.e., a list in which each group will be assigned a single work following a first-come-first-served algorithm and trying to maximize the student's preferences
- Phase 3 (final assignment phase)
  - By **Friday the 27th of May** each group will have the possibility to accept or reject the final assignment and eventually, **but only in very specific cases**, to select a different one:
    - Accept the project will be the default, no action will be necessary.
    - Reject (or change) the project will be possible during a further discussion (either on or off-line) with the instructors.

Project 1: The Path-Planning Algorithm A*

Project's summary

A* (pronounced "A-star") is the most famous "path search algorithm", and it is used in many fields of computer science going from path planning for non-autonomous and autonomous vehicles and robots to AI problems. This project requires the implementation and the comparison of a sequential and a parallel version of this algorithm on large and weighted benchmark graphs.

Problem Definition

Given a weighted, directed graph $G = (V, E)$, with a weight function $W: E \rightarrow R$ mapping edges to real-valued weights, the single-source shortest-paths problem finds a shortest path from a given source vertex $s \in V$ to each vertex $v \in V$.
The so-called Dijkstra's algorithm was conceived by Edsger W. Dijkstra in 1956 and it solves this problem in a greedy way.
A* constitutes an important improvement to Dijkstra's algorithm. A* starts from the source node $s \in V$ and finds a path to the given destination node $d \in V$ using a heuristic function to estimate the cost of the cheapest path to the goal. Practically speaking, the Dijkstra's algorithm can be seen as a special case of A*.

Following the indicated references, this project implies.

- Reading large benchmark graphs from the long-term memory.
- Implementing at least one sequential version of A*.
- Implementing at least one parallel version of A*.
- Comparing the sequential and the parallel version of the algorithm.

The comparison must consider the computation time and the memory usage of the main phases (at least, graph loading and path computation) and compare the sequential version with the parallel one with an increasing number of threads (i.e., 1, 2, 3, 4, etc.) on graphs of increasing size and complexity.

Required Background

Background aspects are all covered within the following courses:

- Advanced problem solving and programming classes (e.g., "Algorithm and Programming")

- "System and Device Programming" course, i.e., concurrent programming.

Working Environment

Students can work on their laptop or desktop. Each group can decide whether to develop the application under a Unix-like or a Windows system. The implementation can be done in C, C++, or C++ using the boost library (for GPU optimizations). Parallel algorithms are described on the papers reported in the reference section. These works are available in the reference directory.

Constraints

All projects must be delivered including the following material:

- The source C/C++ files.
- A README text file (written in plain ASCII) including a short "user manual", i.e., a document describing how to compile and run the program, under which system, which API, etc.
- A DOCUMENTATION text file (written in Word, Latex, Mark-down, etc.) including a short "designer manual", i.e., a document including:
    - All main design choices (how the reading part has been performed, how the data structure has been organized, how the parallelism has been designed, etc.)
    - The experimental evaluation of the tools, i.e., tables or graphics reporting a reasonable set of experimental results. The experimental evidence should include a comparison (in terms of memory and of elapsed time) between the original sequential version of the tool and the 2-3 parallel versions with different parallelization levels (i.e., with 1, 2, 4, 8, etc., threads) and different size of the input graph (up to millions of nodes).
- An OVERHEAD set (organized in PowerPoint or similar) to be used during the project discussion in the project evaluation phase.

Contact

Prof. Stefano Quer (stefano.quer@polito.it).

References

- For the Dijkstra's algorithm, please see T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein, "Introduction to Algorithms", McGraw-Hill
- https://en.wikipedia.org/wiki/A*_search_algorithm
- https://arxiv.org/pdf/1708.05296.pdf
- Other references are reported in the directory dedicated to this project.

Project 2: Implementation of alternation functions in C++ for detection and evaluation of bias in ML

Project's summary

Machine learning (ML) applications may systematically discriminate certain social groups (e.g., based on gender, ethnic groups, etc.) when performances are very different with respect to them (e.g., very different false positive rates for males and females). The problem is known as bias in ML. In this context, alternation functions are a novel technique for detecting and evaluating potential machine learning bias. The technique is computationally intensive and optimized management of resources with C++ is desirable. This project aims at implementing and testing the technique, reporting results both from the point of view of bias and the point of view of performances.

Problem Definition

The project is entirely based on the research article of Alelyani,S. "*Detection and Evaluation of Machine Learning Bias*", Appl.Sci.2021,11,6271; https://doi.org/10.3390/app11146271 .
The project consists of the following main steps:

1. Implement the alternation function in C++ and apply it to the dataset used in article.
2. Evaluate the bias in the ML model using the Kullback–Leibler (KL) divergence metric.
3. Report on the performances of both the alternation function and the KL computation.

Consider a dataset $D \in R^{n \times m}$,$D = \{d_1,d_2,...,d_i,...,d_n\}$, where d_i is the ith instance. D can be also represented as a set of attributes' vectors $\{a_1, a_2, \ldots, a_j, \ldots, a_m\}$. In the context of supervised learning, variable $y = \{y_1, y_2, \ldots, y_i, \ldots, y_n\}$ is the target, i.e. a vector of labels with size n. The predicted target, which is obtained from the learning process, is denoted as $\hat{y}$: therefore, $f(D) \rightarrow \hat{y}$ is a classification model that takes dataset D and assigns each data sample d_i to a specific target $\hat{y}i$. The process to evaluate bias in a specific attribute a_x (e.g., gender) is the following:

1. Train the model $f(\cdot)$ on the dataset D.
2. Predict the class label for each data point in $f(D) \rightarrow \hat{y}$.
3. Apply the alternation function on the a_x attribute $\varphi(D) \rightarrow D\varphi$.
4. Train the model $f(\cdot)$ on the alternative dataset $D\varphi$.
5. Predict the alternative predicted label $f(D\varphi) \rightarrow \hat{y}\varphi$.

6. Evaluate KL Divergence between the distributions of y and y^ φ for a_x each level of the attribute a_x (if cross fold validation is used, it should be done distinctively across all folds).

7. The difference computed with respect to the attribute a_x represents the bias.

Performances should be reported both for step 3 and for step 7.

Required Background

C++ aspects are covered within the "System and Device Programming" course. Basics knowledge of how ML works is also required. Further specific background concepts are contained in the reference research article.

Working Environment

Students can work on their laptop or desktop. Each group can decide whether to develop the application under a Unix-like or a Windows system. The implementation will be in C++.

Constraints

All projects must be delivered including the following material:

- The source C++ files.
- A README text file (written in plain ASCII) including a short "user manual", i.e., a document describing how to compile and run the program, under which system, etc.
- A DOCUMENTATION text file (written in Word, Latex, Mark-down, etc.) including a short "designer manual", i.e., a document including:
    - All main design choices (how the input reading part has been performed, how the data structure has been organized, etc.)
    - The experimental evaluation of the tools, i.e., tables or graphics reporting a reasonable set of experimental results.
- An OVERHEAD set (organized in PowerPoint or similar) to be used during the project discussion in the project evaluation phase.

Contact

Prof. Antonio Vetrò (antonio.vetro@polito.it).

References

https://doi.org/10.3390/app11146271