PONTIFICIA
UNIVERSIDAD
CATÓLICA
DE CHILE

INTRODUCCIÓN A LA PROGRAMACIÓN

PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE

Introducción a la Programación

- Temario de hoy
  - (Coordinación)
  - Introducción del profesor
  - Introducción al curso
  - Breve historia de la computación
  - Arquitectura de computadores
  - Introducción al software

- Bastante materia introductoria
  - Avanzaremos en las láminas hasta donde podamos
  - Retomamos en la siguiente clase

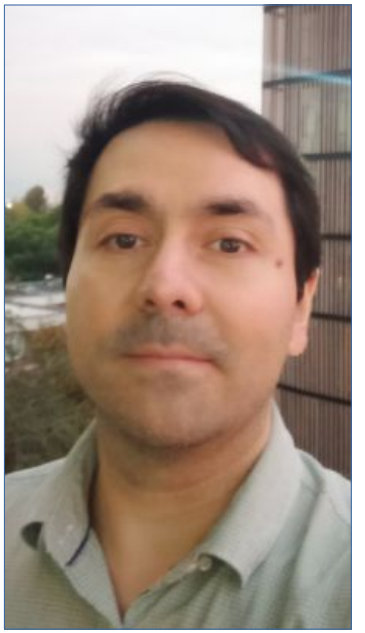
PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE

Cuerpo docente

- Coordinador: Guillermo García
  - guillermo.garcia@uc.cl
- Profesor de cátedra: Mauricio Monsalve
  - mauricio.monsalve@cigiden.cl
- Coordinador de ayudantes: Orlando Avendaño
  - sebaav@uc.cl

PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE

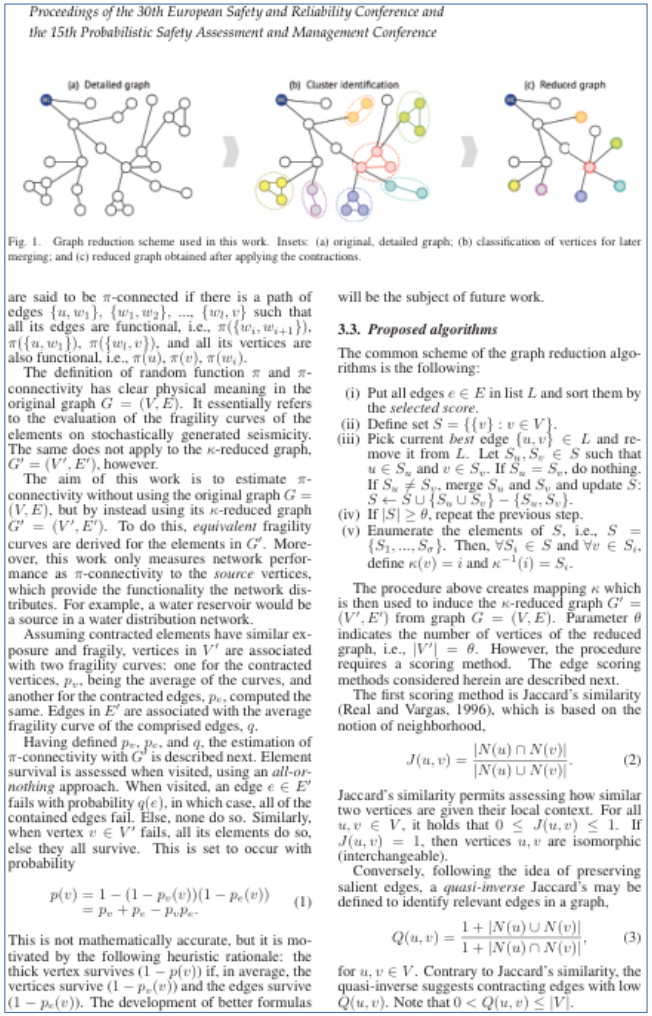
Profesor



- Mauricio Monsalve Moreno
  - Correo preferido: mauricio.monsalve@cigiden.cl
  - Comunicación por Canvas está bien
  - WhatsApp: +56 965257609 (para emergencias)
  - Títulos y grados:
    - PhD in Computer Science, University of Iowa
    - MSc in Computer Science, University of Iowa
    - MSc en Ciencia de la Computación, Universidad de Chile
    - Ing C en Computación, Universidad de Chile
    - Lic en Ciencias de la Ing, mención Computación, Universidad de Chile
  - Perfil en Google Scholar (publicaciones): https://scholar.google.com/citations?user=X0InCdYAAAAJ

PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE

Profesor



Proceedings of the 30th European Safety and Reliability Conference and the 15th Probabilistic Safety Assessment and Management Conference

Fig. 1. Graph reduction scheme used in this work. Insets: (a) original, detailed graph; (b) classification of vertices for later merging; and (c) reduced graph obtained after applying the contractions.

are said to be π-connected if there is a path of edges $\{u, w_1\}$, $\{w_1, w_2\}$, ..., $\{w_l, v\}$ such that all its edges are functional, i.e., $\pi(\{w_i, w_{i+1}\})$, $\pi(\{u, w_1\})$, $\pi(\{w_l, v\})$, and all its vertices are also functional, i.e., $\pi(u), \pi(v), \pi(w_i)$.

The definition of random function π and π-connectivity has clear physical meaning in the original graph $G = (V, E)$. It essentially refers to the evaluation of the fragility curves of the elements on stochastically generated seismicity. The same does not apply to the κ-reduced graph, $G' = (V', E')$, however.

The aim of this work is to estimate π-connectivity without using the original graph $G = (V, E)$, but by instead using its κ-reduced graph $G' = (V', E')$. To do this, *equivalent* fragility curves are derived for the elements in G'. Moreover, this work only measures network performance as π-connectivity to the *source* vertices, which provide the functionality the network distributes. For example, a water reservoir would be a source in a water distribution network.

Assuming contracted elements have similar exposure and fragily, vertices in V' are associated with two fragility curves: one for the contracted vertices, p_v, being the average of the curves, and another for the contracted edges, p_e, computed the same. Edges in E' are associated with the average fragility curve of the comprised edges, q.

Having defined p_v, p_e, and q, the estimation of π-connectivity with G' is described next. Element survival is assessed when visited, using an *all-or-nothing* approach. When visited, an edge $e \in E'$ fails with probability $q(e)$, in which case, all of the contained edges fail. Else, none do so. Similarly, when vertex $v \in V'$ fails, all its elements do so, else they all survive. This is set to occur with probability

$$p(v) = 1 - (1 - p_v(v))(1 - p_e(v)) = p_v + p_e - p_v p_e. \quad (1)$$

This is not mathematically accurate, but it is motivated by the following heuristic rationale: the thick vertex survives $(1 - p(v))$ if, in average, the vertices survive $(1 - p_v(v))$ and the edges survive $(1 - p_e(v))$. The development of better formulas will be the subject of future work.

3.3. *Proposed algorithms*

The common scheme of the graph reduction algorithms is the following:

(i) Put all edges $e \in E$ in list L and sort them by the *selected score*.
(ii) Define set $S = \{\{v\} : v \in V\}$.
(iii) Pick current *best* edge $\{u, v\} \in L$ and remove it from L. Let $S_u, S_v \in S$ such that $u \in S_u$ and $v \in S_v$. If $S_u = S_v$, do nothing. If $S_u \neq S_v$, merge S_u and S_v and update S: $S \leftarrow S \cup \{S_u \cup S_v\} - \{S_u, S_v\}$.
(iv) If $|S| \geq \theta$, repeat the previous step.
(v) Enumerate the elements of S, i.e., $S = \{S_1, ..., S_\sigma\}$. Then, $\forall S_i \in S$ and $\forall v \in S_i$, define $\kappa(v) = i$ and $\kappa^{-1}(i) = S_i$.

The procedure above creates mapping κ which is then used to induce the κ-reduced graph $G' = (V', E')$ from graph $G = (V, E)$. Parameter θ indicates the number of vertices of the reduced graph, i.e., $|V'| = \theta$. However, the procedure requires a scoring method. The edge scoring methods considered herein are described next.

The first scoring method is Jaccard's similarity (Real and Vargas, 1996), which is based on the notion of neighborhood,

$$J(u, v) = \frac{|N(u) \cap N(v)|}{|N(u) \cup N(v)|}. \quad (2)$$

Jaccard's similarity permits assessing how similar two vertices are given their local context. For all $u, v \in V$, it holds that $0 \leq J(u, v) \leq 1$. If $J(u, v) = 1$, then vertices u, v are isomorphic (interchangeable).

Conversely, following the idea of preserving salient edges, a *quasi-inverse* Jaccard's may be defined to identify relevant edges in a graph,

$$Q(u, v) = \frac{1 + |N(u) \cup N(v)|}{1 + |N(u) \cap N(v)|}, \quad (3)$$

for $u, v \in V$. Contrary to Jaccard's similarity, the quasi-inverse suggests contracting edges with low $Q(u, v)$. Note that $0 < Q(u, v) \leq |V|$.

- Investigo en el centro CIGIDEN
- Temas en que trabajo:
  - Riesgo de redes eléctricas (y otras redes)
  - Robustez de la red de urgencias
  - Desarrollo de modelos surrogados
  - Desarrollo de modelos de recuperación
  - Análisis de datos médicos
  - Estimación de pérdidas económicas
  - Estimación de demanda espacial (viajes y preferencias de destino)
  - Estimación espacial de respuesta de sitio

Programo en Python
todos los días

Uso cálculo y álgebra lineal muy seguido

Introducción a la asignatura

PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE

Descripción de la asignatura

La asignatura tiene como propósito que los estudiantes adquieran la capacidad de analizar, diseñar, implementar y ejecutar en una aplicación desde un computador para obtener una solución a un determinado problema, haciendo uso de estructuras (secuenciales, selectivas, repetitivas), arreglos, cadenas de caracteres y subalgoritmos (procedimientos y funciones).

Los estudiantes participarán de clases interactivas y de reforzamiento de ayudantías.

PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE

Competencias asociadas

Competencias asociadas:

Ejerce el mando con liderazgo sobre una sección de fusileros en operaciones militares, a través de un proceso de toma de decisiones basado en la doctrina institucional, con claridad de pensamiento, comprensión situacional, y eficiencia en el uso de los recursos (Competencia N° 1).

Aplica pensamiento crítico y creativo para resolver problemas complejos de su quehacer profesional, utilizando los principios de las disciplinas incluidas en su proceso formativo (Competencia N° 2).

Sub-competencias asociadas:

Distingue los fundamentos y fenómenos que inciden en los sistemas de armas y plataformas tecnológicas de uso en el campo de batalla, con la finalidad de maximizar el uso de sus capacidades en beneficio del cumplimiento de la tarea asignada (Sub competencia N° 1.3).

Resuelve problemas específicos del ámbito militar, integrando el conocimiento adquirido en las distintas disciplinas científicas, con el propósito de posibilitar un eficiente cumplimiento de la tarea asignada (Sub competencia N° 2.1).

PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE

Resultados de aprendizaje

R.A.1: Identifica los componentes de un computador y conceptos esenciales de la programación computacional.

R.A.2: Resuelve problemas en forma rigurosa y creativa, diseñando y describiendo procedimientos ordenados para darles solución.

R.A.3: Utiliza lenguaje de programación para el procesamiento de números, lógica, texto y listas de datos, en una situación dada.

R.A.4: Desarrolla programas orientados a resolver problemas específicos, de acuerdo con soluciones ideadas, verificando su correcto funcionamiento.

PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE

Bibliografía

- Cedano Olvera, M. A. & Rubio González, J. A. (2015). Fundamentos de computación para ingenieros.. Grupo Editorial Patria. https://elibro.net/es/lc/bibliotecasuc/titulos/39445
- Marzal Varó, A. García Sevilla, P. & Gracia Luengo, I. (2016). Introducción a la programación con Python 3. D - Universitat Jaume I. Servei de Comunicació i Publicacions. http://dx.doi.org/10.6035/Sapientia93
- Prieto Espinosa, A. (2013). Introducción a la informática (4a. ed.).. McGraw-Hill España. https://elibro.net/es/lc/bibliotecasuc/titulos/50210
- Vasconcelos Santillán, J. (2018). Introducción a la computación. Grupo Editorial Patria. https://elibro.net/es/lc/bibliotecasuc/titulos/98314

Historia de la computación

PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE

Algo de historia

- **Ábaco**, 3000 AC, Babilonia
- Calculadora con cuentas
- Ábaco con varillas, 1300 DC, China

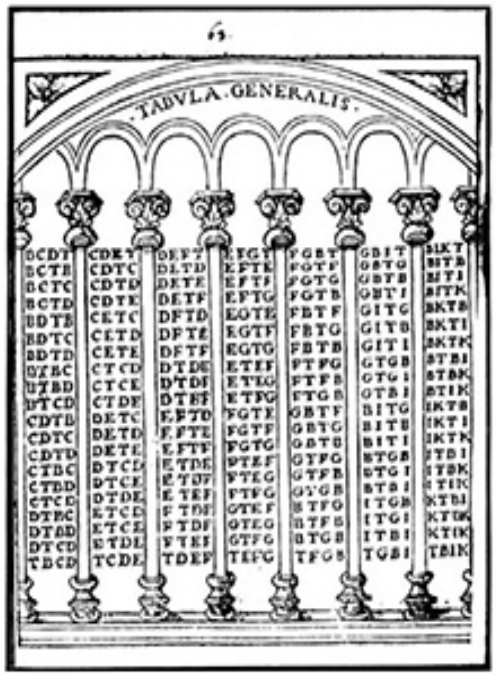
PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE

Mecanismo anticitera

- Descubierto en barco hundido en Anticitera, Grecia en 1900
- Cerca del año 2000, se aplican rayos-X
- Reloj-calendario que predecía fechas, eclipses, posiciones astronómicas y juegos panhelénicos (ej. Olímpicos)
- Se estima origen en 150-100 AC, taller de Arquímedes

PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE

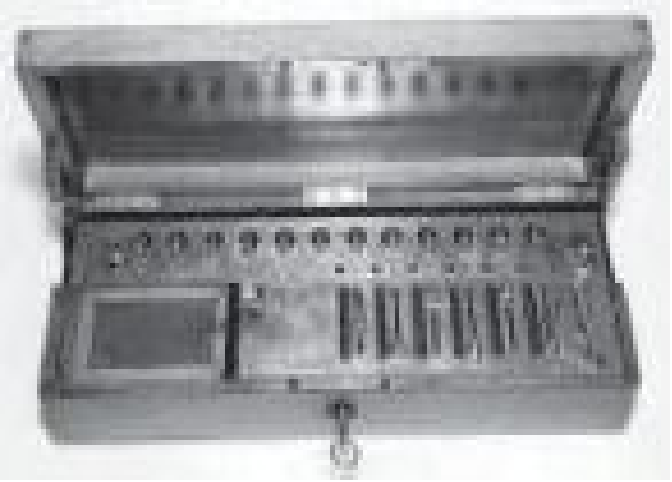
Ruedas y tablas de llulL/LULIO



- Ramon Llull (Raimundo Lulio)
- Alrededor de 1250 desarrolla métodos de elecciones y decisión
- Ruedas y tablas para memorizar y asistir razonamiento lógico
  - "Lulismo" o "Luliano"
- "Ars Magna et Ultima", máquina para deducir cosas por sí misma
  - Lo condenaron por eso

PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE

Calculadoras



- Pascalina: Blaise Pascal, 1642
- Calculadora: Samuel Morland, 1666
- Aritmómetro: Thomas, 1820

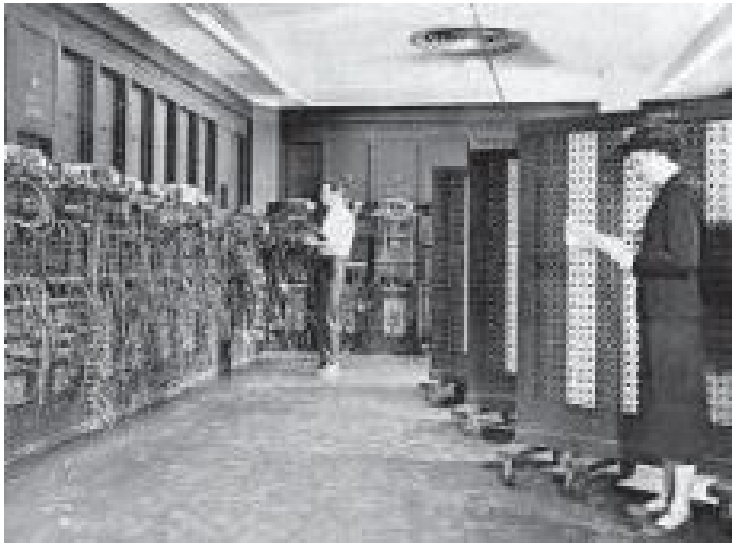
PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE

Máquina diferencial

- Diseñada por Johann Helfrich von Müller, 1786
  - Operaciones polinómicas
- Charles Babbage en 1822 solicitó fondos para construirla, en vano
- Georg Scheutz construye varias en 1855 y las vende
- Babbage diseño la Máquina Analítica, pero los fondos le fueron denegados repetidamente

PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE

Lógica y electrónica



- George Boole en 1854 describe el cálculo proposicional (álgebra lógica)
- En 1904, John Fleming patenta el diodo en vacío, que no permite flujo eléctrico en sentido contrario: comienza la era electrónica
- Computador ENIAC en 1946

PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE

Transición a la computación moderna

- 1945, John von Neumann describe idea de programa almacenado
- 1947, memoria de tambor magnético (antecesor de discos duros)
- 1947, transistor
- 1948, primera computadora digital con programa almacenado
  - Válvulas y tubos de vacío
- 1950, Alan Turing, principios de Inteligencia Artificial
- 1951, Grace Hopper, primer compilador
- Fortran 1957; COBOL 1959; LISP 1959; BASIC 1964; Pascal 1971; C 1972; C++ 1983
- 1968, OTAN, Ing. de Software
- 1968-1969, ARPA-Net

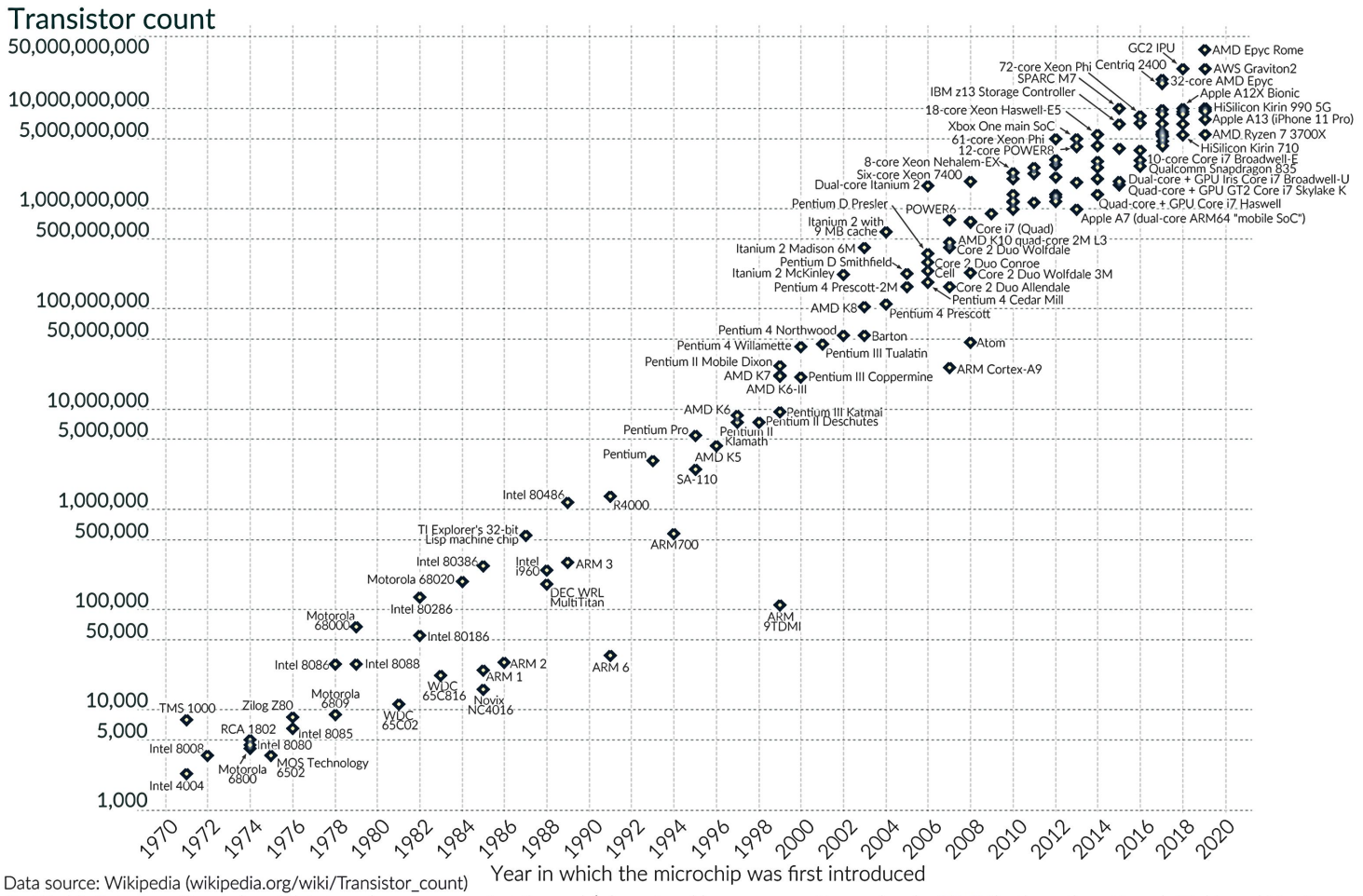
PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE

El hardware evoluciona

Moore's Law: The number of transistors on microchips doubles every two years

Our World in Data

Moore's law describes the empirical regularity that the number of transistors on integrated circuits doubles approximately every two years.
This advancement is important for other aspects of technological progress in computing – such as processing speed or the price of computers.



Data source: Wikipedia (wikipedia.org/wiki/Transistor_count)
OurWorldinData.org – Research and data to make progress against the world's largest problems.

PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE

Cierre de la clase

- La clase cierra mencionando que la velocidad de la luz es un límite físico con el que estamos topando
- Se menciona que la siguiente clase comenzará desde arquitectura de computadores (hardware)