Generating Matroids using HPC-GAP and ArangoDB

Lukas Kühne

August 31, 2017

Joint work with Mohamed Barakat, Reimer Behrends, and Chris Jefferson

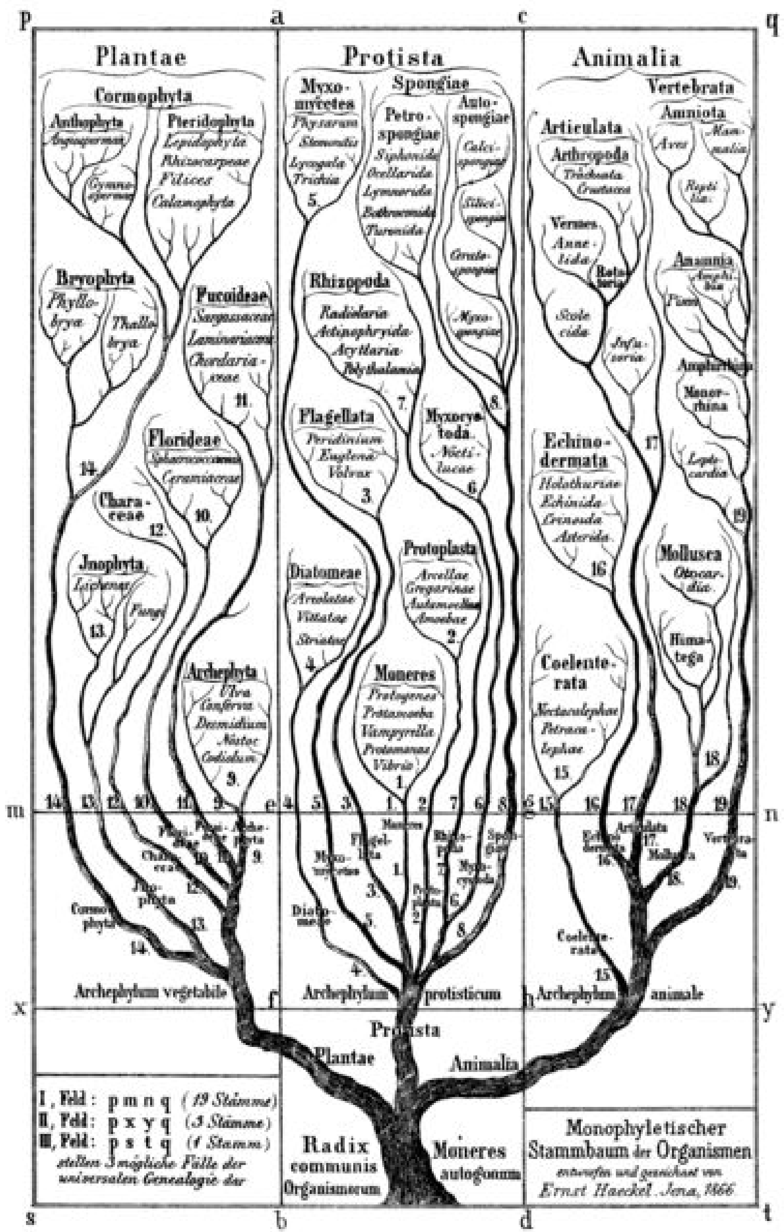
UNIVERSITÄT BONN

Outline

1. Motivation
   - Phylogenetic trees
   - Matroids
2. Parallelized iterator framework
3. Results
4. ArangoDB

Phylogenetic Trees

- **Phylogenetic trees** show the evolutionary relationships among species.
- Studied in bioinformatics.
- Mathematically, they are **binary, rooted trees on** n **labelled leaves**.
- Can be generated via a **search tree**.



Matoids – Definition

Definition

A **matroid** is a pair $(E, \mathcal{I})$, where E is finite set, called **ground set**, and $\mathcal{I}$ is a family of subsets of E, called **independent sets**, with the following properties:

1. The empty set is independent, i.e. $\emptyset \in \mathcal{I}$.
2. Every subset of an independent subset is independent.
3. If A and B are independent sets of $\mathcal{I}$ and $|A| > |B|$, then there exists $x \in A \setminus B$ such that $B \cup \{x\} \in \mathcal{I}$. This property is called **independet set exchange property**.

The cardinality of a maximal independent set of a matroid is called its **rank**.

Matoids – Examples

Example 1 – Vector Matroids

Let E be any finite subset of a vector space V. Define $\mathcal{I}$ to be the subsets of E which are linearly independent.

Matoids – Examples

Example 1 – Vector Matroids

Let E be any finite subset of a vector space V. Define $\mathcal{I}$ to be the subsets of E which are linearly independent.

Example 2 – Graphic Matroids

Let G be a finite graph. Take E to be the set of edges of G and define $\mathcal{I}$ to consist of all subsets of E which do not contain a simple cycle.

Matoids – Examples

Example 1 – Vector Matroids

Let E be any finite subset of a vector space V. Define $\mathcal{I}$ to be the subsets of E which are linearly independent.

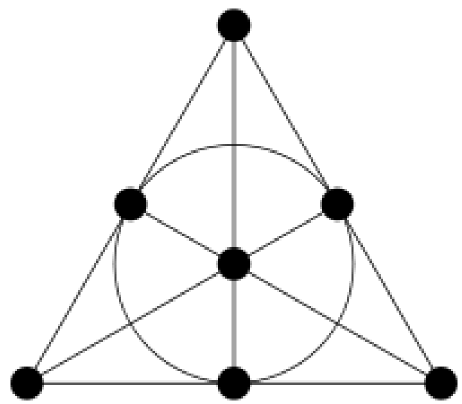
Example 2 – Graphic Matroids

Let G be a finite graph. Take E to be the set of edges of G and define $\mathcal{I}$ to consist of all subsets of E which do not contain a simple cycle.

- Matroids are central objects in combinatorics.
- Introduced by Hassler Whitney in 1935.
- Found applications in many areas, e.g. geometry, algebra and optimization.

Matoids – Representability

- Matroids equivalent to vector matroids of a vector space over a field K are called **representable over K**.
- For example the Fano matroid is representable over $\mathbb{F}_2$ but not over any field K with $\mathrm{char}(K) \neq 2$.
- The study of representable matroids is still widely open.



The Fano matroid. The ground set are the points. A subset of point is independent, if the point do not lie on one line or circle.

Matroids – Our Aims

- Want to perform experiments to study properties like representability on a large testbed of matroids.
- Therefore, we want to generate matroids.
- For simplicity we restrict ourselves to the case of matroids of rank 3.
- In this case, they can be represented as a set of points and lines as the Fano matroid.

Matroids – Search Tree Structure

- The incidence structure of the points and lines can be stored as a bipartite graph.
- We generate matroids characterized by
  - the cardinality of its ground set E,
  - the vector of degrees of the lines in the bipartite graph.
- This gives rise to a **search tree structure**.

Parallelized Iterator Framework

Definition

Let T be a set.

- A **recursive iterator** t **in** T is an iterator which upon popping produces $\mathtt{Pop}(t)$ which is either
  1. a new recursive iterator in T,
  2. an element of T, or
  3. $\mathtt{fail} \notin T$.

  If the pop result $\mathtt{Pop}(t)$ is `fail` then any subsequent pop result of t remains `fail`.

Parallelized Iterator Framework

Definition

Let T be a set.

- A **recursive iterator** t **in** T is an iterator which upon popping produces $\texttt{Pop}(t)$ which is either
  1. a new recursive iterator in T,
  2. an element of T, or
  3. $\texttt{fail} \notin T$.

  If the pop result $\texttt{Pop}(t)$ is $\texttt{fail}$ then any subsequent pop result of t remains $\texttt{fail}$.
- A **full evaluation** of a recursive iterator recursively pops all recursive iterators until each of them pops `fail`.

Parallelized Iterator Framework

Definition

Let T be a set.

- A **recursive iterator** t **in** T is an iterator which upon popping produces $\mathtt{Pop}(t)$ which is either
  1. a new recursive iterator in T,
  2. an element of T, or
  3. $\mathtt{fail} \notin T$.

  If the pop result $\mathtt{Pop}(t)$ is `fail` then any subsequent pop result of t remains `fail`.
- A **full evaluation** of a recursive iterator recursively pops all recursive iterators until each of them pops `fail`.
- If t is a recursive iterator then the subset of elements $T(t) \subset T$ produced upon full evaluation is called the **set of leaves** of t.

Parallelized Iterator Framework

Input: A recursive iterator t, a number $n \in \mathbb{N}_{>0}$ of workers and a global FiFo $e = ()$ accessible by other processes.

Output: none; the side effect is to fill e with leaves in $T(t)$

Initialize a farm w of n workers $w_1, \ldots, w_n$

Initialize a shared prioritized queue $S := (t, 0)$ of iterators

while `true` **do**

  for all nonbusy w_i **parallel do**

    if `NoHighestPriorityIteratorAndNoBusyWorkers`(S) **then**

      `Add`$(e,$ `fail`$)$ and **return none** globally

    $(t_i, p_{t_i}) :=$ `Pop`(S)

    $r_i :=$ `Pop`$_{w_i}(t_i)$; i.e., use worker w_i to pop t_i

    if $r_i \in T$ **then**

      `Add`(e, r_i) and `Add`$(S, (t_i, p_{t_i}))$

    elif $r_i \neq$ `fail` **then**

      `Add`$(S, (t_i, p_{t_i}))$ `Add`$(S, (r_i, p_{t_i} + 1))$

Results – Phylogenetic Trees

Comparison of the run time for generating phylogenetic trees on n leaves.

n	Number of Phylotrees	GAP (mm:ss)	HPC–GAP (mm:ss) (Walltime) 1	2	4	8
10	4,862	00:00	00:02	00:01	00:02	00:03
11	16,796	00:01	00:08	00:06	00:05	00:07
12	58,786	00:02	00:19	00:20	00:21	00:25
13	208,012	00:08	01:16	01:07	01:09	01:31
14	742,900	00:31	03:57	04:07	03:58	05:19
15	2,674,440	01:34	13:08	14:15	13:57	17:06

Results – Matroids

Comparison of the run time for generating simple rank 3 matroids with ground set of cardinality n.

n	Number of Matroids	GAP (hh:mm:ss)	HPC–GAP (hh:mm:ss) (Walltime) 1	2	4	8
7	23	00:00:01	00:00:00	00:00:00	00:00:00	00:00:00
8	68	00:00:09	00:00:09	00:00:06	00:00:06	00:00:05
9	383	00:08:43	00:08:48	00:06:22	00:05:19	00:05:15
10	5249	?	?	?	?	?

- 11: 232928
- 12: 28872972
- 13: Unknown

Summary

- We want to study properties like representability on a large set of matroids.
- To this end we have developed a general framework of parallelized iterators in HPC-GAP.
- We have linked it to a database using ArangoDB.
- Maybe this general setup is also useful in other situations?

Thank you for your attention!