Aalto University

HELSINKI INSTITUTE FOR INFORMATION TECHNOLOGY

UNIVERSITY OF HELSINKI
FACULTY OF SCIENCE

Optimizing Phylogenetic Supertrees Using Answer Set Programming

Laura Koponen[1], *Emilia Oikarinen*[1], Tomi Janhunen[1], and Laura Säilä[2]

[1] HIIT / Dept. Computer Science, Aalto University
[2] Dept. Geosciences and Geography, University of Helsinki

Computational logic day 2015 — Aalto, Finland

Outline

Introduction — the supertree problem

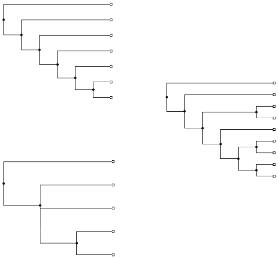
ASP Encodings — trees, quartets and projections

Experiments — Felidae data

Conclusions

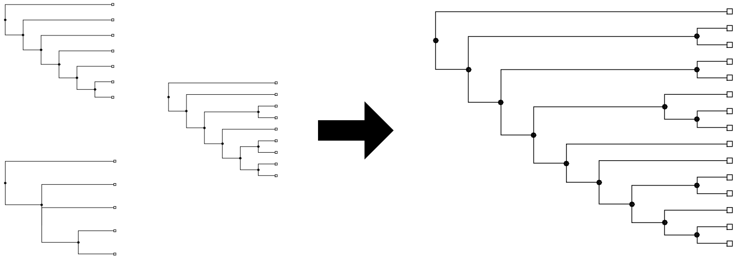
The supertree problem

- Input: a set of overlapping, possibly conflicting phylogenetic trees (rooted, leaf-labeled)



The supertree problem

- Input: a set of overlapping, possibly conflicting phylogenetic trees (rooted, leaf-labeled)



- Output: a phylogenetic tree that covers all taxa from input and reflects the relationships in input as well as possible
  - Several measures can be used used
  - Optimal tree not necessarily unique

Solving the supertree problem

- Typically *heuristic* methods are used, e.g. *matrix representation with Parsimony* (MRP) [Baum, 1992; Ragan,1992]
  - input trees encoded into a binary matrix, and maximum parsimony analysis is then used to construct a tree
  - no guarantee of finding optimal solution
  - large supertrees (hundreds of species) still computationally challenging
- There exist earlier constraint-based approaches for related *phylogeny reconstruction problem*
  - cladistics-based apporach using ASP [Brooks et al., 2007]
  - maximum parsimony using ASP [Kavanagh et al., 2006] and MIP [Sridhar et al., 2008]
  - *maximum quartet consistency* problem using ASP [Wu et al., 2007] and CP [Morgado & Marques-Silva, 2010]

In this paper

- We solve the supertree problem using *answer set programming*
  - Rule-based, expressive language for knowledge representation, efficient solvers (moreover, possible to enumerate *all* optimal solutions)

In this paper

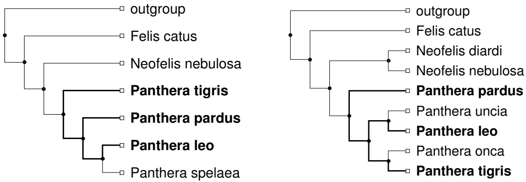
- We solve the supertree problem using *answer set programming*
  - Rule-based, expressive language for knowledge representation, efficient solvers (moreover, possible to enumerate *all* optimal solutions)
- We present two alternative encodings (with different optimization criteria) solving:
  - maximum quartet consistency problem
  - maximum projection consistency problem

In this paper

- We solve the supertree problem using *answer set programming*
  - Rule-based, expressive language for knowledge representation, efficient solvers (moreover, possible to enumerate *all* optimal solutions)
- We present two alternative encodings (with different optimization criteria) solving:
  - maximum quartet consistency problem
  - maximum projection consistency problem
- We apply the encodings on real data (*Felidae*) and compare our supertrees to recent supertrees obtained using the heuristic MRP method

Supertree problem: practical considerations

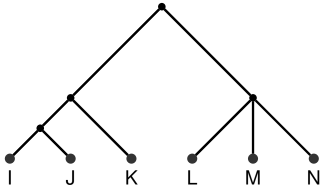
- How to resolve conflicts in the input trees? How to localize the information in trees?



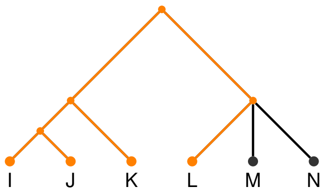
- The search space (number of rooted leaf-labeled trees) grows exponentially

Taxa	Different trees
1	1
2	1
3	4
4	26
5	236
...	...
10	282 137 824
...	...
15	6 353 726 042 486 112
...	...

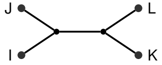
Representing input trees with substructures



Representing input trees with substructures

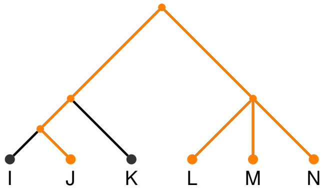


- Quartet (unrooted tree with four leaf nodes)

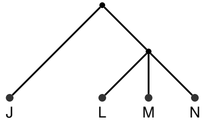


  - n leaf nodes, $\binom{n}{4}$ quartets
  - a 50-taxa tree has 230 300 quartets

Representing input trees with substructures



- Projections



  - $2^n - 1$ different projections for tree with n leaf nodes
  - a 50-taxa tree has 1.13×10^{15} projections
  - to reduce the amount, consider only *subtree projections*

Outline

Introduction — the supertree problem

ASP Encodings — trees, quartets and projections

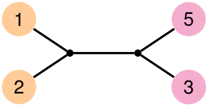
Experiments — Felidae data

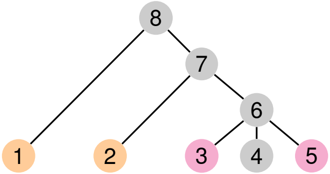
Conclusions

Representing canonical trees

- Non-binary, rooted leaf-labeled trees encoded using `node/1` and `edge/2` predicates
  - inner nodes (`inner/1`) have larger indices than leaf nodes (`leaf/1`)
  - edges directed from larger indices to smaller ones
- Taxa are assigned to leaf nodes using a fixed alphabetical order (`asgn/2`)
- To further reduce symmetries, a *canonical labeling* for nodes is introduced
  - generalization of the condition in [Brooks et al., 2007]
- Special taxon *outgroup* placed as a child on the root

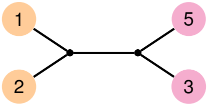
Quartets displayed by a tree

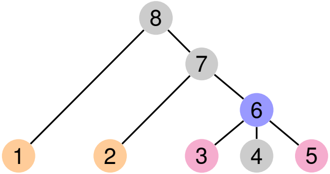




- How to determine if a tree displays quartet ((1, 2), (3, 5))?
  - Are pairs (1, 2) and (3, 5) separated by an edge in the tree?

Quartets displayed by a tree

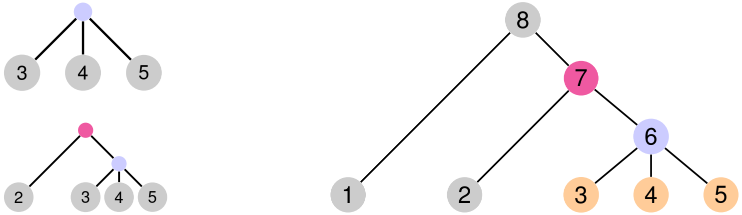




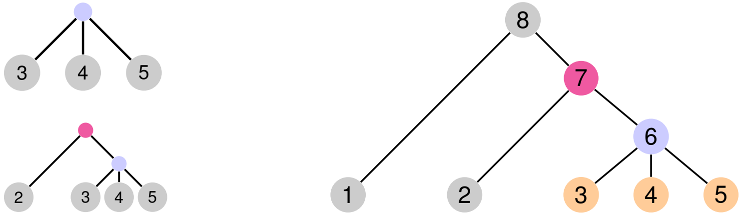
- How to determine if a tree displays quartet ((1, 2), (3, 5))?
  - Are pairs (1, 2) and (3, 5) separated by an edge in the tree?

$$\begin{aligned}\mathtt{satisfied}(A1, A2, A3, A4) \leftarrow\ & \mathtt{quartet}(A1, A2, A3, A4), \\ & \mathtt{reach}(X, A1), \mathtt{reach}(X, A2), \\ & \mathtt{not\ reach}(X, A3), \\ & \mathtt{not\ reach}(X, A4), \\ & \mathtt{inner}(X).\end{aligned}$$

Projections displayed by a tree



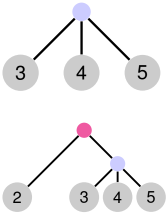
Projections displayed by a tree

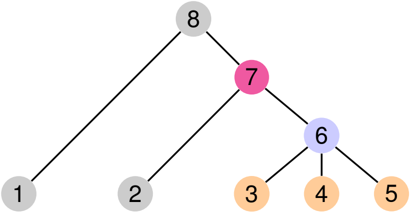


- Projections are by default *assigned* to inner nodes

$$\texttt{asgn}(X, P) \leftarrow \texttt{inner}(X), \texttt{not denied}(X, P).$$

Projections displayed by a tree



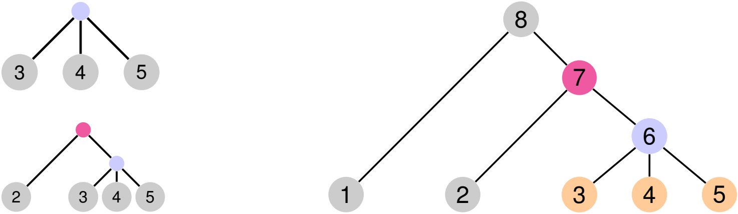


- Projections are by default *assigned* to inner nodes

$$\texttt{asgn}(X,P) \quad \leftarrow \quad \texttt{inner}(X), \texttt{not}\ \texttt{denied}(X,P).$$

- Predicate `denied/2` specifies exceptions

Projections displayed by a tree



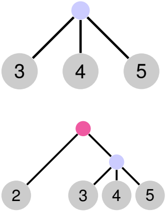
- Projections are by default *assigned* to inner nodes

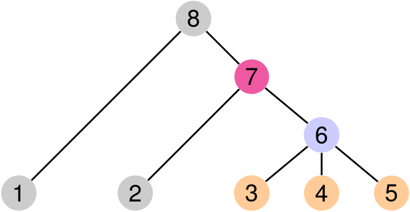
$$\texttt{asgn}(X, P) \quad \leftarrow \quad \texttt{inner}(X), \texttt{not denied}(X, P).$$

- Predicate `denied/2` specifies exceptions
  - Projection P cannot be assigned to X if it is assigned to a node below X

$$\texttt{denied}(X, P) \quad \leftarrow \quad \texttt{edge}(X, Y), \texttt{reach}(Y, P).$$

Projections displayed by a tree





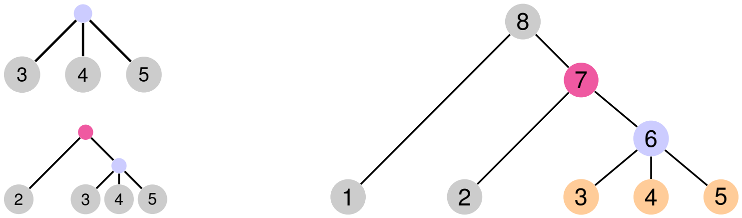
- Projections are by default *assigned* to inner nodes

$$\mathtt{asgn}(X, P) \leftarrow \mathtt{inner}(X), \mathtt{not}\ \mathtt{denied}(X, P).$$

- Predicate `denied/2` specifies exceptions
  - Distinct child projections of P cannot be mapped on the same subtree in the phylogeny

$$\begin{aligned} \mathtt{denied}(X, P) \leftarrow\ & \mathtt{edge}(X, Y), \mathtt{reach}(Y, A), \mathtt{reach}(Y, B), \\ & \mathtt{child}(A, P), \mathtt{child}(B, P), A < B. \end{aligned}$$

Projections displayed by a tree



- Projections are by default *assigned* to inner nodes

$$\texttt{asgn}(X, P) \quad \leftarrow \quad \texttt{inner}(X), \texttt{not}\ \texttt{denied}(X, P).$$

- Predicate `denied/2` specifies exceptions
  - If projection P is assigned at inner node X, then its child projections must have been assigned below X in the tree

Outline

Introduction — the supertree problem

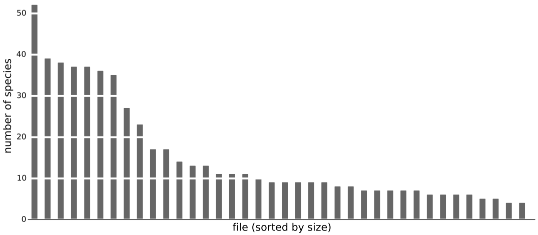
ASP Encodings — trees, quartets and projections

Experiments — Felidae data

Conclusions

Dataset: Felidae

- 38 source trees with 105 species of cats from [Säilä et al., 2011, 2012]



- Problem: 105 species are too much for the current encodings

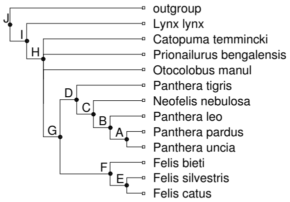
Scalability: genus-specific projections of data

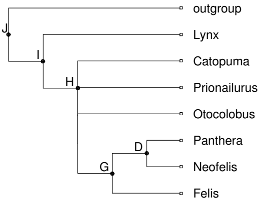
			CLASP		WASP		ACYC		CLASP-S	
Genus	Taxa	Trees	qtet	proj	qtet	proj	qtet	proj	qtet	proj
Leopardus	8	6	0.6	0.1	1.7	0.2	1.1	0.4	0.6	0.1
Dinofelis	9	2	0.1	0.0	0.0	0.1	0.1	0.1	0.0	0.1
Homotherium	9	3	0.7	0.0	0.1	0.1	0.1	0.0	0.0	0.0
Felis	11	12	39.6	21.9	291	121	123	59.6	27.7	20.8
Panthera	11	22	1400	45.6	–	456	–	175	944	67.1

- Time (s) to find one optimum for genus-specific data using different solvers using quartet (qtet) and projection (proj) encoding (– marks timeout of 1 hour).
- The projection encoding with CLASP looks as the most promising combination

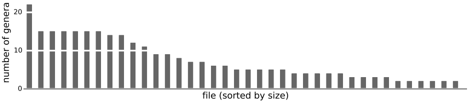
Genus-level Felidae supertree

- Idea: project trees onto genus-level





- 105 species of cats ⇒ 34 genera, 28 source trees



Genus-level Felidae supertree — results

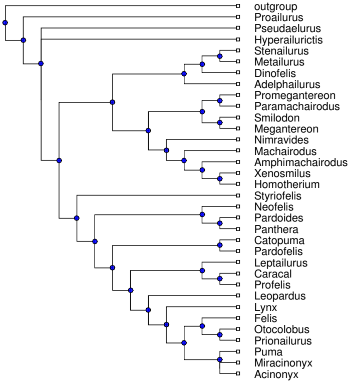
- Quartet encoding was still too slow (timeout 48 hours)
  - Suboptimal solutions could be obtained
- Projection encoding produced optimal supertrees
  - For this data, unique optimum exists
- The supertrees were compared to recent supertrees computed using MRP [Säilä et al. 2011, 2012]
  - In [Säilä et al. 2011, 2012] MRP trees selected with *best resolution* (MRP-R) and *best support* (MRP-S)
  - These are projected onto genus-level to allow for comparison

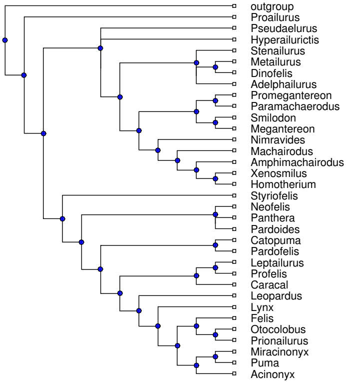
Supertree comparison — quality measures

Scheme	Quartets %	Resolution	Support
Proj	**0.84**	0.90	0.43
MRP-S	0.77	0.85	**0.45**
MRP-R	0.83	**0.93**	0.42

- Resolution: percentage of resolved nodes in the tree
- Quartets %: percentage of displayed quartets from input
- Support: [Wilkinson et al., 2005]

Supertree comparison





genus-level MRP-R

projection encoding optimum

Outline

Conclusions

- Two encodings for solving the supertree problem $\Rightarrow$ projection-based encoding looks promising in terms of performance and tree quality
- Large supertrees not possible yet
  - need for a strategy to, e.g., split the instance
  - more analysis of bottlenecks — need for more data, both artificial and real
- Furthermore, work is need on improving the properties of the objective function
  - Currently larger trees get more weight, though this is not (always) desirable