A generic Approach for Sparse Path Problems

Marc Pouly

marc.pouly@unifr.ch

Cork Constraint Computation Centre
University College Cork, Ireland

ICAART Conference, Valencia 2010

Tree-Decomposition Methods

Tree-decomposition methods have been developed for many different formalisms

Discrete Probabilities

- Bayesian Networks
- Credal Networks

Belief Functions

(Dempster-Shafer Theory)

Density Functions

- Gaussian Potentials
- Gaussian Hints

Semiring Constraints

- Crisp constraints
- Probabilistic constraints
- Weighted constraints
- Possibilistic constraints
- Set constraints

Relational Algebra

Possibility Potentials

Logic

- Propositional Logic
- Predicate Logic
- Assumption based Reasoning

Linear Systems

(Sparse Matrix Techniques)

- Affine Spaces
- Pos. def. Matrices

The same algorithms are re-invented for each formalism !

Valuation Algebras

- Axiomatic framework enabling generic tree-decomposition
- Common characteristics of these formalisms:
  - Knowledge exists in pieces $\rightsquigarrow$ valuations
  - Knowledge refers to questions (variables) $\rightsquigarrow$ labeling
  - Pieces of knowledge can be combined
  - Knowledge can be projected
- A valuation algebra therefore consists of:
  - Variables r & Valuations Φ
  - Combination $\otimes$ and Projection $\downarrow$
  - 6 Axioms describing their behaviour

The Inference Problem

Given a set of valuations $\{\phi_1, \dots, \phi_n\}$ and a query x, compute

$$(\phi_1 \otimes \phi_2 \otimes \cdots \otimes \phi_n)^{\downarrow x}$$

Depending on the valuation algebra, this task ...

- ... evaluates Bayesian networks
- ... answers queries in relational databases
- ... solves linear equation systems
- ... checks satisfiability of constraint problems
- ... computes Fourier and Hadamard transforms

Complexity Concerns

Example

Valuations: $\{\phi_1, \phi_2, \phi_3\}$

Domains: $d(\phi_1) = \{A, B, D\}, \; d(\phi_2) = \{B, C\}, \; d(\phi_3) = \{C\}$

Query: $x = \{A, C\}$

Domains grow under combination !

If $\phi = \phi_1 \otimes \phi_2 \otimes \phi_3$ then $d(\phi) = \{A, B, C, D\}$

- Complexity of $\otimes, \downarrow$ often increase exponentially with domain size
- Algorithms must limit domain size of intermediate results

The Promise of Tree-Decomposition Methods

- Tree-decomposition methods find alternative factorizations:

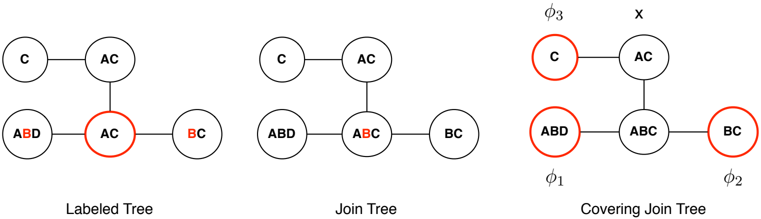
$$(\phi_1 \otimes \phi_2 \otimes \phi_3)^{\downarrow\{A,C\}} = \left(\phi_1^{\downarrow\{A,B\}} \otimes \phi_2\right)^{\downarrow\{A,C\}} \otimes \phi_3$$

  Here, the largest domain has only 3 variables

- How can we find such factorizations ? $\rightsquigarrow$ Join Trees

Join Trees & Treewidth

- Join Tree $=$ Labeled Tree $+$ Running Intersection Property
- Each query x must be covered by some node
- The domain of each factor ϕ_i must be covered by some node

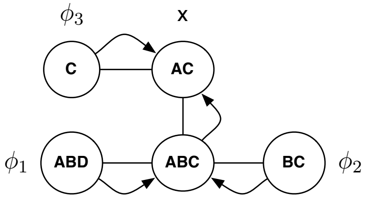


- Treewidth $=$ largest join tree node $\rightsquigarrow$ 3

Local Computation

Message-passing algorithm identifies the factorization:

$$\left(\phi_1^{\downarrow\{A,B\}} \otimes \phi_2\right)^{\downarrow\{A,C\}} \otimes \phi_3$$



All results are bounded by the node labels ...

Treewidth determines complexity

Quasi-Regular Semirings

Algebraic structure $\langle A, +, \times, *, \mathbf{0}, \mathbf{1} \rangle$

- $+$ and $\times$ are associative, $+$ is commutative
- $\times$ distributes over $+$: $a \times (b + c) = (a \times b) + (a \times c)$
- zero element: $\mathbf{0}$, unit element: $\mathbf{1}$
- quasi-inverse a^* such that $a^* = aa^* + \mathbf{1}$

Some Examples:

- Boolean Semiring: $\langle \{0, 1\}, \vee, \wedge, 0, 1 \rangle$ with $a^* = 1$
- Tropical Semiring: $\langle \mathbb{N} \cup \{0, \infty\}, \min, +, \infty, 0 \rangle$ with $a^* = 0$
- Probabilistic Semiring: $\langle [0, 1], \max, \cdot, 0, 1 \rangle$ with $a^* = 1$

Algebraic Path Problem

Input: Matrix M with values from a quasi-regular semiring

- All-Pairs Algebraic Path Problem:

$$X = MX + I$$

- Single-Source Algebraic Path Problem:

$$x = xM + b$$

M^* and bM^* are solutions to these fixpoint equations

Quasi-Inverse Matrices

Theorem (Lehmann, 1976)

M^* of a matrix $M : n \times n \rightarrow A$ over a quasi-regular semiring A can be computed from the quasi-inverses of the semiring elements.

- For example using the Floyd-Warshall-Kleene Algorithm
- Complexity: $O(n^3)$

Factorized Path Problems

- Considering decomposed graphs for path problems is natural (e.g. shortest distance over multiple maps)
- A sparse matrix can be regarded as a decomposition:

	Berlin	Paris	Rome	Berne
Berlin	0	1111	∞	∞
Paris	∞	0	∞	∞
Rome	1181	∞	0	∞
Berne	∞	436	∞	0

=

0	Berlin	Paris
Berlin	0	1111
Paris	∞	0

$\rightsquigarrow M_1$

0	Berne	Paris
Berne	0	436
Paris	∞	0

$\rightsquigarrow M_2$

0	Rome	Berlin
Rome	0	1181
Berlin	∞	0

$\rightsquigarrow M_3$

- Shortest distance from Rome to Berlin:

$$M^* = (M_1 + M_2 + M_3)^{* \downarrow \{Rome, Berlin\}}$$

Path Problems and Sparse Matrix Techniques

Can this be done by tree-decomposition methods ?

1. Answer from sparse matrix people: (Radhakrishnan et al, 1992)
   - LDU & fill-ins restriction $\rightsquigarrow$ treewidth complexity
   - This forms a VA and is equal to LC (Kohlas & Pouly, 2009)
   - This tackles the single-source problem
   - Repeated application for the all-pairs / multi-pairs problem
2. There is a second possibility ...
   - Quasi-inverse matrices (may) form a valuation algebra
   - This tackles the all-pairs / multi-pairs problem directly

Valuation Algebra Operations

We only need to verify the Valuation Algebra Axioms !

Labeling: $d(M^*) = s$ if $M^* : s \times s \rightarrow A$

Projection: For $t \subseteq d(M^*)$, $(M^*)^{\downarrow t}$ is matrix restriction

Combination: For M_1^* and M_2^* with $d(M_1^*) = s$ and $d(M_2^*) = t$

$$M_1^* \otimes M_2^* \quad = \quad \left(M_1^{*\uparrow s \cup t} \; + \; M_2^{*\uparrow s \cup t}\right)^*$$

Valuation Algebra of Closures

Theorem (Pouly, 2008)

If A is a Kleene Algebra, then this algebra of matrix closures with combination and projection forms a valuation algebra

In addition to a quasi-regular semiring, we *need*

1. Idempotent addition: $a + a = a$ for all $a \in A$
2. Closure Property: $a^{**} = a^{*}$

Kleene Algebras guarantee these properties

Solving Factorized Path Problems I

Input:

- Matrices $\{M_1, \ldots, M_n\}$ taking values from a Kleene Algebra
- Query set: $\{(s_1, t_1), \ldots, (s_m, t_m)\} \subseteq r \times r$

Naive Algorithm:

1. Compute $M = M_1 + M_2 + \ldots + M_n$
2. Compute M^*
3. Answer queries (table lookup)

Complexity: $O(|s|^3)$ where $s = d(M_1) \cup \ldots \cup d(M_n)$

Solving Factorized Path Problems II

By Local Computation:

1. Construct join tree
2. Run generic local computation algorithm which returns

$$\left[\left(M_1^* + \ldots + M_n^*\right)^*\right]^{\downarrow\{s_i,t_i\}}$$

Complexity: $O(|V| \cdot \omega^3)$ where ω is the treewidth

Example

- 14 European countries $\rightsquigarrow$ 14 valuations
- 60 cities (max. 7 per country) $\rightsquigarrow$ $s = 60$
- Distances between neighboring capitals are the only known international distances
- Treewidth: $\omega = 13$ join tree nodes: $|V| = 23$
- Complexity: $O(s^3)$ versus $O(|V| \cdot \omega^3)$
- For comparison: $60^3 = 216'000$ and $|V| \cdot \omega^3 = 50'531$

Handling Query Sets

- LC requires that all queries are covered by the join tree.
- In path problems we often have large query sets.
- This increases the treewidth unnecessarily.
- Instead, we ignore the query set for LC and compute queries later on the propagated join tree.

  $\rightsquigarrow$ compilation and query phase.

Query Answering Procedure

Input:

- Input: propagated join tree (V, E) with $\phi^{\downarrow \lambda(i)}$ for all $i \in V$.
- Input: query (X, Y)

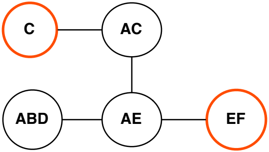
Algorithm:

1. find a path $(p_1, \ldots, p_k)$ such that $X \in \lambda(p_1)$ and $Y \in \lambda(p_k)$;
2. initialize $\eta = \phi^{\downarrow \lambda(p_1)}$
3. for $i = 1 \ldots k-1$ do

$$\eta \quad := \quad \phi^{\downarrow \lambda(p_{i+1})} \otimes \eta^{\downarrow \lambda(p_1) \cup (\lambda(p_i) \cap \lambda(p_{i+1}))}$$

4. return $\eta^{\downarrow \{X, Y\}}$;

Query Answering Example



$\phi^{\downarrow\{C\}}$ $\phi^{\downarrow\{A,C\}}$ $\phi^{\downarrow\{A,E\}}$ $\phi^{\downarrow\{A,B,D\}}$ $\phi^{\downarrow\{E,F\}}$

1. Query: $\{C, F\}$, Path: $\{C\} \to \{A, C\} \to \{A, E\} \to \{E, F\}$
2. $\eta = \phi^{\downarrow\{C\}}$
3. $\eta = \phi^{\downarrow\{A,C\}} \otimes \eta^{\downarrow\{C\}\cup(\{C\}\cap\{A,C\})}$
4. $\eta = \phi^{\downarrow\{A,E\}} \otimes \eta^{\downarrow\{C\}\cup(\{A,C\}\cap\{A,E\})}$
5. $\eta = \phi^{\downarrow\{E,F\}} \otimes \eta^{\downarrow\{C\}\cup(\{A,E\}\cap\{E,F\})} = \phi^{\downarrow\{C,E,F\}}$
6. query answer: $\eta^{\downarrow\{C,F\}}$

Query Answering Complexity

$$\eta \;:=\; \phi^{\downarrow \lambda(p_{i+1})} \otimes \eta^{\downarrow \lambda(p_1) \cup (\lambda(p_i) \cap \lambda(p_{i+1}))}$$

- longest path has at most $|V|$ nodes
- largest domain: $\lambda(p_1) \cup \lambda(p_{i+1})$
- complexity is bounded by $2\times$ treewidth

$$\mathcal{O}\Big(|V| \cdot (2 \cdot \omega)^3\Big) \;=\; \mathcal{O}\Big(|V| \cdot \omega^3\Big)$$

- same complexity as propagation !

Conclusion

- We deliver the algebraic foundation of sparse matrix techniques for the solution of path problems
- Existing methods are equal to tree-decomposition algorithms in AI $\rightsquigarrow$ generic algorithms
- Transfer of research results (e.g. updating)
- We introduced a new VA based on matrix closures