Probabilistic graphical models such as Bayesian networks and junction trees are widely used to represent joint probability distributions in a number of domains. Exact inference is a key problem in exploring such probabilistic graphical models. In our research, we propose efficient techniques for parallelizing exact inference at various levels and mapping the parallelism onto state-of-the-art multicores processors. We convert exact inference into a scheduling problem for the DAG structured computations, and optimize the scheduling scheme for various multicores architectures. The experimental results show that our proposed methods achieved superior performance compared with various baseline methods.

**Abstract**

**Background**

- **Bioinformatics**
  - Gene regulation network
  - Gene expression analysis
  - Protein structure prediction
- **Automation**
  - Adaptive testing
  - Automatic troubleshooting
  - Decision support system
- **Network scale is large**
- **Real time constraints**

**Bayesian network**

- Each node represents a random variable
- Each edge indicates the dependency relationship
- A conditional probability table (CPT) per node

**Real applications**

- Gene regulation network
- Gene expression analysis
- Protein structure prediction
- Automation
  - Adaptive testing
  - Automatic troubleshooting
  - Decision support system

**Junction tree**

- A clique is a set of r.v. from Bayesian Network
- A separator is the shard r.v. between adjacent cliques
- A potential table (POT) describes the joint distribution of r.v. in a clique or separator

**Evidence propagation using node level primitives**

1. Marginalize a clique POT to a separator POT
2. Divide the result with a updated separator POT
3. Extend the division result
4. Multiply the extended POT with the clique POT
5. Marginalize the clique POT to separator POTs

**Efficiency of junction tree rerooting**

- A straightforward method: $O(wN^2)$
- The proposed method: $O(wN)$
Junction tree decomposition

- Decompose a junction tree into chains or subtrees
- Perform evidence propagation in each chain (subtree) independently
- Merge all the duplicated cliques in parallel
- Suitable for systems with distributed memory and high communication latency

Duplicated cliques are partially updated
- Obtain fully updated cliques by merging
- Merge cliques using separators reduces communication cost

Decompose a junction tree to chains

Exact inference and DAG scheduling

From junction tree to task graph

Given an arbitrary junction tree, the task graph is constructed by replacing cliques by node primitives with precedence constraints

Tasks corresponding to large POTs are partitioned at run time

Task graph representation for exact inference

- The task graph is stored as an adjacency array
- Precedence is maintained using dependency degree and successor links
- The POTs are stored as a separate list, linked by the adjacency array

Scheduler design and processor architectures

- Light weighted dynamical scheduler with task partitioning
- Adaptive to multicore architectures
  - Cache based homogeneous multicore processors
    - Distributed scheduling activities
    - Collaboration for load balancing
    - Software lock-free data structure
  - Heterogeneous multicore processors
    - Centralized scheduler on PPE
    - Task partitioning due to limited local store
    - Overlap between scheduling and computation

Experiments

Platforms with dual quadcore processors
- Intel Xeon 5330 (Clovertown)
- AMD Opteron 2350 (Barcelona)

Baseline methods for comparison
- Intel PNL based exact inference
- OpenMP based implementation
- Data parallel method

Compare with baseline methods

Speedup with various junction trees