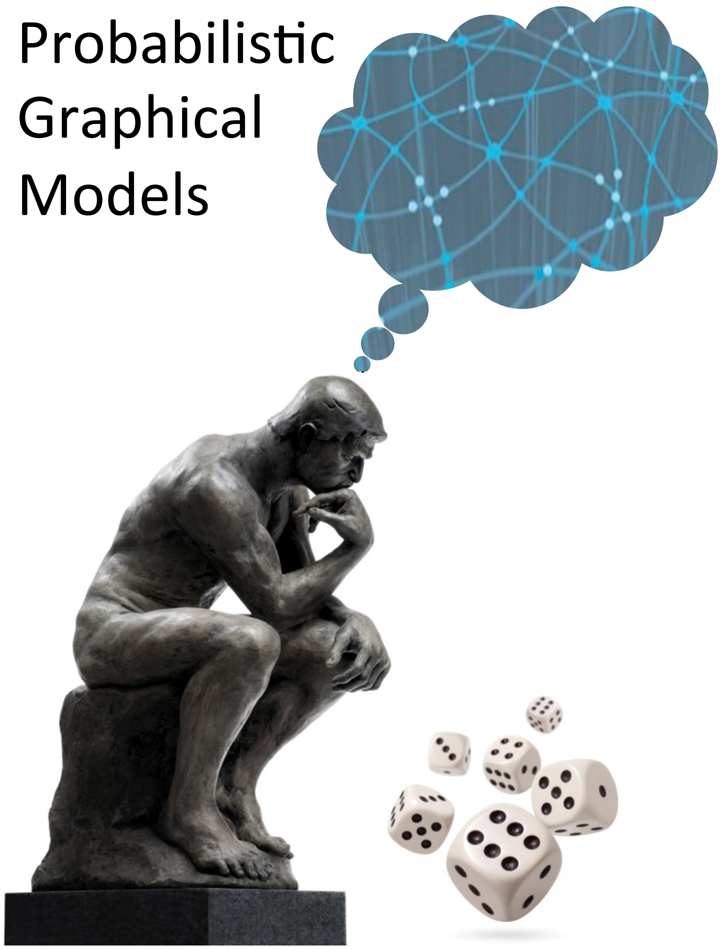
Probabilistic Graphical Models

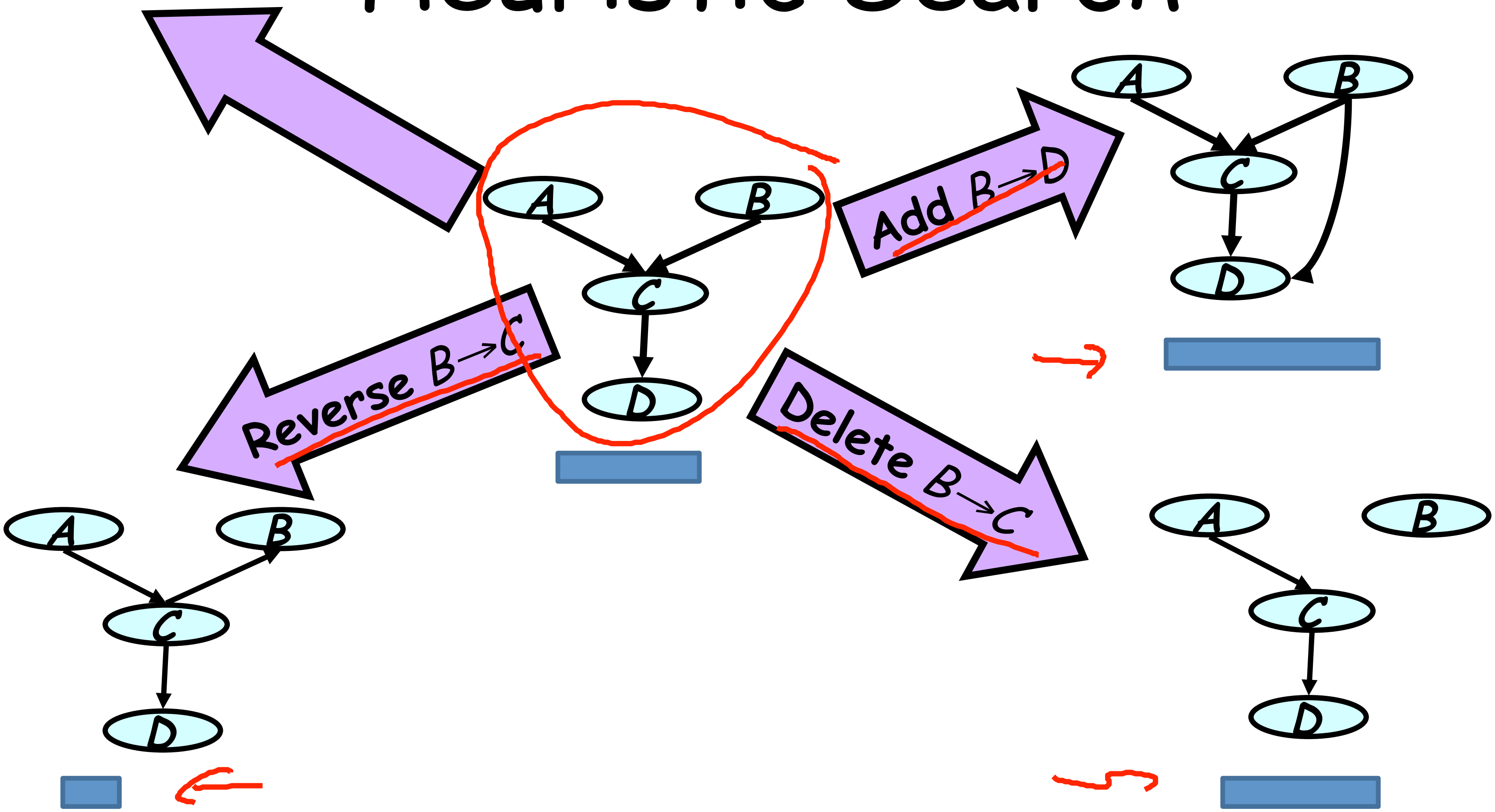


Learning

BN Structure

General Graphs: Decomposability

Heuristic Search



Daphne Koller

Naïve Computational Analysis

- Operators per search step:

  $n(n-1)$ possible edges $O(n^2)$ present – delete – reverse 2 absent – add 1

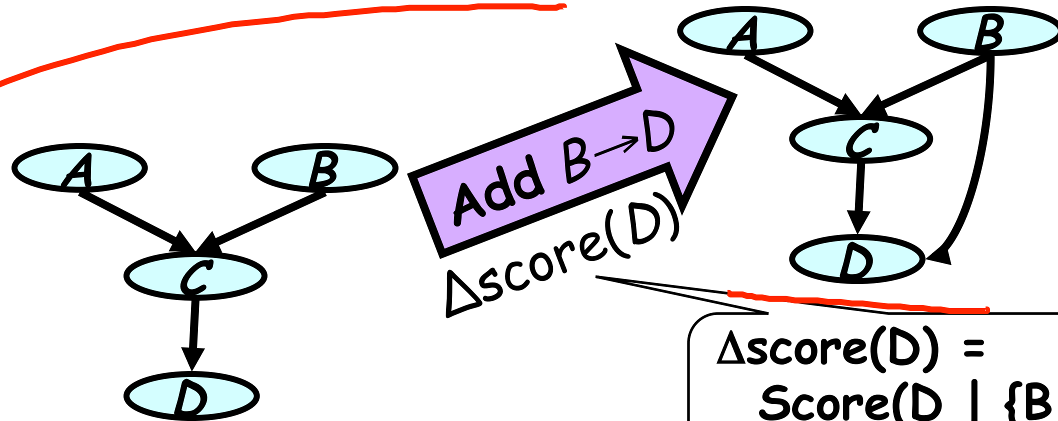
- Cost per network evaluation:
  - Components in score n components
  - Compute sufficient statistics $O(M)$ } $O(n \cdot M)$
  - Acyclicity check $O(m) \leftarrow$ # edges

- Total: $O(\underline{n^2}\ (\underline{Mn} + \underline{m}))$ per search step

  d

Daphne Koller

Exploiting Decomposability



$$\Delta\text{score}(D) = \text{Score}(D \mid \{B,C\}) - \text{Score}(D \mid \{C\})$$

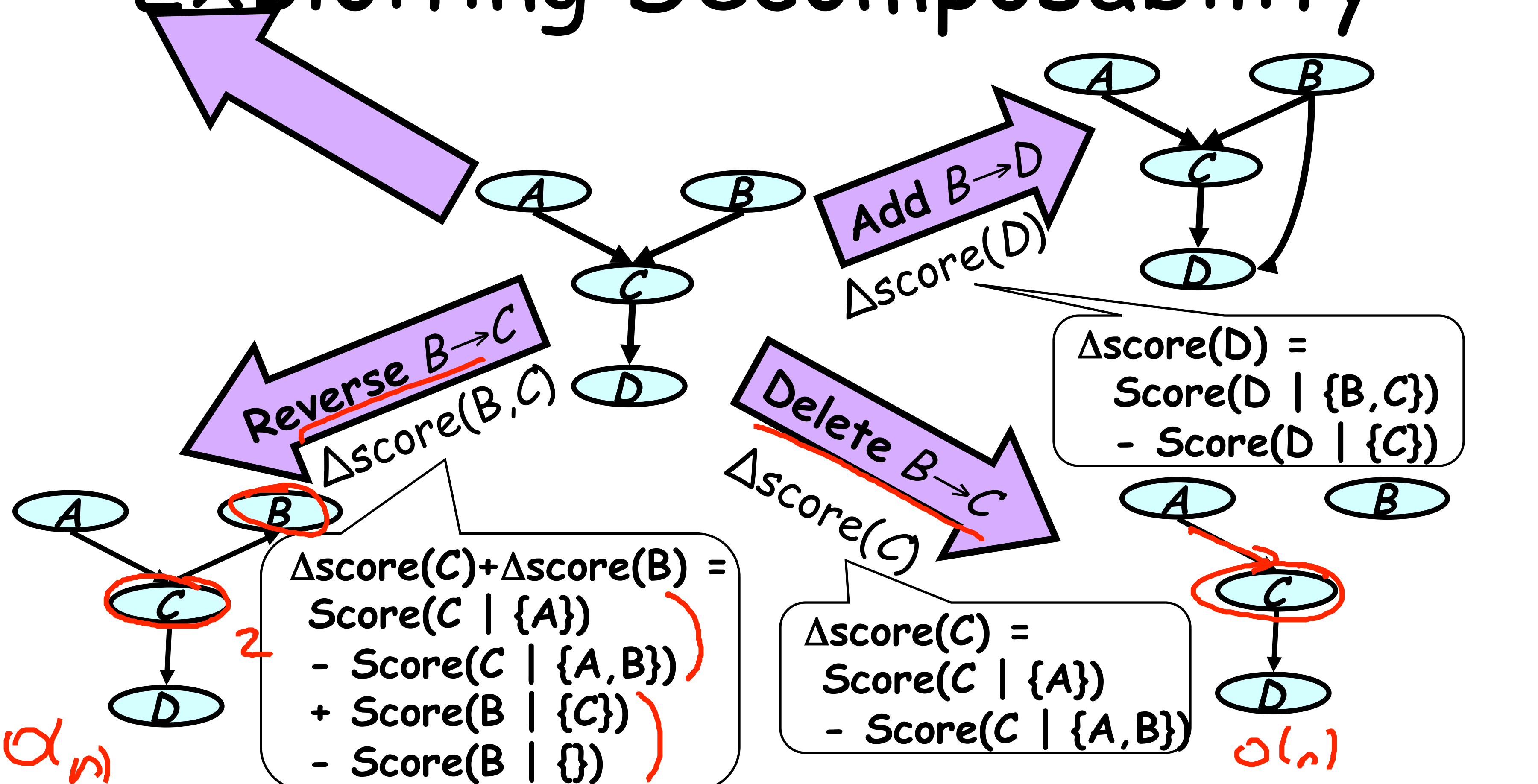
$$\text{score} = \text{Score}(A \mid \{\}) + \text{Score}(B \mid \{\}) + \text{Score}(C \mid \{A,B\}) + \text{Score}(D \mid \{C\})$$

$$\text{score} = \text{Score}(A \mid \{\}) + \text{Score}(B \mid \{\}) + \text{Score}(C \mid \{A,B\}) + \boxed{\text{Score}(D \mid \{B,C\})}$$

O(n) savings

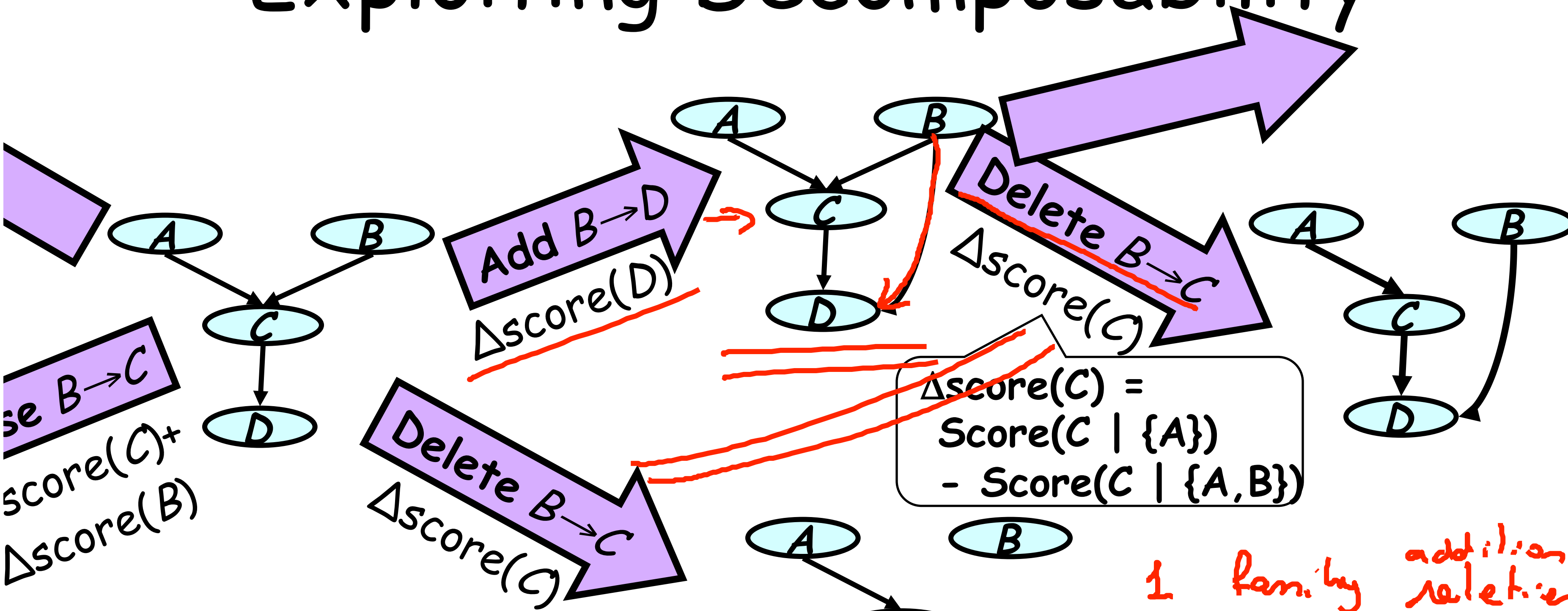
Daphne Koller

Exploiting Decomposability



Daphne Koller

Exploiting Decomposability



To recompute scores,
only need to re-score families
that changed in the last move

1 family addition deletion

2 in reversal

O(n)

Daphne Koller

Computational Cost

- Cost per move
  - Compute O(n) delta-scores damaged by move
  - Each one takes O(M) time

$O(n\,M)$

- Keep priority queue of operators sorted by delta-score – O(n log n)

$O(nM + n\log n)$

More Computational Efficiency

- Reuse and adapt previously computed sufficient statistics
- Restrict in advance the set of operators considered in the search

A, B, C

$M[A,B,C]$

$M[A,B] = \sum_{C} M[A,B,C]$

Summary

- Even heuristic structure search can get expensive for large n
- Can exploit decomposability to get orders of magnitude reduction in cost
- Other tricks are also used for scaling