Limited Discrepancy AND/OR Search and its Application to Optimization Tasks in Graphical Models

Javier Larrosa    Emma Rollón    Rina Dechter

UPC BarcelonaTech (Spain), University of California at Irvine (USA)
larrosa@cs.upc.edu, erollon@cs.upc.edu, dechter@ics.uci.edu

July 6, 2016
Motivation

Graphical Model

- $X$ set of **variables**
- $f_i(X_i)$ set of (local, $X_i \subset X$) **cost functions**.

Min-Sum Problem

$$\min_X F(X) = \sum_i f_i(X_i)$$

**Applications:** Image processing, Natural Language Processing, Bioinformatics, Planning, Resource Allocation, ...
Motivation

**Solving Method:** Depth-first Search

<table>
<thead>
<tr>
<th>Search Space</th>
<th>OR tree</th>
<th>AND/OR tree</th>
<th>AND/OR graph</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size (exp. on)</td>
<td>n. of variables</td>
<td>path width</td>
<td>induced width</td>
</tr>
</tbody>
</table>

- LDS (on the OR tree) very successful **anytime** algorithm (*toulbar2, daoopt*)
- Can we adapt LDS to AND/OR search spaces???
- We show that LDSAO is faster than LDS on the min-sum problem.
The search is guided by a **heuristic** $h(n)$

$h(n)$ is usually good, but **not perfect**

**Advantage:** memory efficient

**Drawback:** early mistakes are fatal
**Limited Discrepancy Search (LDS) [Harvey and Ginsberg, 95]**

1. **Discrepancy**: right turn (going against the heuristic)
2. **Leaf discrepancies**: number of right turns
3. There are \( \binom{n}{k} \) leaves with \( k \) discrepancies
4. **LDS**: Search in increasing order of discrepancies
5. \( k \)-th iteration: visits leaves with \( k \) or less discrepancies

![Diagram of LDS search]

- \( X_1 \)
- \( X_2 \)
- \( X_3 \)
- \( X_4 \)
- Nodes labeled with their respective discrepancies (0 to 4)
Algorithm

Function LDS()
begin
  for $k = 0 \ldots n$ do
    if Probe(root, $k$) then return true
  return false
end

Function Probe(node, $k$)
begin
  if isLeaf(node) then return isGoal(node)
  if $k = 0$ then return Probe(left(node),0)
  else return (Probe(right(node),$k - 1$) or Probe(left(node),$k$))
end
Limited Discrepancy Search (LDS)

1. Successful in a number of domains
2. Several enhancements have been proposed (e.g. ILDS, DBDS,...)
3. In optimization problems (i.e., find best solution) LDS becomes an anytime algorithm
AND/OR search trees [Nilsson, 80]

1. **OR nodes**: decision points
2. **AND nodes**: independent sub-problems
3. Solution tree
4. Depth-first AND/OR Search

![Diagram of AND/OR tree with nodes and decision points]
Limited Discrepancy AND/OR search (LDSAO)

Definition

- Discrepancies of a leaf: right turns after OR nodes
- Discrepancies of a solution tree: maximum among branches

There are $O(n \cdot \binom{h}{k})$ solution trees with $k$ discrepancies

LDSAO: searches solution trees in increasing number of discrepancies
Function ProbeOr(nodeOr, k)
begin
  if k = 0 then return ProbeAnd(left(nodeOr),0)
  return ProbeAnd(right(nodeOr),k−1) or ProbeAnd(left(nodeOr),k)
end

Function ProbeAnd(nodeAnd, k)
begin
  if isLeaf(nodeAnd) then
    return isGoal(nodeAnd)
  for nodeOr ∈ Successors(nodeAnd) do
    if not ProbeOr(nodeOr,k) then
      return false
  return true
end
Search in Graphical Models can be represented with either OR trees or AND/OR trees (exploiting conditional independencies).

\[ F(x_1, \ldots, x_4) = f(x_1, x_2) + f(x_1, x_3) + f(x_1, x_4) + f(x_3, x_4) \]
Therefore, one can use LDS or LDSAO.

**Properties**

- LDSAO iterates faster than LDS (because paths are shorter in the AND/OR tree)
- LDSAO visits more complete assignments than LDS (because of $k$ discrepancies in AND/OR may map to $> k$ discrepancies in OR)
Experimental Results

- Any-time performance of LDS vs LDSAO on the min-sum problem
- Heuristic: static MBE [Kask and Dechter 99; Ihler et al 2011] (i-bound set to 10, 15, 16)
- Six benchmarks (138 instances)
- Time limit: 1 hour
Experimental Results

cat_paths_60_160_0000

UB (in thousands)

LDS
LDSAO

time (sec.)
Future Work

- Overcome the static ordering limitation of AND/OR search (dynamic variable orderings seem to be better)
- Overcome the non-any-time nature of LDSAO during each iteration
- Improvement of search effort (i.e., unbalanced AND/OR trees)
- Add AND/OR to LDS improvements