Decision Diagrams for Discrete Optimization

Willem-Jan van Hoeve
Tepper School of Business
Carnegie Mellon University

based on joint work with
David Bergman, Andre A. Cire, Sam Hoda, and John N. Hooker
Outline

• Motivation and background
  – multi-valued decision diagrams (MDDs)
• Constraint Programming with MDDs
• MDDs as bounding mechanism
  – Relaxations
  – Restrictions
• Conclusions
Decision Diagrams

-→: 0
--→: 1

\[ f(x_1, x_2, x_3) = (\neg x_1 \land \neg x_2 \land \neg x_3) \lor (x_1 \land x_2) \lor (x_2 \land x_3) \]

- Binary Decision Diagrams were introduced to compactly represent Boolean functions [Lee, 1959], [Akers, 1978], [Bryant, 1986]
- Main operation: merge isomorphic subtrees of a given binary decision tree
- MDDs are multi-valued decision diagrams (i.e., for discrete variables)
**Brief background**

- Original application areas: circuit design, verification
- Usually *reduced ordered* BDDs/MDDs are applied
  - fixed variable ordering
  - minimal exact representation
- Recent interest from optimization community
  - cut generation [Becker et al., 2005]
  - 0/1 vertex and facet enumeration [Behle & Eisenbrand, 2007]
  - post-optimality analysis [Hadzic & Hooker, 2006, 2007]
- Interesting variant
  - approximate MDDs
    [H.R. Andersen, T. Hadzic, J.N. Hooker, & P. Tiedemann, CP 2007]
Exact MDDs for discrete optimization

(1) $x_1 + x_2 + x_3 \geq 1$
(2) $x_1 + x_4 + x_5 \geq 1$
(3) $x_2 + x_4 \geq 1$
Exact MDDs for discrete optimization

\[
\begin{align*}
(1) & \quad x_1 + x_2 + x_3 \geq 1 \\
(2) & \quad x_1 + x_4 + x_5 \geq 1 \\
(3) & \quad x_2 + x_4 \geq 1
\end{align*}
\]
Exact MDDs for discrete optimization

(1) \( x_1 + x_2 + x_3 \geq 1 \)
(2) \( x_1 + x_4 + x_5 \geq 1 \)
(3) \( x_2 + x_4 \geq 1 \)
Exact MDDs for discrete optimization

\begin{align*}
(1) & \quad x_1 + x_2 + x_3 \geq 1 \\
(2) & \quad x_1 + x_4 + x_5 \geq 1 \\
(3) & \quad x_2 + x_4 \geq 1
\end{align*}
Exact MDDs for discrete optimization

Each path corresponds to a solution

(1) \( x_1 + x_2 + x_3 \geq 1 \)
(2) \( x_1 + x_4 + x_5 \geq 1 \)
(3) \( x_2 + x_4 \geq 1 \)

\[
\begin{align*}
(1,0,1,1,0)
\end{align*}
\]
Approximate MDDs

- Exact MDDs can be of exponential size in general
- Can we limit the size of the MDD and still have a meaningful representation?
  - Yes, first proposed by Andersen et al. [2007]: Limit the width of the MDD (the maximum number of nodes on any layer)

- This talk: applications to CP and IP
MDDs for Constraint Programming

Motivation

Constraint Programming applies
• systematic search and
• inference techniques
to solve combinatorial problems

Inference mainly takes place through:
• **Filtering** provably inconsistent values from variable domains
• **Propagating** the updated domains to other constraints

\[
\begin{align*}
x_1 & > x_2 \\
x_1 + x_2 &= x_3 \\
# alldifferent(x_1, x_2, x_3, x_4) \\
x_1 & \in \{2\}, \ x_2 \in \{1\}, \ x_3 \in \{3\}, \ x_4 \in \{0\}
\end{align*}
\]

\[
\begin{align*}
x_1 & > x_2 \\
x_1 + x_2 &= x_3 \\
# alldifferent(x_1, x_2, x_3, x_4) \\
x_1 & \in \{2\}, \ x_2 \in \{1\}, \ x_3 \in \{3\}, \ x_4 \in \{0\}
\end{align*}
\]
Drawback of domain propagation

Observations:

- Communication between constraints only via variable domains
- Information can only be expressed as a domain change
- Other (structural) information that may be learned from a constraint is lost: it must be projected onto variable domains
- Potential solution space implicitly defined by Cartesian product of variable domains (very coarse relaxation)

This drawback can be addressed by communicating more expressive information, using MDDs [Andersen et al. 2007]

- Explicit representation of more refined potential solution space
Illustrative Example

\[ \text{AllEqual}(x_1, x_2, \ldots, x_n), \text{ all } x_i \text{ binary} \]

domain representation, size \(2^n\)

MDD representation, size 2
MDD-based constraint programming

- Maintain limited-width MDD
  - Serves as relaxation
  - Typically start with width 1 (initial variable domains)
  - Dynamically adjust MDD, based on constraints

- Constraint Propagation
  - Edge filtering: Remove provably inconsistent edges (those that do not participate in any solution)
  - Node refinement: Split nodes to separate edge information

- Search
  - As in classical CP, but may now be guided by MDD
Specific MDD propagation algorithms

- Linear equalities and inequalities  
  [Hadzic et al., 2008]
  [Hoda et al., 2010]
- \textit{Alldifferent} constraints  
  [Andersen et al., 2007]
- \textit{Element} constraints  
  [Hoda et al., 2010]
- \textit{Among} constraints  
  [Hoda et al., 2010]
- Sequential scheduling constraints  
  [Hoda et al., 2010]
  [Cire & v.H., 2011]
- \textit{Sequence} constraints (combination of \textit{Amongs})  
  [v.H., 2011]
- Generic re-application of existing domain filtering algorithm for any constraint type  
  [Hoda et al., 2010]
Among constraints

- Given a set of variables $X$, and a set of values $S$, a lower bound $l$ and upper bound $u$, 

$$\text{Among}(X, S, l, u) := l \leq \sum_{x \in X} (x \in S) \leq u$$

“among the variables in $X$, at least $l$ and at most $u$ take a value from the set $S$”

- Example: $X$ represents 7-day shift schedule for an employee that must work either 1 or 2 night shifts: 
  $$\text{Among}(X, \{\text{night}\}, 1, 2)$$

- (WLOG assume that $X$ are binary and $S = \{1\}$ )
Example: MDD for Among

Exact MDD for Among($\{x_1, x_2, x_3, x_4\}, \{1\}, 2, 2$)

Size $O(n(u - l))$
MDD Filtering for Among

Goal: Given an arbitrary MDD and an Among constraint, remove all inconsistent edges from the MDD (establish “MDD-consistency”)

Approach:
- Compute path lengths from the root and from the sink to each node in the MDD
- Remove edges that are not on a path with length between lower and upper bound
- Complete (MDD-consistent) version
  - Maintain all path lengths; quadratic time
- Partial version (does not remove all inconsistent edges)
  - Maintain and check bounds (longest and shortest paths); linear time
Node refinement for Among

For each layer in MDD, we first apply edge filter, and then try to refine

- consider incoming edges for each node
- split the node if there exist incoming edges that are not equivalent (w.r.t. path length)

Example:

- We will propagate Among($\{x_1, x_2, x_3, x_4\}, \{1\}, 2, 2$) through a BDD of maximum width 3
Example

$\text{Among}({x_1,x_2,x_3,x_4},{1},2,2)$
Example

Among({x_1, x_2, x_3, x_4}, {1}, 2, 2)
Example

Among\(\{x_1, x_2, x_3, x_4\}, \{1\}, 2, 2\)
Example

Among({x_1, x_2, x_3, x_4}, {1}, 2, 2)
Experiments

• Multiple among constraints
  – 50 binary variables total
  – 5 variables per among constraint, indices chosen from normal distribution with uniform-random mean in [1..50] and stdev 2.5, modulo 50
  – Classes: 5 to 200 among constraints (step 5), 100 instances per class

• Nurse rostering instances (horizon $n$ days)
  – Work 4-5 days per week
  – Max A days every B days
  – Min C days every D days
  – Three problem classes

• Compare width 1 (traditional domains) with increasing widths
Multiple Amongs: Search tree size

width 1 vs 4

width 1 vs 16
Multiple Amongs: Running Time

width 1 vs 4

width 1 vs 16
# Nurse rostering problems

<table>
<thead>
<tr>
<th>Size</th>
<th>Width 1</th>
<th>Width 4</th>
<th>Width 32</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>BT</td>
<td>CPU</td>
<td>BT</td>
</tr>
<tr>
<td>Class 1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>61,225</td>
<td>55.63</td>
<td>8,138</td>
</tr>
<tr>
<td>80</td>
<td>175,175</td>
<td>442.29</td>
<td>5,025</td>
</tr>
<tr>
<td>Class 2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>179,743</td>
<td>173.45</td>
<td>17,923</td>
</tr>
<tr>
<td>80</td>
<td>179,743</td>
<td>459.01</td>
<td>8,747</td>
</tr>
<tr>
<td>Class 3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>91,141</td>
<td>84.43</td>
<td>5,148</td>
</tr>
<tr>
<td>80</td>
<td>882,640</td>
<td>2,391.01</td>
<td>33,379</td>
</tr>
</tbody>
</table>
• MDDs provide substantial advantage over traditional domains for filtering multiple *Among* constraints
  – Strength of MDD can be controlled by the width
  – Wider MDDs yield greater speedups
  – Huge reduction in the amount of backtracking and solution time

• Intensive processing at search nodes can pay off when more structural information is communicated between constraints
Relaxation MDDs


Motivation and outline

- Limited width MDDs provide a (discrete) relaxation to the solution space
- Can we exploit MDDs to obtain bounds for discrete optimization problems?
Handling objective functions

Suppose we have an objective function of the form

$$\min \sum_i f_i(x_i)$$

for arbitrary functions $f_i$.

In an exact MDD, the optimum can be found by a shortest r-s path computation (edge weights are $f_i(x_i)$).
Approach

• Construct the relaxation MDD using a top-down compilation method
• Find shortest path → provides bound B
• Extension to an exact method
  1. Isolate all paths of length B, and verify if any of these paths is feasible*
  2. if not feasible, set $B := B + 1$ and go to 1
  3. otherwise, we found the optimal solution

* Feasibility can be checked using MDD-based CP
Case Study: Independent Set Problem

- Given graph $G = (V, E)$ with vertex weights $w_i$
- Find a subset of vertices $S$ with maximum total weight such that no edge exists between any two vertices in $S$

$$\text{max} \quad \sum_i w_i x_i$$

s.t. \quad x_i + x_j \leq 1 \quad \text{for all } (i,j) \text{ in } E

x_i \text{ binary} \quad \text{for all } i \text{ in } V
Exact top-down compilation

Merge equivalent nodes

• {vertices that can still be included}
Node Merging

Node Merging.

---: 0
---: 1

x_1

x_2

x_3

x_4

x_5

{3,4}

∅

{4}

∅

∅

{5}

{5}

{5}

{5}

{4,5}

{2,3,4,5}

{1,2,3,4,5}

Relaxation.

This procedure generates an exact MDD when the given width is exceeded.
Relaxation MDD

Exact MDD

Relaxation MDD (width ≤ 3)
Relaxation MDD

---: 0  Exact MDD

---: 1

Relaxation MDD (width ≤ 3)
Relaxation MDD

---: 0
----: 1

Exact MDD

\[ (0,0,0,1,0) \]

Relaxation MDD (width ≤ 3)
Relaxation MDD

Exact MDD

Relaxation MDD (width ≤ 3)

\[(1,0,0,0,1)\]
Evaluate Objective Function

---: 0

Exact MDD

---: 1

Relaxation MDD (width ≤ 3)

\[
\max f(x) = 12
\]

\[
\max f(x) = 13
\]
Tightening the Upper Bound

- Value extraction method
  - Given: an MDD relaxation, $M$
  - Given: a valid upper bound, $v$
  - Extract all paths in $M$ that correspond to solutions with objective function value equal to $v$ in the form of another MDD $M|_{z=v}$

- Creating $M|_{z=v}$ can be done efficiently

- Apply MDD-based CP to $M|_{z=v}$ in order to either
  - Decrease $v$ to $v-1$ (if no solution exists)
  - Find a feasible (and optimal) solution
Experimental Results

• Impact of maximum width on strength of bound (and running time)
• Evaluate value extraction method
• Compare MDD bounds to LP bounds

• DIMACS clique instances (unweighted graphs)
Impact of width on relaxation

upper bound

maximum width

brock_200-2 instance
Compare MDD and LP bounds

• CPLEX root node relaxation
  – no primal heuristics, no presolve
  – maximum 5 minutes

• MDD bound – version 1
  – maximum width 100
  – apply value extraction for the same time as CPLEX

• MDD bound – version 2
  – maximum width \( \frac{3,000,000}{n} \) (fill memory)
  – no value extraction
%difference, i.e., $100 \cdot \frac{z_{\text{LP}} - z_{\text{MDD}}}{z_{\text{MDD}}}$
Large MDD versus LP (CPLEX)

CPLEX time: total 10,424s, average 158s
MDD time: total 381s, average 6s
Restriction MDDs
Definition

• Restriction MDDs represent a subset of feasible solutions
  – we require that every r-s path corresponds to a feasible solution
  – but not all solutions need to be represented
• Goal: Use restriction MDDs as a heuristic to find good feasible solutions
Creating Restriction MDDs

Using an exact top-down compilation method, we can create a limited-width restriction MDD by

1. merging nodes, or
2. deleting nodes

while ensuring that no solution is lost
Restriction MDD (width ≤ 3)

---: 0
---: 1

x_1

{x_1, x_2, x_3}

x_2

{x_1, x_2, x_3}

x_3

{x_1, x_2, x_3}

∅ {4} {5} {5} {4,5}
Restriction MDD (width ≤ 3)

Node merging by example
Node merging heuristics

• Random
  – select two nodes \( \{u_1, u_2\} \) uniformly at random

• Objective-driven
  – select two nodes \( \{u_1, u_2\} \) such that
    \[ f(u_1), f(u_2) \leq f(v) \text{ for all nodes } v \neq u_1, u_2 \text{ in the layer} \]

• Similarity
  – select two nodes \( \{u_1, u_2\} \) that are ‘closest’
  – problem dependent (or based on semantics)
Node deletion by example

Restriction MDD (width ≤ 3)

---: 0
—: 1

$\{1,2,3,4,5\}$

$\{3,4\}$
$\{2,3,4,5\}$

$\{3,4\}$
$\{2,3,4,5\}$

$\emptyset$
$\{4\}$
$\{5\}$
$\{3,4,5\}$

$\{4,5\}$
Node deletion heuristics

- **Random**
  - select node $u$ uniformly at random

- **Objective-driven**
  - select node $u$ such that $f(u) \leq f(v)$ for all nodes $v \neq u$ in the layer

- **Information-driven**
  - problem specific
**Experimental Results**

- Comparison to greedy heuristic
  - select vertex $v$ with smallest degree and add it to independent set
  - remove $v$ and its neighbors and repeat

- MDD version 1: maximum width 100
- MDD version 2: maximum width $8,000,000/n$
Greedy versus MDD

%difference, i.e., $100 \cdot \frac{z_{MDD} - z_{Gr}}{z_{Gr}}$

CPU times for Greedy and MDD are similar
Conclusions

- Limited-width MDDs can be a very useful tool for discrete optimization
  - The maximum width provides a natural trade-off between computational efficiency and strength
  - Powerful inference mechanism for constraint propagation
  - Generic discrete relaxation and restriction method for MIP-style problems
- Many open questions
  - MDD variable ordering, interaction with search, formal characterizations, ...