Decision Diagrams for Discrete Optimization

Willem-Jan van Hoeve
Tepper School of Business
Carnegie Mellon University
www.andrew.cmu.edu/user/vanhoeve/mdd/

Acknowledgments:

David Bergman, Andre Cire, Samid Hoda, John Hooker, Brian Kell, Joris Kinable, Ashish Sabharwal, Horst Samulowitz, Vijay Saraswat, Marla Slusky, Tallys Yunes
What can MDDs do for discrete optimization?

- Compact representation of all solutions to a problem
- Limit on size gives approximation
- Control strength of approximation by size limit

MDDs for integer optimization

- MDD relaxations provide upper bounds
- MDD restrictions provide lower bounds
- Incorporation in branch-and-bound can be very effective

MDDs for constraint programming and scheduling

- MDD propagation natural generalization of domain propagation
- Orders of magnitude improvement possible
Binary Decision Diagrams were introduced to compactly represent Boolean functions [Lee, 1959], [Akers, 1978], [Bryant, 1986].

BDD: merge isomorphic subtrees of a given binary decision tree.

MDDs are multi-valued decision diagrams (i.e., for discrete variables).

\[ f(x_1, x_2, x_3) = (\neg x_1 \land \neg x_2 \land \neg x_3) \lor (x_1 \land x_2) \lor (x_2 \land x_3) \]
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
    [Andersen, Hadzic, Hooker & 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

(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

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

Each path corresponds to a solution

\( (1,0,1,1,0) \)
Approximate MDDs

• Exact MDDs can be of exponential size in general

• We can limit the size (width) of the MDD to obtain a relaxation [Andersen et al., 2007]
  – strength is controlled by the width

• Can provide bounds on objective function

• Can also be used for cut generation, constraint propagation, guiding search, ...
MDDs for Integer Optimization

Motivation

• Conventional integer programming relies on branch-and-bound based on continuous LP relaxations
  – Relaxation bounds
  – Feasible solutions
  – Branching

• We propose a novel branch-and-bound algorithm for discrete optimization based on decision diagrams
  – Relaxation bounds – Relaxed BDDs
  – Feasible solutions – Restricted BDDs
  – Branching – Nodes of relaxed BDDs

• Potential benefits: stronger bounds, efficiency, memory requirements, models need not be linear
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 $S$

$$\max \quad \sum_i w_i x_i$$

s.t. \quad \begin{align*}
    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
\end{align*}$$
Exact top-down compilation

---: 0
--: 1

state information: eligible vertices

\[
\begin{align*}
\{3,4\} & \quad \{5\} & \quad \{3,4,5\} \\
\{3,4\} & \quad \{5\} & \quad \{3,4,5\} \\
\emptyset & \quad \{4\} & \quad \{5\} \\
\ & \quad \{5\} & \quad \{4,5\}
\end{align*}
\]

Merge equivalent nodes
Node Merging

Relaxed BDD: The procedure generates an exact BDD when the given width is exceeded

Theorem: This procedure generates an exact BDD, according to Bergman et al., 2012

state information: eligible vertices

[Diagram with nodes and edges connecting variables x1, x2, x3, x4, x5]
Relaxed BDD

Exact BDD

Relaxed BDD (width ≤ 3)
Relaxed BDD

Exact BDD

Relaxed BDD (width ≤ 3)
Relaxed BDD

---: 0
____: 1

Exact BDD

Relaxed BDD (width ≤ 3)

(0,0,0,1,0)
Relaxed BDD

---: 0
---: 1

Exact BDD

Relaxed BDD (width ≤ 3)

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

---: 0
---: 1

Exact BDD

Relaxed BDD (width ≤ 3)

\[ \text{max } f(x) = 12 \]

\[ \text{max } f(x) = 13 \]
Restricted BDD

Restricted MDD (width ≤ 3)

---: 0
—: 1

\begin{itemize}
  \item \(x_1\)
  \item \(x_2\)
  \item \(x_3\)
\end{itemize}

\begin{figure}
\centering
\begin{tikzpicture}
  \node (r) at (0,0) [circle, draw, fill=blue!20] {$r$};
  \node (s1) at (-1,-1) [circle, draw, fill=blue!20] {$\{3,4\}$};
  \node (s2) at (1,-1) [circle, draw, fill=blue!20] {$\{2,3,4,5\}$};
  \node (s3) at (-1,-2) [circle, draw, fill=blue!20] {$\{3,4\}$};
  \node (s4) at (-2,-3.5) [circle, draw, fill=blue!20] {$\emptyset$};
  \node (s5) at (1,-2) [circle, draw, fill=blue!20] {$\{3,4,5\}$};
  \node (s6) at (2,-3.5) [circle, draw, fill=blue!20] {$\{4,5\}$};
  \node (s7) at (1,-1) [circle, draw, fill=blue!20] {$\{5\}$};
  \node (s8) at (-1,-2) [circle, draw, fill=blue!20] {$\{4\}$};

  \draw (r) -- (s1);
  \draw (r) -- (s2);
  \draw (s1) -- (s3);
  \draw (s1) -- (s4);
  \draw (s2) -- (s7);
  \draw (s2) -- (s5);
  \draw (s3) -- (s5);
  \draw (s3) -- (s6);
  \draw (s4) -- (s8);
  \draw (s7) -- (s5);
  \draw (s7) -- (s6);
\end{tikzpicture}
\end{figure}
Variable Ordering

• Order of variables greatly impacts BDD size
  – also influences bound from relaxed BDD (see next)

• Finding ‘optimal ordering’ is NP-hard

• Insights from independent set as case study
  – formal bounds on BDD size
  – measure strength of relaxation w.r.t. ordering
Exact BDD orderings for Paths
Many Random Orderings

Better orderings give stronger bounds

For each random ordering, plot the exact BDD width and the bound from width-10 BDD relaxation
Formal Results for Independent Set

<table>
<thead>
<tr>
<th>Graph Class</th>
<th>Bound on Width</th>
</tr>
</thead>
<tbody>
<tr>
<td>Paths</td>
<td>1</td>
</tr>
<tr>
<td>Cliques</td>
<td>1</td>
</tr>
<tr>
<td>Interval Graphs</td>
<td>n/2</td>
</tr>
<tr>
<td>Trees</td>
<td>n/2</td>
</tr>
<tr>
<td>General Graphs</td>
<td>Fibonacci Numbers</td>
</tr>
</tbody>
</table>

(The proof for general graphs is based on a maximal path decomposition of the graph)
Branch and Bound

Width 3 relaxed decision diagram

Upper Bound = 4
Branch and Bound
Branch and Bound
Branch and Bound
Branch and Bound
Branch and Bound
Branch and Bound
Branch and Bound
Branch and Bound
Branch and Bound

\[
\max \{ 2, 3, 2 \} = 3
\]
• Novel branching scheme
  – Branch on pools of partial solutions
  – Remove symmetry from search
    • Symmetry with respect to feasible completions
  – Can be combined with other techniques
    • Use decision diagrams for branching, and LP for bounds
  – Immediate parallelization
    • Send nodes to different workers, recursive application
    • DDX10 (CPAIOR 2014)
Computational Results

• Compare with IBM ILOG CPLEX
  – State-of-the-art integer programming technology

• Use typical, strong formulations
  – Edge formulation and clique formulation for maximum independent set problem
    • $O(n)$ variables, $O(n^2)$ constraints

• Random Erdös-Rényi G(n,p) graphs and DIMACS Clique graphs
  – Compare end gaps after 1,800 seconds
Random graphs: $n=250$
Random graphs: \( n=500 \)
Random graphs: $n=750$
Random graphs: \( n=1500 \)
DIMACS Graphs: End Gap (1,800s)
Parallelization: BDD vs CPLEX

- $n = 170$, each data point avg over 30 instances
- 1 worker: BDD 1.25 times faster than CPLEX (density 0.29)
- 32 workers: BDD 5.5 times faster than CPLEX (density 0.29)
- BDDs scale to well to (at least) 256 workers
General Approach

• In general, our approach can be applied when problem is formulated as a **dynamic programming model**
  – We can build exact BDD from DP model using top-down compilation scheme (exponential size in general)
  – Note that we do **not** use DP to solve the problem, only to represent it

• Other problem classes considered
  – MAX-CUT, set covering, set packing, MAX 2-SAT, SAT, ...
**MAX-CUT representation**

- Value of a cut \((S, T)\) is
  \[
  \sum_{s, t \mid s \in S, t \in T} w(s, t)
  \]

- Example: cut \(\{1, 2\}, \{3, 4\}\) has value 2

- MAX-CUT: Find a cut with maximum value

- How can we represent this in a BDD?
  - state represents vertices included in \(S\)?
  - we propose a state to represent the *marginal cost* of including vertex in \(S\)
MAX-CUT example BDD

- **State:** $j$th element is additional value of adding vertex $j$ to $S$ (if positive)
• **State:** $j^{th}$ element is additional value of adding vertex $j$ to $S$ (if positive)
Computational Results

- Compare with IBM ILOG CPLEX
- Typical MIP formulation + triangle inequalities
  - $O(n^2)$ variables, $O(n^3)$ constraints
- Benchmark problems
  - $g$ instances
  - Helmberg and Rendl instances, which were taken from Rinaldi’s random graph generator
  - $n$ ranges from 800 to 3000 – very large/difficult problems, mostly open
  - Also compared performance with BiqMac
MIP vs BDD: 60 seconds (n=40)

Number of MCP Instances Solved in 60 Seconds (n=40)
MIP vs BDD: 1,800 seconds (n=40)

Number of MCP Instances Solved in 1800 Seconds (n=40)

- BDD (LEL)
- BDD (FC)
- IP (presolve-off)
- IP (presolve-on)

number solved vs density

0.2 0.4 0.6 0.8 1.0
## BiqMac vs BDD

<table>
<thead>
<tr>
<th>instance</th>
<th>BiqMac</th>
<th></th>
<th>BDD</th>
<th></th>
<th>Best known</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>LB</td>
<td>UB</td>
<td>LB</td>
<td>UB</td>
<td>LB</td>
<td>UB</td>
</tr>
<tr>
<td>g50</td>
<td>5880</td>
<td>5988.18</td>
<td>5880</td>
<td>5899*</td>
<td>5880</td>
<td>5988.18</td>
</tr>
<tr>
<td>g32</td>
<td>1390</td>
<td>1567.65</td>
<td>1410*</td>
<td>1645</td>
<td>1398</td>
<td>1560</td>
</tr>
<tr>
<td>g33</td>
<td>1352</td>
<td>1544.32</td>
<td>1380*</td>
<td>1536*</td>
<td>1376</td>
<td>1537</td>
</tr>
<tr>
<td>g34</td>
<td>1366</td>
<td>1546.70</td>
<td>1376*</td>
<td>1688</td>
<td>1372</td>
<td>1541</td>
</tr>
<tr>
<td>g11</td>
<td>558</td>
<td>629.17</td>
<td>564</td>
<td>567*</td>
<td>564</td>
<td>627</td>
</tr>
<tr>
<td>g12</td>
<td>548</td>
<td>623.88</td>
<td>556</td>
<td>616*</td>
<td>556</td>
<td>621</td>
</tr>
<tr>
<td>g13</td>
<td>578</td>
<td>647.14</td>
<td>580</td>
<td>652</td>
<td>580</td>
<td>645</td>
</tr>
</tbody>
</table>
MDDs for Constraint Programming

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

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

\[
x_1 \in \{1,2\}, \ x_2 \in \{1,2,3\}, \ x_3 \in \{2,3\}
\]

\[
x_1 < x_2 \quad x_2 \in \{2,3\}
\]

**alldifferent**(\(x_1, x_2, x_3\))

\[
x_1 \in \{1\}
\]
**Illustrative Example**

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

\[ x_1 + x_2 + \ldots + x_n \geq n/2 \]

- Domain representation, size \(2^n\)
- MDD representation, size 2
Drawback of domain propagation

- All structural relationships among variables are projected onto the domains
- Potential solution space implicitly defined by Cartesian product of variable domains (very coarse relaxation)

We can communicate more information between constraint using MDDs [Andersen et al. 2007]

- Explicit representation of more refined potential solution space
- Limited width defines relaxed MDD
- Strength is controlled by the imposed width
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]

- *Alldifferent* constraints  
  [Andersen et al., 2007]

- *Element* constraints  
  [Hoda et al., 2010]

- *Among* constraints  
  [Hoda et al., 2010]

- Disjunctive scheduling constraints  
  [Hoda et al., 2010]  
  [Cire & v.H., 2011, 2013]

- *Sequence* constraints (combination of *Amongs*)  
  [Bergman et al., 2013]

- Generic re-application of existing domain filtering algorithm for any constraint type  
  [Hoda et al., 2010]
Case Study: Disjunctive Scheduling
Disjunctive Scheduling

- Sequencing and scheduling of activities on a resource

- Activities
  - Processing time: $p_i$
  - Release time: $r_i$
  - Deadline: $d_i$

- Resource
  - Nonpreemptive
  - Process one activity at a time
Common Side Constraints

• Precedence relations between activities

• Sequence-dependent setup times

• Induced by objective function
  – Makespan
  – Sum of setup times
  – Sum of completion times
  – Tardiness / number of late jobs
  – ...

-...
MDD Representation

- Natural representation as ‘permutation MDD’
- Every solution can be written as a permutation \( \pi \)
  \[ \pi_1, \pi_2, \pi_3, \ldots, \pi_n : \text{activity sequencing in the resource} \]
- Schedule is *implied* by a sequence, e.g.:
  \[ \text{start}_{\pi_i} \geq \text{start}_{\pi_{i-1}} + p_{\pi_{i-1}} \quad i = 2, \ldots, n \]
MDD Representation

Path \{1\} – \{3\} – \{2\}:

\begin{align*}
0 & \leq \text{start}_1 \leq 1 \\
6 & \leq \text{start}_2 \leq 7 \\
3 & \leq \text{start}_3 \leq 5
\end{align*}

<table>
<thead>
<tr>
<th>Act</th>
<th>r_i</th>
<th>p_i</th>
<th>d_i</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>2</td>
<td>9</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>3</td>
<td>8</td>
</tr>
</tbody>
</table>
We can apply several propagation algorithms:

- *Alldifferent* for the permutation structure
- Earliest start time / latest end time
- Precedence relations
• State information at each node $i$
  – labels on all paths: $A_i$
  – labels on some paths: $S_i$
  – earliest starting time: $E_i$
  – latest completion time: $L_i$

• Top down example for arc $(u,v)$
All-different Propagation

- All-paths state: $A_u$
  - Labels belonging to all paths from node $r$ to node $u$
  - $A_u = \{3\}$
  - Thus eliminate $\{3\}$ from $(u,v)$
Some-paths state: $S_u$

- Labels belonging to some path from node $r$ to node $u$
- $S_u = \{1,2,3\}$
- Identification of Hall sets
- Thus eliminate $\{1,2,3\}$ from $(u,v)$
Propagate Earliest Completion Time

- Earliest Completion Time: $E_u$
  - Minimum completion time of all paths from root to node $u$

- Similarly: Latest Completion Time
Propagate Earliest Completion Time

<table>
<thead>
<tr>
<th>Act</th>
<th>r_i</th>
<th>d_i</th>
<th>p_i</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>7</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>8</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>10</td>
<td>3</td>
</tr>
</tbody>
</table>

- \( E_u = 7 \)
- Eliminate 4 from \((u,v)\)
Arc with label $j$ infeasible if $i \ll j$ and $i$ not on some path from $r$

Suppose $4 \ll 5$

- $S_u = \{1, 2, 3\}$
- Since $4$ not in $S_u$, eliminate $5$ from $(u,v)$

Similarly: Bottom-up for $j \ll i$
Theorem: Given the exact MDD $M$, we can deduce all implied activity precedences in polynomial time in the size of $M$

- For a node $u$,
  - $A_{u}^{\downarrow}$: values in all paths from root to $u$
  - $A_{u}^{\uparrow}$: values in all paths from node $u$ to terminal

- Precedence relation $i \ll j$ holds if and only if $(j \not\in A_{u}^{\downarrow})$ or $(i \not\in A_{u}^{\uparrow})$ for all nodes $u$ in $M$

- Same technique applies to relaxed MDD
1. Provide precedence relations from MDD to CP
   - update start/end time variables in CP model
   - other inference techniques may utilize them
   - help to guide search

2. Filter the MDD using precedence relations from other (CP) techniques

3. In context of MIP, these can be added as linear inequalities
Experiments

• MDD propagation implemented in IBM ILOG CPLEX CP Optimizer 12.4 (CPO)
  – State-of-the-art constraint based scheduling solver
  – Uses a portfolio of inference techniques and LP relaxation
  – MDD is added as user-defined propagator
TSP with Time Windows

Dumas/Ascheuer instances
- 20-60 jobs
- lex search
- MDD width: 16
Total Tardiness Results

- **MDD-128**
- **MDD-64**
- **MDD-32**
- **MDD-16**
- **CPO**

**Total Tardiness**

**Total Weighted Tardiness**
## Sequential Ordering Problem (TSPLIB)

<table>
<thead>
<tr>
<th>instance</th>
<th>vertices</th>
<th>bounds</th>
<th>CPO</th>
<th>CPO+MDD, width 2048</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>best</td>
<td>time (s)</td>
</tr>
<tr>
<td>br17.10</td>
<td>17</td>
<td>55</td>
<td>55</td>
<td>0.01</td>
</tr>
<tr>
<td>br17.12</td>
<td>17</td>
<td>55</td>
<td>55</td>
<td>0.01</td>
</tr>
<tr>
<td>ESC07</td>
<td>7</td>
<td>2125</td>
<td>2125</td>
<td>0.01</td>
</tr>
<tr>
<td>ESC25</td>
<td>25</td>
<td>1681</td>
<td>1681</td>
<td>TL</td>
</tr>
<tr>
<td>p43.1</td>
<td>43</td>
<td>28140</td>
<td>28205</td>
<td>TL</td>
</tr>
<tr>
<td>p43.2</td>
<td>43</td>
<td>[28175, 28480]</td>
<td>28545</td>
<td>TL</td>
</tr>
<tr>
<td>p43.3</td>
<td>43</td>
<td>[28366, 28835]</td>
<td>28930</td>
<td>TL</td>
</tr>
<tr>
<td>p43.4</td>
<td>43</td>
<td>83005</td>
<td>83615</td>
<td>TL</td>
</tr>
<tr>
<td>ry48p.1</td>
<td>48</td>
<td>[15220, 15805]</td>
<td>18209</td>
<td>TL</td>
</tr>
<tr>
<td>ry48p.2</td>
<td>48</td>
<td>[15524, 16666]</td>
<td>18649</td>
<td>TL</td>
</tr>
<tr>
<td>ry48p.3</td>
<td>48</td>
<td>[18156, 19894]</td>
<td>23268</td>
<td>TL</td>
</tr>
<tr>
<td>ry48p.4</td>
<td>48</td>
<td>[29967, 31446]</td>
<td>34502</td>
<td>TL</td>
</tr>
<tr>
<td>ft53.1</td>
<td>53</td>
<td>[7438, 7531]</td>
<td>9716</td>
<td>TL</td>
</tr>
<tr>
<td>ft53.2</td>
<td>53</td>
<td>[7630, 8026]</td>
<td>11669</td>
<td>TL</td>
</tr>
<tr>
<td>ft53.3</td>
<td>53</td>
<td>[9473, 10262]</td>
<td>12343</td>
<td>TL</td>
</tr>
<tr>
<td>ft53.4</td>
<td>53</td>
<td>14425</td>
<td>16018</td>
<td>TL</td>
</tr>
</tbody>
</table>

* solved for the first time
Summary

What can MDDs do for discrete optimization?

- *Compact representation* of all solutions to a problem
- Limit on size gives *approximation*
- Control strength of approximation by size limit

MDDs for integer optimization

- MDD *relaxations* provide upper bounds
- MDD *restrictions* provide lower bounds
- Incorporation in branch-and-bound can be very effective

MDDs for constraint programming and scheduling

- MDD propagation natural generalization of domain propagation
- Orders of magnitude improvement possible