Decision Diagrams for Constraint Programming

Part 3

Willem-Jan van Hoeve

Tepper School of Business
Carnegie Mellon University

www.andrew.cmu.edu/user/vanhoeve/mdd/
Plan

What can MDDs do for Combinatorial Optimization?

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

MDDs for Constraint Programming and Scheduling

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

MDDs for Discrete Optimization

- MDD relaxations provide upper bounds
- MDD restrictions provide lower bounds
- New branch-and-bound scheme

Many Opportunities: integrated methods, theory, applications,...
MDDs for Discrete Optimization
References


See [http://www.andrew.cmu.edu/user/vanhoeve/mdd/](http://www.andrew.cmu.edu/user/vanhoeve/mdd/)
Motivation

- Conventional integer programming relies on branch-and-bound based on continuous LP relaxations
  - Relaxation bounds
  - Feasible solutions
  - Branching
- We investigate a 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 $S$

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

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

state information: eligible vertices

Merge equivalent nodes
Node Merging

Theorem: This procedure generates a reduced exact BDD

[Bergman et al., 2012]

Relaxed BDD: merge non-equivalent nodes when the given width is exceeded
Relaxed BDD

Exact BDD

Relaxed BDD (width ≤ 3)
Relaxed BDD

---: 0
----: 1

Exact BDD

Relaxed BDD (width ≤ 3)
Relaxed BDD

Exact BDD

Relaxed BDD (width \leq 3)

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

---: 0
---: 1

Exact BDD

- \( r \)

Relaxed BDD (width \( \leq 3 \))

- \( r \)

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

\begin{align*}
\text{Exact BDD} & \\
\text{Relaxed BDD (width \leq 3)} &
\end{align*}

---: 0
\begin{itemize}
\item \text{Exact BDD}
\item \text{Relaxed BDD (width \leq 3)}
\end{itemize}

\begin{align*}
\max f(x) &= 12 \\
\max f(x) &= 13
\end{align*}
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
Exact BDD orderings for Paths
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>Clique</td>
<td>1</td>
</tr>
<tr>
<td>Interval Graphs</td>
<td>1</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)

*INFORMS J. Computing (2014)*
Many Random Orderings

For each random ordering, plot the exact BDD width and the bound from width-10 BDD relaxation.

Better orderings give stronger bounds.
Variable ordering heuristics

- Several possibilities
  - choose vertex at random
  - choose vertex that appears in fewest states in current layer
  - choose vertex according to maximal path decomposition

- Each data point is average over 20 instances
- For random, line segment indicates range over 5 instances
Quality of the bound in practice

• Benchmarks
  – Random Erdös-Rényi $G(n,p)$ graphs
  – DIMACS clique graphs (87 instances)
  – Compare with CPLEX 12.5
    (standard MIP model and clique cover model)
Bounds in practice

random graphs (n=500)
Bounds in practice

random graphs (n=1500)
Restricted BDDs

- Relaxed BDDs find upper bounds for independent set problem
- Can we use BDDs to find lower bounds as well (i.e., good feasible solutions)?
- Restricted BDDs represent a subset of feasible solutions
  - we require that every r-t path corresponds to a feasible solution
  - but not all solutions need to be represented
- Goal: Use restricted BDDs as a heuristic to find good feasible solutions
Creating Restricted BDDs

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

1. merging nodes, or
2. deleting nodes

while ensuring that no non-solutions are introduced
Node merging by example

Restricted BDD (width ≤ 3)
Node merging by example

Restricted BDD (width \( \leq 3 \))

---: 0
—: 1

\[ x_1 \]

\[ x_2 \]

\[ x_3 \]
Node deletion by example

Restricted MDD (width ≤ 3)

In practice, node deletion superior to node merging (similar or better bounds, but much faster)
Experimental Evaluation

- Compare with Integer Programming (CPLEX)
  - LP relaxation + cutting planes
  - Root node solution
- DIMACS instance set
- Restricted BDDs with varying maximum width
Each data point is geometric mean over 20 instances.
BDD-based Branch and Bound

• Search in conventional branch and bound
  – branch on variable \( (x \leq v \text{ or } x \geq v) \)
  – branch on constraints \( (act_1 << act_2 \text{ or } act_2 << act_1) \)

• We will ‘branch’ on states in the BDD instead

at each search state, can evaluate BDD lower and upper bound
Relaxed BDD (width ≤ 3)

Bound = 13

Last Exact Layer

Bound = 13
Node Queue

Relaxed BDD (width ≤ 3)

Q:

x₁

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

x₂

{3,4} {2,3,4,5}

{3,4} 5 {5} 4 0 {3,4,5}

x₃

x₄

x₅

Bound = 13

Last Exact Layer

Upper bound = 13

Bound = 13
Node Queue

Q: \{3,4\} 5 \{5\} 4 0 \{3,4,5\}

Exact solution: 11

Upper bound = 13
Lower bound = 11
Node Queue

Q:  
{5}  4  0  {3,4,5}  

Upper bound = 13
Lower bound = 12

Exact solution: 12
Node Queue

Q:

0 \{3, 4, 5\}

Exact solution: 10

Upper bound = 13
Lower bound = 12
Node Queue

Q:

Optimal Solution: 12
New Branching Scheme

• 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
    • Define CP search with MDD inside global constraint
  – Immediate parallelization
    • Send nodes to different workers, recursive application
    • DDX10 (CPAIOR 2014)
Computational Results: DIMACS

Graph showing the number solved over time for different algorithms: BDD, CPLEX-TIGHT, CPLEX.
DIMACS Graphs: End Gap (1,800s)
**Parallelization: Centralized Architecture**

**Master** maintains a **pool** of BDD nodes to process
- nodes with larger upper bound have higher priority

**Workers** receive BDD nodes, generate *restricted & relaxed* BDDs, and send new BDD nodes and bounds to master
- they also maintain a **local pool** of nodes
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
• 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, ...

INFORMS J. Computing (to appear)
J. Heuristics (2014)
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 \(T\)
• **State:** $j^{th}$ element is additional value of adding vertex $j$ to $T$ (if positive)
• **State:** $j^{th}$ element is additional value of adding vertex $j$ to $T$ (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)
MIP vs BDD: 1,800 seconds (n=40)
## BiqMac vs BDD

<table>
<thead>
<tr>
<th>instance</th>
<th>BiqMac</th>
<th></th>
<th></th>
<th>BDD</th>
<th></th>
<th></th>
<th>Best known</th>
<th></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>
<td>LB</td>
<td>UB</td>
<td></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>
<td>g50</td>
<td>5988.18</td>
<td></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>
<td>g32</td>
<td>1560</td>
<td></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>
<td>g33</td>
<td>1537</td>
<td></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>
<td>g34</td>
<td>1541</td>
<td></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>
<td>g11</td>
<td>627</td>
<td></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>
<td>g12</td>
<td>621</td>
<td></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>
<td>g13</td>
<td>645</td>
<td></td>
</tr>
</tbody>
</table>
Summary

What can MDDs do for Combinatorial Optimization?

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

MDDs for Constraint Programming and Scheduling

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

MDDs for Discrete Optimization

• MDD *relaxations* provide upper bounds
• MDD *restrictions* provide lower bounds
• New branch-and-bound scheme

Many Opportunities: integrated methods, theory, applications,...
Opportunities / Open issues

• Extend application to CP
  – Which other global constraints are suitable? (Cumulative?)
  – Can we develop search heuristics based on the MDD? (yes)
  – Can we more efficiently store and manipulate approximate MDDs? (Implementation issues)
  – Can we obtain a tighter integration with CP domains?

• MDD technology
  – How should we handle constraints that partially overlap on the variables? Build one large MDD or have partial MDDs communicate?
  – How do we communicate information between MDDs on different subproblems (e.g., jobshop)? (Lagrangians)
Opportunities / Open issues (cont’d)

• Formal characterization
  – Can MDDs be used to identify tractable classes of CSPs?
  – Can we identify classes of global constraints for which establishing MDD consistency is hard/easy?
  – Can MDDs be used to prove approximation guarantees?
  – Can we exploit a connection between MDDs and tight LP representations of the solution space?

• Optimization
  – Relaxed/restricted MDDs can provide bounds for any nonlinear (separable) objective function. Demonstrate the performance on an actual application.
Opportunities / Open issues (cont’d)

• Beyond classical CP
  – How can MDDs be helpful in presence of uncertainty? E.g., can we use approximate MDDs to represent policy trees for stochastic optimization?
  – Can we utilize limited-width BDDs for SAT? (yes)
  – Can MDDs help generate nogoods, e.g., in lazy clause generation? (yes)
  – Tighter integration of MDDs in MIP solvers? (yes)

• Applications
  – So far we have looked mostly at generic problems. Are there specific application areas for which MDDs work particularly well? (Bioinformatics?)
7. Consider the following CSP

\[ 4x_1 + 2x_2 + x_3 + x_4 + 2x_5 + 4x_6 = 7 \]

\[ x_1, x_2, ..., x_6 \in \{0,1\} \]

a) Draw an exact BDD for this problem using the variable ordering \( x_1, x_2, x_3, x_4, x_5, x_6 \)

b) Draw an exact BDD for this problem using the variable ordering \( x_1, x_6, x_2, x_5, x_3, x_4 \)

c) Which of the two orderings yields the smallest width?
8. Consider the following set covering instance:

\[
\begin{align*}
\text{minimize} & \quad 3x_1 + 2x_2 + x_3 + 4x_4 + 2x_5 \\
\text{s.t.} & \quad x_1 + x_2 + x_3 \geq 1 \\
& \quad x_1 + x_4 + x_5 \geq 1 \\
& \quad x_2 + x_4 \geq 1
\end{align*}
\]

What state representation would you use to define the BDD? Construct a restricted BDD with maximum width 3. Does it yield the optimal solution?