# Decision Diagrams for Optimization 

## Andre Augusto Cire

Dept. of Management, University of Toronto Scarborough Rotman School of Management

MIP 2015, Chicago, June 2015

## Collaborators



David Bergman
University of Connecticut


Joris Kinable
Carnegie Mellon University


Christian Tjandraatmadja Carnegie Mellon University


## Our Main Research Goal

Investigate the use of Decision Diagrams for solving discrete optimization problems

## Contributions so far

- New relaxation/bounding technique
- Bounds can be superior to state-of-the-art methods in certain problems
- Generic primal heuristic
- Scales to large-scale problems
- Inference techniques
- New types of cuts for MIPs and other optimization technologies
- Novel complete solution technique
- Solved open instances from classical benchmarks
- Parallel method that scales almost linearly with number of processors


## Decision Diagrams

- Graphical representation of Boolean functions

$$
\begin{array}{r}
f(x)=\left(x_{1} \Leftrightarrow x_{2}\right) \wedge\left(x_{3} \Leftrightarrow x_{4}\right) \\
\begin{array}{cccc|c}
x_{1} & x_{2} & x_{3} & x_{4} & \mathrm{f}(\mathrm{x}) \\
\hline 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 1 & 0 \\
0 & 1 & 1 & 0 & 0 \\
0 & 0 & 1 & 1 & 1 \\
\cdots & \ldots & \ldots & \ldots & \cdots
\end{array}
\end{array}
$$

## Decision Diagrams

- Graphical representation of Boolean functions

$$
\begin{array}{r}
f(x)=\left(x_{1} \Leftrightarrow x_{2}\right) \wedge\left(x_{3} \Leftrightarrow x_{4}\right) \\
\begin{array}{cccc|c}
x_{1} & x_{2} & x_{3} & x_{4} & f(x) \\
\hline 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 1 & 0 \\
0 & 1 & 1 & 0 & 0 \\
0 & 0 & 1 & 1 & 1 \\
\ldots & \ldots & \ldots & \ldots & \ldots
\end{array}
\end{array}
$$



## Decision Diagrams

- Graphical representation of Boolean functions

$$
\begin{array}{r}
f(x)=\left(x_{1} \Leftrightarrow x_{2}\right) \wedge\left(x_{3} \Leftrightarrow x_{4}\right) \\
\begin{array}{cccc|c}
x_{1} & x_{2} & x_{3} & x_{4} & f(x) \\
\hline 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 1 & 0 \\
0 & 1 & 1 & 0 & 0 \\
0 & 0 & 1 & 1 & 1 \\
\ldots & \ldots & \ldots & \ldots & \ldots
\end{array}
\end{array}
$$



## Decision Diagrams

- Graphical representation of Boolean functions

$$
\begin{array}{r}
f(x)=\left(x_{1} \Leftrightarrow x_{2}\right) \wedge\left(x_{3} \Leftrightarrow x_{4}\right) \\
\begin{array}{cccc|c}
\mathrm{x}_{1} & \mathrm{x}_{2} & \mathrm{x}_{3} & \mathrm{x}_{4} & \mathrm{f}(\mathrm{x}) \\
\hline 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 1 & 0 \\
0 & 1 & 1 & 0 & 0 \\
0 & 0 & 1 & 1 & 1 \\
\ldots & \ldots & \cdots & \cdots & \ldots
\end{array}
\end{array}
$$



## Decision Diagrams

- Graphical representation of Boolean functions

$$
f(x)=\left(x_{1} \Leftrightarrow x_{2}\right) \wedge\left(x_{3} \Leftrightarrow x_{4}\right)
$$

- Dual role
- Computational model
- Graphical encoding
- [Lee'59, Akers'78, Bryant'86]



## Decision Diagrams

- Application in several areas
- Circuit design
- Formal verification
- Symbolic model checking
- Our focus: Optimization
- Literals $\rightarrow$ variables
- Arcs $\rightarrow$ value assignments
- Paths encode solutions



## Decision Diagrams

$\max 2 x_{1}+x_{2}-4 x_{3}+x_{4}$ subject to

$$
\begin{aligned}
& x_{1}-x_{2}=0 \\
& x_{3}-x_{4}=0 \\
& x_{1}, x_{2}, x_{3}, x_{4} \in\{0,1\}
\end{aligned}
$$



## Decision Diagrams

## $\max 2 x_{1}+x_{2}-4 x_{3}+x_{4}$

subject to

$$
\begin{aligned}
& x_{1}-x_{2}=0 \\
& x_{3}-x_{4}=0 \\
& x_{1}, x_{2}, x_{3}, x_{4} \in\{0,1\}
\end{aligned}
$$



## Decision Diagrams

## $\max 2 x_{1}+x_{2}-4 x_{3}+x_{4}$

subject to

$$
\begin{aligned}
& x_{1}-x_{2}=0 \\
& x_{3}-x_{4}=0 \\
& x_{1}, x_{2}, x_{3}, x_{4} \in\{0,1\}
\end{aligned}
$$



## Decision Diagrams

## $\max 2 x_{1}+x_{2}-4 x_{3}+x_{4}$

subject to

$$
\begin{aligned}
& x_{1}-x_{2}=0 \\
& x_{3}-x_{4}=0 \\
& x_{1}, x_{2}, x_{3}, x_{4} \in\{0,1\}
\end{aligned}
$$



## Decision Diagrams

$\max 2 x_{1}+x_{2}-4 x_{3}+x_{4}$ subject to

$$
\begin{aligned}
& x_{1}-x_{2}=0 \\
& x_{3}-x_{4}=0 \\
& x_{1}, x_{2}, x_{3}, x_{4} \in\{0,1\}
\end{aligned}
$$

- Maximizing a linear (or separable) function:
- Arc lengths: contribution to the objective
- Longest path: optimal solution



## Decision Diagrams

- Uses of this framework:
- Solution counting (Lobbing'96)
- Large-scale network flows (Hachtel et al’97)
- Postoptimality analysis (Hadzic \& Hooker’08)
- Few others, typically domain-specific.
- Our goal: exploit the use of decision diagrams in generic optimization methods

E.g., Linear Programming Relaxation


## Modeling <br> Framework

E.g., Linear Inequalities
E.g., valid cuts

## Generic Optimization Techniques

E.g., Mixed-integer Programming
E.g., Feasibility Pump

## Search

E.g., Branch and bound
E.g., Linear Programming Relaxation

## Modeling <br> Framework

E.g., Linear Inequalities
E.g., valid cuts

## Generic Optimization Techniques

E.g., Mixed-integer Programming
E.g., Feasibility Pump

## Search

E.g., Branch and bound

## Modeling Framework

Ex.: Maximum independent set problem


## Modeling Framework

Ex.: Maximum independent set problem



## Modeling Framework

## Ex.: Maximum independent set problem



- Integer Programming Formulation:
$\max 3 x_{1}+4 x_{2}+2 x_{3}+2 x_{4}+7 x_{5}$ subject to

$$
\begin{aligned}
& x_{1}+x_{2} \leq 1 \\
& x_{1}+x_{3} \leq 1 \\
& x_{2}+x_{3} \leq 1 \\
& x_{3}+x_{4} \leq 1 \\
& x_{4}+x_{5} \leq 1 \\
& x_{1}, x_{2}, x_{3}, x_{4}, x_{5} \in\{0,1\}
\end{aligned}
$$

## Modeling Framework

Ex.: Maximum independent set problem


- Our model: Dynamic Programming
- Exploit recursiveness
- Model is formulated through states
- Decisions (or controls): define state transitions
- Decision diagram: State-Transition Graph
- Nodes corresponds to states
- Arcs are state transitions
- Arc weights are transition costs


## Modeling Framework

- DP model for the maximum independent set:
- State: vertices that can be added to an independent set (eligible vertices)
- Decision: select or not a vertex i from the eligibility set
- Formal model:

$$
\begin{gathered}
V_{i}(S)= \begin{cases}\max \left\{V_{i-1}(S \backslash\{i\}), V_{i-1}(S \backslash N(i))+1\right\}, & i \in S \\
V_{i-1}(S \backslash N(i)), & o . w .\end{cases} \\
V_{i}(\varnothing)=0, \quad i=1, \ldots, 5
\end{gathered}
$$

## Maximum Independent Set Problem

$$
\text { (r) }\left\{v_{1}, v_{2}, v_{3} v_{4}, v_{5}\right\}
$$



State: set of eligible vertices

__ include

-     -         -             - • exclude


## Maximum Independent Set Problem



State: set of eligible vertices
$\qquad$
-----. exclude

## Maximum Independent Set Problem



State: set of eligible vertices
$\qquad$
-----. exclude

## Maximum Independent Set Problem



State: set of eligible vertices

__ include

-     -         -             - • exclude


## Maximum Independent Set Problem



State: set of eligible vertices

_ include

-     -         -             - • exclude


## Maximum Independent Set Problem



State: set of eligible vertices


$$
\ldots \text { include }
$$

## Maximum Independent Set Problem



State: set of eligible vertices

___ include

-----. exclude

## Maximum Independent Set Problem



State: set of eligible vertices

$$
\overline{--e^{--}} \text {include }
$$



## Other Example: Maximum Cut Problem



$v_{1}$
right

-     -         -             - • left


## Other Example: Maximum Cut Problem


right

-     -         -             - • left


## Some quick observations

- Variable ordering plays a big role on size
- Closely connected to treewidth and bandwidth
- Independent Set: polynomial for certain classes of graphs
- TSP: parameterized-size depending on precendence relations
- In general, decision diagrams grow exponentially large
- Proof: Extended Formulations for the Independent Set Problem
E.g., Linear Programming Relaxation


## Generic Optimization Techniques

E.g., Mixed-integer Programming

## Relaxed Decision Diagrams

- In practice, we cannot work with exact diagrams
- Alternative: limit the size to approximate the feasible space
- Parameter on the width of the diagram
- Relaxed Decision Diagrams: Over-approximation
- Introduced by [Andersen et al’07]


## Relaxed Decision Diagrams



## Relaxed Decision Diagrams


$x=(0,1,0,0,1)$
Solution value $=11$
___ include


## Relaxed Decision Diagrams


$x=(0,1,1,0,1)$
Upper bound = 13
___ include
----- exclude


## Compiling Relaxed Decision Diagrams

- Model is augmented with a state agregation operator
- Recipe on how to merge nodes so that no feasible solution is lost
- $V_{i}(S)= \begin{cases}\max \left\{V_{i-1}\left(S \backslash\{i j), V_{i-1}(S \backslash N(i))+1\right\},\right. & i \in S \\ V_{i-1}(S \backslash N(i)), & \text { o.w. }\end{cases}$
$V_{i}(\varnothing)=0, \quad i=1, \ldots, 5$
- $\Delta\left(S_{1}, S_{2}\right)=S_{1} \cup S_{2}$


## Building Relaxed Decision Diagrams



Max Width $=2$

$$
\overline{\ldots-e^{-} \cdot} \text { include }
$$

## Building Relaxed Decision Diagrams



Max Width $=2$

$$
\ldots \text { include }
$$

## Building Relaxed Decision Diagrams



Max Width $=2$

$$
\overline{--n^{--}} \text {include }
$$

## Building Relaxed Decision Diagrams



Max Width $=2$

$$
\overline{--n^{--}} \text {include }
$$

## Building Relaxed Decision Diagrams



Max Width $=2$

$$
\overline{-\overbrace{}^{---}} \text {include }
$$

## Building Relaxed Decision Diagrams



Max Width = 2

$$
\overline{-\_ \text {- }} \text { include }
$$



## Relaxation Bound: Maximum Independent Set



## Strengthening Diagram Relaxations

- Filtering operations
- "Redundant" constraints
- Additive Bounding
- Incorporate dual information from LP relaxations
-DD-Based Lagrangian Relaxations


## Strengthening Diagram Relaxations

- Filtering operations
- "Redundant" constraints
- Additive Bounding
- Incorporate dual information from LP relaxations
-DD-Based Lagrangian Relaxations


## DD-Based Lagrangian Relaxation



- We are solving
max $f(x)$
subject to $x \in \operatorname{Relaxed} D D$


## DD-Based Lagrangian Relaxation



- We are solving $\max f(x)$ subject to $x \in$ RelaxedDD
$x=(0,1,1,0,1)$
Upper bound = 13
—_ include
-----. exclude


## DD-Based Lagrangian Relaxation



- Let $\mathrm{A}, \mathrm{b}$ be such that:
$A x \leq b$ for any feasible $x$
- DD-Based Lagrangian:
$\max f(x)+\lambda(b-A x)$ subject to
$x \in$ RelaxedDD
- Gives an upper bound for any $\lambda \geq 0$


## DD-Based Lagrangian Relaxation



- Solution (0,1,1,0,1) violates constraint

$$
x_{2}+x_{3} \leq 1
$$

- We penalize with term

$$
+\lambda\left(1-x_{2}-x_{3}\right)
$$

by simply changing the cost structure of the DD
-_ include
-----. exclude

## DD-Based Lagrangian Relaxation



- Solution (0,1,1,0,1) violates constraint

$$
x_{2}+x_{3} \leq 1
$$

- We penalize with term

$$
+\lambda\left(1-x_{2}-x_{3}\right)
$$

by simply changing the cost structure of the DD

## DD-Based Lagrangian Relaxation



- Solution (0,1,1,0,1) violates constraint

$$
x_{2}+x_{3} \leq 1
$$

- We penalize with term

$$
+\lambda\left(1-x_{2}-x_{3}\right)
$$

by simply changing the cost structure of the DD

## DD-Based Lagrangian Relaxation



- Solution (0,1,1,0,1) violates constraint

$$
x_{2}+x_{3} \leq 1
$$

- We penalize with term

$$
+\lambda\left(1-x_{2}-x_{3}\right)
$$

by simply changing the cost structure of the DD
—— include
--- - . exclude

## Computational Analysis

- Incorporated into IBM ILOG CP Optimizer (CPO)
- State-of-the-art constraint-based scheduling solver
- Uses a portfolio of inference techniques and LP relaxations


## TSP with Time Windows



## DD-Based Lagrangian

Solution Times (secs)


## Other Results

- Asymmetric TSP with Precedence Constraints
- Closed 3 TSPLIB open instances
- Easy modeling for certain problems
- Example: Time-Dependent TSPs
E.g., Linear Programming Relaxation


## Primal

 HeuristicsE.g., Feasibility Pump

## Generic Optimization Techniques

E.g., Mixed-integer Programming

## Restricted Decision Diagrams

- Under-approximation of the feasible set


Max Width = 2
include

-     -         -             - . exclude



## Restricted Decision Diagrams

- Under-approximation of the feasible set

$(1,0,0,0,1) \rightarrow$ Lower bound $=10$



## Primal Bound: Set Covering



E.g., Linear Programming Relaxation

## Primal

 HeuristicsE.g., Feasibility Pump

## Generic Optimization Techniques

E.g., Mixed-integer Programming

Inference
E.g., valid cuts

## Quick Notes on Inference

- Cut generation for MIPs
- Several techniques from Behle’07
- Recent: Polar set cuts from Relaxed Decision Diagrams
- Talk to Christian Tjandraatmadja! (poster yesterday!)
- Highly-structured Cuts
- Precedence relations that must hold in scheduling problems
- We are still exploring notion of decision diagram separation
- Cire \& Hooker, ISAIM 2014
E.g., Linear Programming Relaxation


## Modeling <br> Framework

E.g., Linear Inequalities
E.g., valid cuts

## Generic Optimization Techniques

E.g., Mixed-integer Programming
E.g., Feasibility Pump

## Search

E.g., Branch and bound

## Exact Method

- Novel decision diagram branch-and-bound scheme
- Relaxed diagrams play the role of the LP relaxation
- Restricted diagrams are used as primal heuristics
- Branching is done on the nodes of the diagram
- Branching on pools of partial solutions
- Eliminate search symmetry


Relaxed



Relaxed


Up to a certain layer, the diagrams are the same (i.e., one layer before you start forcefully merging)


Relaxed


Thus, an optimum solution must necessarily pass through one of these nodes




Relaxed

$\begin{array}{ccc}\left\{v_{3}, v_{4}, v_{5}, v_{6}\right\} \\ 0 & \left\{v_{3}, v_{4}, v_{5}\right\} & \left\{v_{3}, v_{6}\right\} \\ 1 & 1\end{array}$



Relaxed



Relaxed


Explore each separately, saving the best solution/bound found

## Maximum Cut

- Reduced certain optimality gaps

| instance | old \% gap | new \% gap | \% reduction |
| :---: | :---: | :---: | :---: |
| g11 | 11.17 | 0.53 | 95.24 |
| g50 | 1.84 | 0.32 | 82.44 |
| g32 | 11.59 | 10.64 | 8.20 |
| g12 | 11.69 | 10.79 | 7.69 |
| g33 | 11.70 | 11.30 | 3.39 |
| g34 | 12.32 | 11.99 | 2.65 |

## Maximum Independent Set: 500 variables



## Maximum Independent Set: 1500 variables



## Parallel Search with Decision Diagrams

- New branching scheme is very suitable to parallelism
- Idea: explore DP States in different cores
- Relatively little information needs to be shared
- Most of the computational work involves computing relaxations/restrictions, done locally by each computer core
- Easier to distribute load
- Joint work with Horst Samulowitz, Vijay Saraswat (IBM Research), and Ashish Sabharwal (Allen Inst.)


## Parallel Search: Why bother?

- Current technology
- Integer Programming
- Gurobi: Average speedup factor (Gu, 2013)
- 1.7x on 5 cores
- 1.8x on 25 cores
- CPLEX (Mittleman, 2009)
- 1.67x on 4 cores
- SAT
- 2013 SAT competition
- 8x on 32 cores
- Constraint Programming
- Only focus on infeasible instances/finding all solutions


## Parallel Search with Decision Diagrams

| C125.9 | 1 core | 4 cores | 16 cores | 64 cores | 256 cores |
| :--- | :---: | :---: | :---: | :---: | :---: |
| Time to solve (s) | 1100.91 | 277.07 | 70.74 | 19.53 | 8.07 |
| Speedup | - | $3.97 x$ | $15.56 x$ | $56.37 x$ | $136.42 x$ |




CPLEX


## Thank you!

Decision Diagram Page:<br>http://www.andrew.cmu.edu/user/vanhoeve/mdd/

acire@utsc.utoronto.ca

## Parallel Architecture

- We consider a centralized architecture
- Master maintains a pool of states to process
- Workers receive states, generate relaxed diagrams, and send new states to master
- Suitable to small architectures (up to
 256 cores)


## Master \& Workers Pools

- Master keeps a priority queue of states
- States with better optimization bounds have a higher priority of being explored
- Workers also keep a local priority queue
- Relaxed (and restricted) decision diagrams are computed very quickly
- Reduce communication to master
- Key issue: large memory consumption
- Pools may grow quickly for very large problems
- If memory is almost exceeded, priority queue becomes a regular queue (depth-first search)


## Load Balancing

- Crucial question in many parallelization scheme
- In our case: How to distribute states among workers?
- Too many nodes at once: many workers will be idle
- Too few nodes: communication becomes bottleneck


## Load Balancing


where $\mathbf{c}$ and $\mathbf{c}^{\prime}$ are some constants (in our experiments, $\mathrm{c}=\mathrm{c}^{\prime}=2$ )

## Load Balancing

$75 \%$ of nodes with best optimization bounds

- Speed up the processing of promising nodes


## Computational Results

- Relaxed decision diagrams implemented in C++
- Parallel architecture implemented in X10
- IBM X10 Team: Vijay Saraswat et al
- x10-lang.org
- Tested in a computer cluster with 256 cores
- 16 computers, each with 32 cores, 64 GB RAM



CPLEX


## Other results

- Also observe same behaviour for other problem classes
- Proved optimality for some maxcut instances for the first time
- Testing on some variations of constrained TSP
- Other architectures
- Work-stealing models

Thank you!

## Relaxed Decision Diagrams

- Computational study on the max. independent set problem
- Able to provide tighter bounds than integer programming models
- Application on Single-Machine Scheduling Problems
- Closed open TSPLib instances, orders of magnitude improvement over constraint programming models, plus theoretical properties
- Application on Timetabling Problems
- Orders of magnitude speed up in solving times compared to state-of-the-art approaches, plus theoretical properties


## Modeling Framework

Ex.: Maximum independent set problem


- Our model: Dynamic Programming
- Exploit recursiveness
- Solved by stages
- Passing from one stage to another corresponds to transitioning from a state to another
- Decision diagram: State-Transition Graph
- Nodes corresponds to states
- Arcs are state transitions
- Arc weights are transition costs


## Modeling Framework

Ex.: Maximum independent set problem


- DP model for the maximum independent set:

$$
\begin{aligned}
& V_{i}(S)= \begin{cases}\max \left\{V_{i-1}(S \backslash\{i\}), V_{i-1}(S \backslash N(i))+1\right\}, & i \in S \\
V_{i-1}(S \backslash N(i)), & o . w .\end{cases} \\
& V_{i}(\phi)=0, \quad i=1, \ldots, 5
\end{aligned}
$$

- Highlights:
- Stage i: select vertex i
- State: set of eligible vertices


## Filtering


$\max 4 x_{1}+4 x_{2}+x_{3}$
subject to

$$
\begin{aligned}
& x_{1}+x_{2}+x_{3} \leq 4 \\
& x_{1}, x_{2}, x_{3} \in\{1,2\}
\end{aligned}
$$

- Max Width = 2
- State: left-hand side of constraint


## Filtering


$\max 4 x_{1}+4 x_{2}+x_{3}$
subject to

$$
\begin{aligned}
& x_{1}+x_{2}+x_{3} \leq 4 \\
& x_{1}, x_{2}, x_{3} \in\{1,2\}
\end{aligned}
$$

- Max Width = 2
- State: left-hand side of constraint
- Longest path: $x_{1}=x_{2}=x_{3}=1$


## Filtering



$$
\begin{aligned}
& \max 4 x_{1}+4 x_{2}+x_{3} \\
& \text { subject to } \\
& \quad x_{1}+x_{2}+x_{3} \leq 4 \\
& \quad x_{1}, x_{2}, x_{3} \in\{1,2\}
\end{aligned}
$$

- Note that top-down is a forward recursion:

$$
V_{i}(\ldots)=V_{i-1}(\ldots)+\ldots
$$

## Filtering



$$
\begin{aligned}
& \max 4 x_{1}+4 x_{2}+x_{3} \\
& \text { subject to } \\
& \quad x_{1}+x_{2}+x_{3} \leq 4 \\
& \quad x_{1}, x_{2}, x_{3} \in\{1,2\}
\end{aligned}
$$

- But what happens when we do a backward recursion?


## Filtering



$$
\max 4 x_{1}+4 x_{2}+x_{3}
$$

subject to

$$
\begin{aligned}
& x_{1}+x_{2}+x_{3} \leq 4 \\
& x_{1}, x_{2}, x_{3} \in\{1,2\}
\end{aligned}
$$

- But what happens when we do a backward recursion?


## Filtering



$$
\max 4 x_{1}+4 x_{2}+x_{3}
$$

subject to

$$
\begin{aligned}
& x_{1}+x_{2}+x_{3} \leq 4 \\
& x_{1}, x_{2}, x_{3} \in\{1,2\}
\end{aligned}
$$

- Underlying concept: Use "redundant" DP formulations to remove arcs, e.g.:

$$
V_{i}^{\prime}(\ldots)=V_{i-1}^{\prime}(\ldots)+V_{i+1}^{\prime}(\ldots)+\ldots
$$

## Some theoretical insights



- Let $\mathbf{X}$ the set of solutions represented by an MDD
- Optimizing a linear function $f$ over the MDD is equivalent to solving the LP problem:

| Minimize $c x$ |
| :--- | :--- |
| subject to |
| $x$ is a flow from $r$ to $t$ |$=\quad$| Minimize $c x$ |
| :--- |
| subject to |
| $x \in \operatorname{conv}(X)$ |

## Some theoretical insights



- Let $\mathbf{A x} \geq \mathbf{b}$ be a set of constraints that we dualize over the MDD.
- If $z^{*}$ is the optimal shortest path after dualization, then

$$
z^{*}=\begin{aligned}
& \text { Minimize } c x \\
& \text { subject to } \\
& A x \geq b \\
& x \in \operatorname{conv}(X)
\end{aligned}
$$

