An MDD approach to multidimensional bin packing

Brian Kell and Willem-Jan van Hoeve

Carnegie Mellon University

CPAIOR
May 20, 2013

Outline

Introduction

Multidimensional bin packing

- Given: A set of m bin capacities and a set of n item sizes.
- Each bin capacity and each item size is a d-tuple of nonnegative integers.
- Objective: Assign each item to a bin so that no bin capacity is exceeded in any dimension.
- This is a satisfaction problem.

Multivalued decision diagram (MDD)

- Edge-labeled acyclic directed multigraph.
- Nodes arranged in layers.
- Top layer contains the *root*; bottom layer contains the *sink*.
- Each edge goes from one layer to the next.

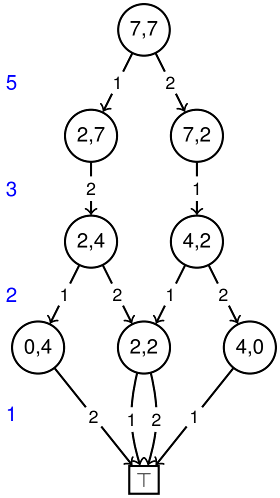
Direct MDD representation for bin packing

Structure of the MDD

- The state of a node in layer *i* is the list of remaining bin capacities after the first $i - 1$ items have been placed.
- Edges represent the options for packing the next item (i.e., the various bins).
- Each path from the root to the sink represents a feasible solution.

Example (to the right)

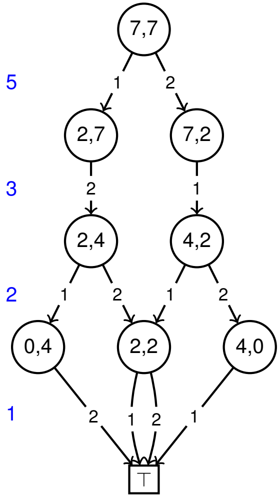
- One dimension.
- Two bins, each of capacity 7.
- Four items, with sizes 5, 3, 2, and 1.



Exact MDD construction

Top-down compilation

- Build the MDD layer by layer.
- For each node in the current layer, identify outgoing edges.
- For each outgoing edge, determine the resulting state.
- If the next layer already contains a node with that state, point the edge there. Otherwise construct a new node.



Exact MDD construction

Exploratory construction

- Maintain a collection T of nodes to be processed.
- To process a node in layer i:
    - Identify outgoing edges.
    - For each outgoing edge, determine the resulting state.
    - If layer $i + 1$ already contains a node with that state, point the edge there. Otherwise construct a new node.

Observations

- If T is a queue: top-down compilation.
- If T is a stack: depth-first search.
- Use heuristic to identify most promising nodes: heuristic-driven depth-first search.

Approximate MDDs

- In general, exact MDDs can have exponential size.
- *Approximate MDDs* approximate the structure of exact MDDs.
- Represent the solution set to a relaxation or restriction of the original instance.
- Idea (Andersen, Hadzic, Hooker, Tiedemann, 2007): Limit the width of each layer of the MDD.

Approximate MDDs by merging

- When the width of a layer exceeds a preset limit, reduce the size of the layer by *merging* nodes.
- The state of a merged node should be a relaxation or restriction of the states of the original nodes.
- For multidimensional bin packing:
  - Relaxation merge = componentwise maximum.
  - Restriction merge = componentwise minimum.
- Bergman, van Hoeve, Hooker (2011): Merge nodes a pair at a time.
- New approach: Use a clustering algorithm to partition the nodes, and merge each cluster.
  - E.g., median cut algorithm of Heckbert (1982).

Restriction MDDs by deletion

- When the width of a layer exceeds a preset limit, reduce the size of the layer by *deleting* some nodes.
- Use a heuristic to determine the most promising nodes to keep.
- Trades cost of clustering algorithm for cost of heuristic.

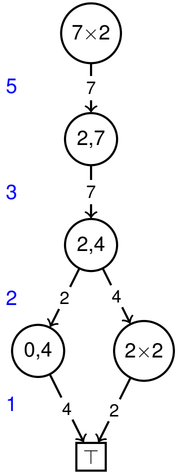
Ullage MDD representation for bin packing

Structure of the MDD

- *Ullage*: the amount by which a container falls short of being full (i.e., remaining bin capacity).
- To handle symmetry in bins, node states are now multisets of ullages.
- Items are assigned to ullages rather than specific bins.

Example (to the right)

- One dimension.
- Two bins, each of capacity 7.
- Four items, with sizes 5, 3, 2, and 1.



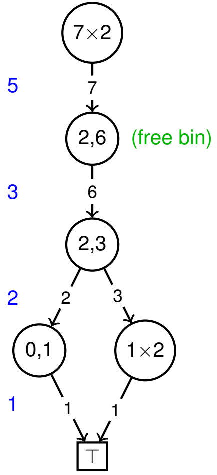
Node states in the ullage MDD representation

Detecting equivalent states

- Round down each ullage to the nearest sum of a subset of remaining item sizes (these sums can be precomputed).
- “Dead” bin: so small that none of the remaining items will fit (can be ignored).

Detecting feasibility and infeasibility

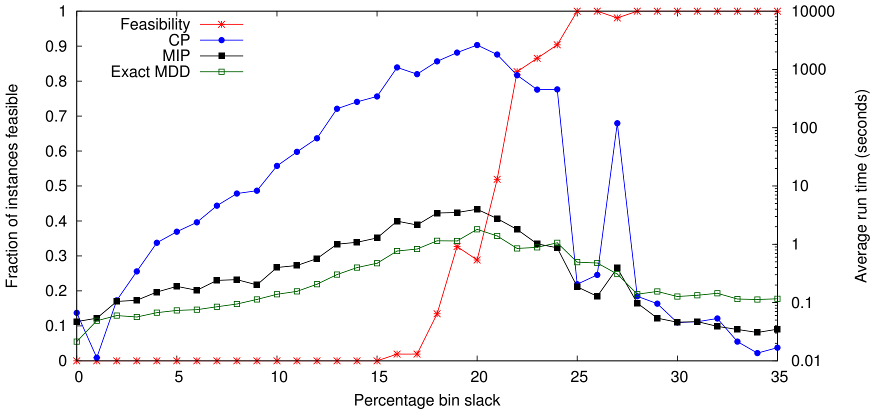
- Total ullage may be too small for remaining items (implies infeasibility).
- “Free” bin: large enough in every dimension that all remaining items will fit (implies feasibility).



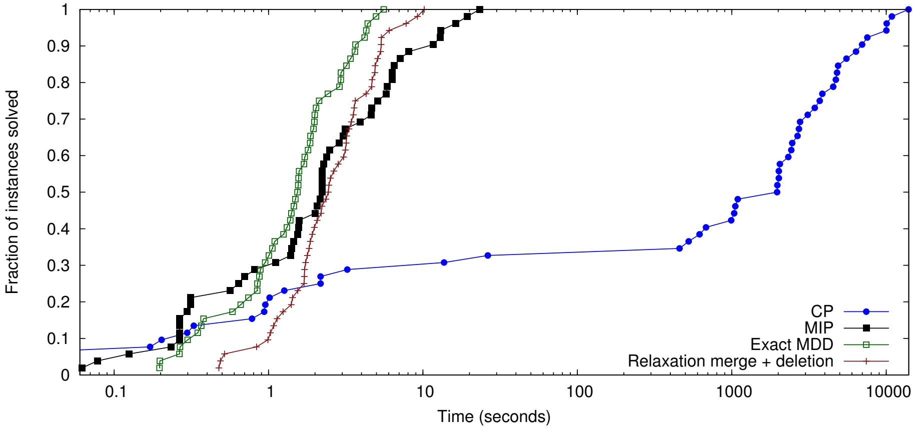
Experimental setup

- Test instances:
  - 6 dimensions
  - 6 identical bins
  - 18 items of randomly generated sizes in $\{0, \ldots, 1000\}$
  - 0% to 35% bin slack: 52 instances at each value
- Our algorithms implemented in Java, using ullage MDD representation
- Maximum MDD width: 5000 nodes
- For comparison: AIMMS 3.13
  - CP solver: CPOptimizer 12.4, using independent `cp::BinPacking` constraints
  - MIP solver: CPLEX 12.4

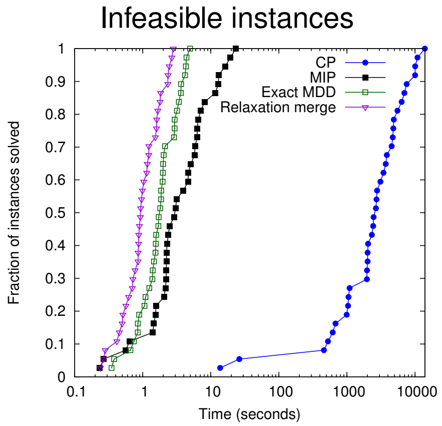
Feasibility and hardness profiles

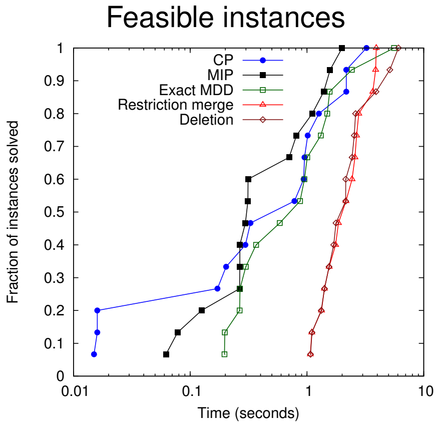


Performance profile: 20% bin slack



Infeasible vs. feasible instances: 20% bin slack





Conclusions

- Several variations of a generic algorithm for construction of exact and approximate MDDs.
  - Heuristic-driven depth-first method for exact MDDs.
  - Application of clustering algorithm for approximate MDDs.
- Techniques to apply MDDs effectively to multidimensional bin packing.
  - Ullage MDD representation to handle symmetry.
  - Tricks to detect equivalent states and to detect feasibility and infeasibility.
- Experimental results show these techniques can outperform current CP and MIP solvers.