A Branch-and-Price Algorithm for the Bin Packing Problem with Conflicts

Ruslan Sadykov[1] François Vanderbeck[1,2]

[1]INRIA — Bordeaux Sud-Ouest, France

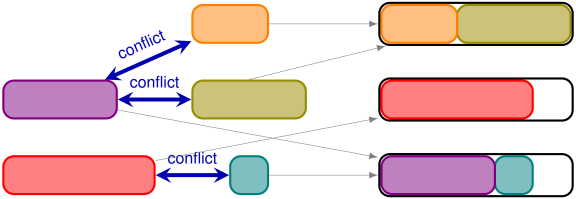
[2]Université Bordeaux I, France

ISMP, August 25, 2009

Contents

Problem definition

- Given
  - an infinite number of bins of size W,
  - a set $N = \{1, \ldots, n\}$ of items i of sizes w_i,
  - a graph $G = (N, E)$ of conflicts between items.
- Pack the items into a minimum number of bins.
- Two items in conflict cannot be in the same bin.



IP formulation

Variables:

- $y_k = 1$ if bin k is used, otherwise $y_k = 0$
- $x_{ik} = 1$ if item i is put to bin k, otherwise $x_{ik} = 0$

$$
\begin{aligned}
\min \sum_{k \in K} y_k & & \\
\sum_{k \in K} x_{ik} &= 1, & i \in N, \\
\sum_{i=1}^{n} w_i x_{ik} &\leq W y_h, & k \in K, \\
x_{ik} + x_{jk} &\leq y_k, & (i,j) \in E, k \in K, \\
y_k &\in \{0,1\}, & k \in K, \\
x_{ik} &\in \{0,1\}, & i \in N, k \in K.
\end{aligned}
$$

Contents

Set covering formulation

$\mathcal{B}$ = set of valid single bin packings
$\lambda_b = 1$ if subset $b \in \mathcal{B}$ of items occupies a bin

Formulation

$$\min \sum_{b \in \mathcal{B}} \lambda_b$$
$$\sum_{b \in \mathcal{B}: i \in b} \lambda_b \geq 1, \qquad i \in N,$$
$$\lambda_b \in \{0,1\}, \; b \in \mathcal{B}.$$

Pricing problem

$$\max \sum_{i \in N} \pi_i x_i$$
$$\sum_{i \in N} w_i x_i \leq W,$$
$$x_i + x_j \leq 1, (i,j) \in E,$$
$$x_i \in \{0,1\}, i \in N.$$

(Knapsack with conflicts)

Set covering formulation

$\mathcal{B}$ = set of valid single bin packings
$\lambda_b = 1$ if subset $b \in \mathcal{B}$ of items occupies a bin

Formulation

$$\min \sum_{b \in \mathcal{B}} \lambda_b$$
$$\sum_{b \in \mathcal{B}: i \in b} \lambda_b \geq 1, \quad i \in N,$$
$$\lambda_b \in \{0,1\}, \; b \in \mathcal{B}.$$

Pricing problem

$$\max \sum_{i \in N} \pi_i x_i$$
$$\sum_{i \in N} w_i x_i \leq W,$$
$$x_i + x_j \leq 1, (i,j) \in E,$$
$$x_i \in \{0,1\}, i \in N.$$

(Knapsack with conflicts)

CG approach in the literature

Elhedhli, Li, Gzara (2008):

- **Subproblem** with clique constraints: CPLEX 10.
- **Branching**: specialised *Ryan&Foster* type ($P + 1$ descendants: combination of P pairs of items + adding a conflict for each pair separately).
- **Rounding heuristic** based on subsets b such that λ_b is close to 1.

Muritiba, Iori, Malaguti, Toth (2008):

- **Pre-processing** using combinatorial lower bounds and specialized meta-heuristics.
- **Subproblem**: greedy algorithm + CPLEX 10.
- **Branching**: *Largest fractional part* on λ variables.

CG approach in the literature

Elhedhli, Li, Gzara (2008):

- **Subproblem** with clique constraints: CPLEX 10.
- **Branching**: specialised *Ryan&Foster* type ($P+1$ descendants: combination of P pairs of items + adding a conflict for each pair separately).
- **Rounding heuristic** based on subsets b such that λ_b is close to 1.

Muritiba, Iori, Malaguti, Toth (2008):

- **Pre-processing** using combinatorial lower bounds and specialized meta-heuristics.
- **Subproblem**: greedy algorithm + CPLEX 10.
- **Branching**: *Largest fractional part* on λ variables.

Test instances

Due to Gendreau, Laporte, Semet (2004):

- Sizes (integer):
  - **Uniforms**(u): $w_i \in U[20, 100]$, $W = 150$.
  - **Triples**(t): $w_i \in U[250, 500]$ (in triples), $W = 1000$.
- "Conflictness" of items: $p_i \in U[0, 1)$.
- Conflict graph density: $\delta \in \{0, 0.1, \ldots, 0.9\}$.
- $(i, j) \in E$ if and only if $(p_i + p_j)/2 \geq 1 - \delta$.

Results

Percentage of instances **not solved** within 1 hour:

type & # of jobs	u250	u500	u1000	t120	t249	t501
ELG	0.5%	6.5%	–	1.5%	5%	–
MIMT	0%	5%	2%	5%	4%	4%

Test instances

Due to Gendreau, Laporte, Semet (2004):

- Sizes (integer):
  - **Uniforms**(u): $w_i \in U[20, 100]$, $W = 150$.
  - **Triples**(t): $w_i \in U[250, 500]$ (in triples), $W = 1000$.
- "Conflictness" of items: $p_i \in U[0, 1)$.
- Conflict graph density: $\delta \in \{0, 0.1, \ldots, 0.9\}$.
- $(i, j) \in E$ if and only if $(p_i + p_j)/2 \geq 1 - \delta$.

Results

Percentage of instances **not solved** within 1 hour:

type & # of jobs	u250	u500	u1000	t120	t249	t501
ELG	0.5%	6.5%	–	1.5%	5%	–
MIMT	0%	5%	2%	5%	4%	4%

Our algorithm

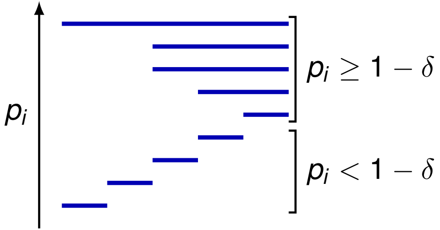
- We use a generic Branch-and-Price solver BaPCod:
  - generic branching,
  - “diving” heuristic.
- We exploit the structure of the conflict graphs of the test instances (interval graphs):

$(i, j) \in E$ iff $\frac{p_i + p_j}{2} \geq 1 - \delta$

Our algorithm

- We use a generic Branch-and-Price solver BaPCod:
  - generic branching,
  - "diving" heuristic.
- We exploit the structure of the conflict graphs of the test instances (interval graphs):

$$(i,j) \in E \ \text{iff} \ \frac{p_i+p_j}{2} \geq 1-\delta$$

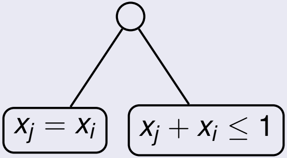


Generic branching scheme

λ is fractional $\Leftrightarrow$ exists a pair i, j such that $\sum_{i,j \in b} \lambda_b \neq \{0, 1\}$

Branching: either item i and j are in the same bin or not, we add constraints to the pricing subproblem.

Ryan&Foster scheme



The interval conflict graph structure can be broken.

Generic branching scheme

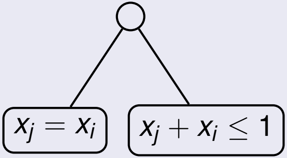
- Does not brake graph structure.
- Stronger LP bound after branching.
- More subproblems to solve.

Generic branching scheme

λ is fractional $\Leftrightarrow$ exists a pair i, j such that $\sum_{i,j \in b} \lambda_b \neq \{0, 1\}$

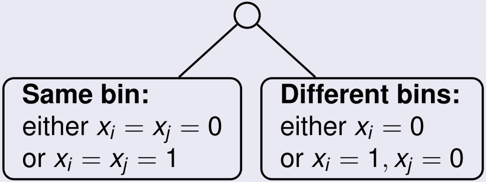
Branching: either item i and j are in the same bin or not, we add constraints to the pricing subproblem.

Ryan&Foster scheme



The interval conflict graph structure can be broken.

Generic branching scheme

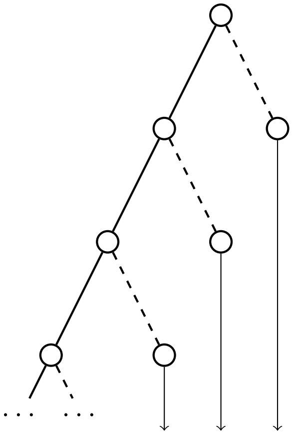


- Does not brake graph structure.
- Stronger LP bound after branching.
- More subproblems to solve.

“Diving” rounding heuristic

Combines:

- Depth-first search on λ variables
- Diversification (Limited Discrepancy Search)
- Pre-processing



Contents

Problem definition

- Given
  - a set $N = \{1, \ldots, n\}$ of items i of sizes w_i and profits p_i,
  - a graph $G = (N, E)$ of conflicts between items.
- Find a subset of items of a maximum total profit which “fit” to the bin of size W.
- This subset cannot contain any pair of items in conflict.

Formulation reminder

$$\max \sum_{i \in N} \pi_i x_i$$

$$\begin{aligned} \sum_{i \in N} w_i x_i &\leq W, \\ x_i + x_j &\leq 1, (i,j) \in E, \\ x_i &\in \{0,1\}, i \in N. \end{aligned}$$

Existent approaches

Arbitrary conflict graphs

NP-hard, $\approx$ 100 seconds for solving exactly a 1000-items instance (Hifi, Michrafy, 2007) — slow.

Structured conflict graphs

- **Trees** and **chordal** graphs: dynamic programming algorithm with complexity $O(nW^2)$ (Pferschy, Shauer, 2008) — slow.
- **Interval** graphs: these instances we can solve fast.

Existent approaches

Arbitrary conflict graphs

NP-hard, $\approx$ 100 seconds for solving exactly a 1000-items instance (Hifi, Michrafy, 2007) — slow.

Structured conflict graphs

- **Trees** and **chordal** graphs: dynamic programming algorithm with complexity $O(nW^2)$ (Pferschy, Shauer, 2008) — slow.
- **Interval** graphs: these instances we can solve fast.

Existent approaches

Arbitrary conflict graphs

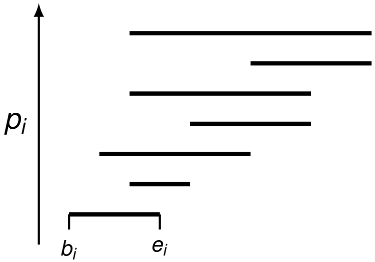
NP-hard, $\approx$ 100 seconds for solving exactly a 1000-items instance (Hifi, Michrafy, 2007) — slow.

Structured conflict graphs

- **Trees** and **chordal** graphs: dynamic programming algorithm with complexity $O(nW^2)$ (Pferschy, Shauer, 2008) — slow.
- **Interval** graphs: these instances we can solve fast.

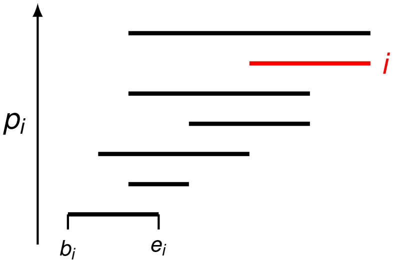
Interval conflict graphs: dynamic programming

$P(i, w)$ — solution value of the subproblem with the first i items and bin size w



Interval conflict graphs: dynamic programming

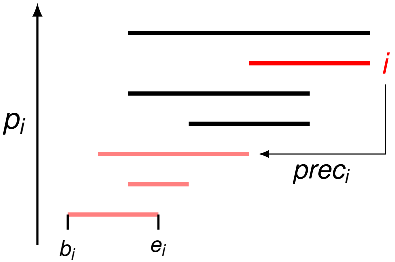
$P(i, w)$ — solution value of the subproblem with the first i items and bin size w



$$P(i, w) = \max \{$$

Interval conflict graphs: dynamic programming

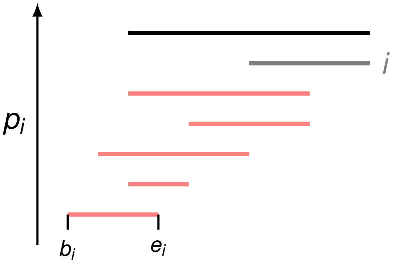
$P(i, w)$ — solution value of the subproblem with the first i items and bin size w



$$P(i, w) = \max \{ \; P(prec_i, w - w_i) + p_i,$$

Interval conflict graphs: dynamic programming

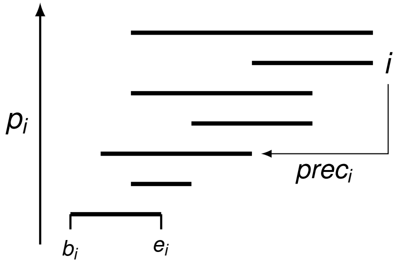
$P(i, w)$ — solution value of the subproblem with the first i items and bin size w



$$P(i, w) = \max \{ P(prec_i, w - w_i) + p_i, \; P(i-1, w) \}$$

Interval conflict graphs: dynamic programming

$P(i, w)$ — solution value of the subproblem with the first i items and bin size w



$$P(i, w) = \max\left\{ P(prec_i, w - w_i) + p_i, \; P(i-1, w) \right\}$$

Complexity: $O(nW)$, same as for the usual knapsack problem!

Contents

Numerical experiments

10 instances not solved by Muritiba et al. (2008) were tested.

- 8 instances were solved to optimality.
- For 9 instances, we found improved solutions.

Name	Type, # items	# nodes	Sol.	Improv.	Time
6_3_6	t120	533	41	0	3m11s
7_3_4	t249	1	83	-1	1m13s
8_3_4	t501	136	167	-2	9m05s
3_4_4	u500	1	204	-3	14m04s
3_4_5	u500	1	206	-1	15m58s
3_4_7	u500	1	208	-4	12m11s
3_4_8	u500	1	205	-1	10m08s
3_4_9	u500	471	197	-3	1h00m01s
4_4_8	u1000	1	404	-7	1h42m16s
4_4_10	u1000	1	398	-8	2h46m46s

Numerical experiments

10 instances not solved by Muritiba et al. (2008) were tested.

- 8 instances were solved to optimality.
- For 9 instances, we found improved solutions.

Name	Type, # items	# nodes	Sol.	Improv.	Time
6_3_6	t120	533	41	0	3m11s
7_3_4	t249	1	83	-1	1m13s
8_3_4	t501	136	167	-2	9m05s
3_4_4	u500	1	204	-3	14m04s
3_4_5	u500	1	206	-1	15m58s
3_4_7	u500	1	208	-4	12m11s
3_4_8	u500	1	205	-1	10m08s
3_4_9	u500	471	197	-3	1h00m01s
4_4_8	u1000	1	404	-7	1h42m16s
4_4_10	u1000	1	398	-8	2h46m46s

Effect of the “diving” heuristic!

Conclusions and perspectives

Conclusions

- Knapsack problem with interval conflict graph can be solved efficiently (and fast!) by dynamic programming.
- Generic branch-and-price solver BaPCod is competitive with specialized oracle.
- Very good performance of the generic “diving” heuristic.

Future research

- Improvement of BaPCod (other generic primal heuristics, pre-processing,...)
- We need faster (in practice) algorithms for the knapsack problem with specialized and arbitrary conflict graphs

Conclusions and perspectives

Conclusions

- Knapsack problem with interval conflict graph can be solved efficiently (and fast!) by dynamic programming.
- Generic branch-and-price solver BaPCod is competitive with specialized oracle.
- Very good performance of the generic “diving” heuristic.

Future research

- Improvement of BaPCod (other generic primal heuristics, pre-processing,...)
- We need faster (in practice) algorithms for the knapsack problem with specialized and arbitrary conflict graphs