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

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

[1]INRIA — Bordeaux Sud-Ouest, ReAlOpt team

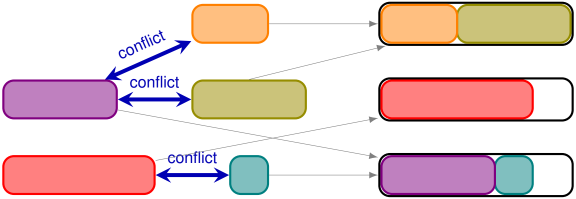
[2]Université Bordeaux I

ROADEF, February 25, 2010

Contents

Problem definition

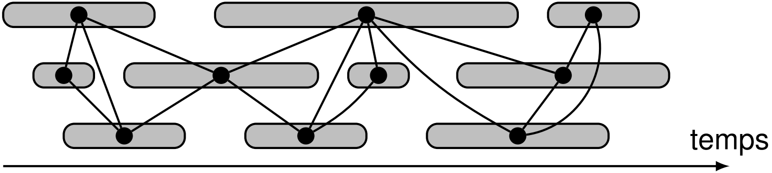
- Given
  - an infinite number of bins of size W,
  - a set $N = \{1, \ldots, n\}$ of items i of sizes w_i,
  - a graph $G = (N, E)$ of conflicts between items.
- Pack the items into a minimum number of bins.
- Two items in conflict cannot be in the same bin.



Motivations

Mutual exclusion scheduling (generalisation)

(Baker & Coffman, 96), (Gardi, 05)



- Total length of tasks assigned to a person $\leq W$
- Overlapping tasks cannot be assigned to the same person : conflict graph is interval

Other applications

- Examination scheduling (Laporte & Desroches, 84)
- Parallel computation (Jansen, 99), (Beaumont et al., 08)
- Product delivery (Christofides et al.,79)

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 $x_{ik} = 0$

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

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)

Knapsack with conflicts : existent approaches

Arbitrary conflict graphs

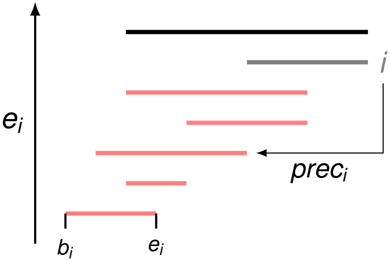
NP-hard, $\approx$ 100 seconds for solving exactly a 1000-items instance (Hifi, Michrafy, 07) — slow.

Structured conflict graphs

- **Trees** and **chordal** graphs : dynamic programming algorithm with complexity $O(nW^2)$ (Pferschy, Shauer, 09) — 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



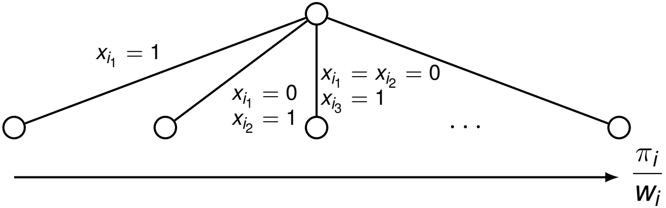
$$P(i, w) = \max \{ P(prec_i, w - w_i) + p_i, \; P(i-1, w) \}$$

Complexity : $O(nW)$, same as for the knapsack without conflicts !

Interval conflict graphs : enumeration algorithm

A simple Branch-and-Bound algorithm, extension of (Carraghan & Pardalos, 90) :

- Dual bound : relaxation of conflicts between non-fixed items and integrality.
- Depth-first search.
- Branching :



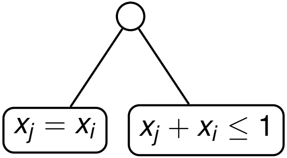
- Time we spent at each node explored is $O(n)$.

Generic branching scheme

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

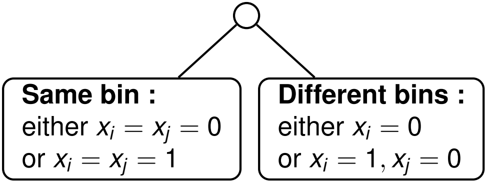
Branching : either item *i* and *j* are in the same bin or not, we add constraints to the pricing subproblem.

Ryan&Foster scheme



The interval conflict graph structure can be broken.

Generic branching scheme



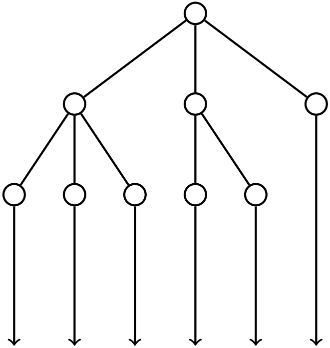
- Does not brake graph structure.
- Better LP bound after branching.
- More subproblems to solve.

“Diving” rounding generic heuristic

Will be presented at ISCO’10 : (Joncour et al., 10)

Combines :

- Depth-first search : we choose a column ($\lambda_B \leftarrow 1$) and change the right-hand side of the formulation
- The number of generated columns at each node (except the root is limited)
- Diversification (Limited Discrepancy Search)
- Pre-processing



MaxDepth = 2
MaxDiscrepancy = 2

Implementation

We used BaPCod (a generic Branch-and-Price Code) being developed by ReAlOpt

By default, we have :

- Stabilized column generation procedure
- Generic branching
- Generic "diving" heuristic

The user supplies

- Compact problem formulation
- An oracle for solving the pricing problem

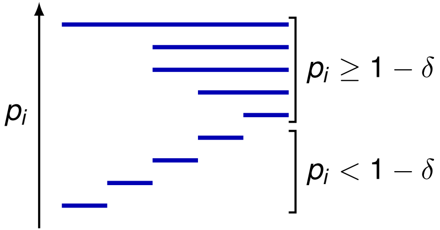
Contents

Instances de test

Due to (Gendreau, Laporte, and Semet, 04) :

- 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 : $\delta \in \{0, 0.1, \dots, 0.9\}$.
- Conflict graphs structure :

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



Comparison of exact algorithms

MIMT : (Muritiba, Iori, Malaguti, and Toth, 09)
ELGN : (Elhedhli, Li, Gzara, and Naoum-Sawaya, 09)

The times adjusted using `www.spec.org`

	MIMT		ELGN		Notre	
class	not opt.	time	not opt.	time	not opt.	time
t60	0%	61	0%	3	0%	2
t120	5%	2976	3%	119	0%	24
t249	4%	2581	11%	398	0%	51
t501	4%	5060	–	–	0%	280
u120	0%	46	0%	47	0%	6
u250	0%	171	1%	183	0%	23
u500	5%	3512	14%	1254	0%	120
u1000	2%	3057	–	–	0%	884

Comparison of heuristics

- **PH** : population heuristic based on tabu search (Muritiba et al., 09)
- **DH** : "diving heurisitc" (without LDS)
- **DH LDS** : "diving heuristic" (with LDS)

	PH		DH		DH LDS	
class	gap	time	gap	time	gap	time
t60	0.45%	60	0.11%	2	0.00%	2
t120	0.62%	64	0.66%	4	0.00%	9
t249	0.39%	83	0.35%	8	0.00%	51
t501	0.21%	94	0.16%	17	0.00%	280
u120	0.10%	36	0.18%	4	0.00%	5
u250	0.21%	83	0.07%	9	0.00%	23
u500	0.20%	112	0.03%	17	0.00%	120
u1000	0.22%	172	0.01%	34	0.002%	822

Open instances

10 instances which was not solved by (Muritiba et al., 09).

Name	class	# nodes	Sol.	Improv.	Time
6_3_6	t120	609	41	0	3m45s
7_3_4	t249	1	83	-1	29s
8_3_4	t501	1	167	-2	53s
3_4_4	u500	1	204	-3	30s
3_4_5	u500	1	206	-1	29s
3_4_7	u500	1	208	-4	21s
3_4_8	u500	1	205	-1	25s
3_4_9	u500	1	196	-4	36s
4_4_8	u1000	1	404	-7	4m5s
4_4_10	u1000	1	397	-9	7m9s

Effect of the “diving” heuristic

Instances with arbitrary conflict graphs

- The same instances except that the conflicts where generated arbitrarily.
- Time limit : 1 hour.

class	not opt.	time (opt.)	gap (not opt.)	nodes enum.
t60	0%	0	–	384
t120	0%	3	–	1316
t249	5.6%	135	1.1%	3418
t501	27.8%	476	0.6%	16479
u120	0%	1	–	255
u250	1.1%	11	1.0%	405
u500	8.9%	47	0.5%	481
u1000	11.1%	331	0.3%	398

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

- We need fast algorithms for the knapsack problem with arbitrary conflict graphs (can we do noticeably better that Cplex ?)
- Improvement of BaPCod (other generic primal heuristics, pre-processing,...)