Solving Bin Packing Problems with Fragmentation

M. Casazza, [1] **A. Ceselli** [2]

Università degli Studi di Milano
Dipartimento di Informatica

[1] Currently at LIP6 – Univ. Pierre et Marie Curie – Paris 6

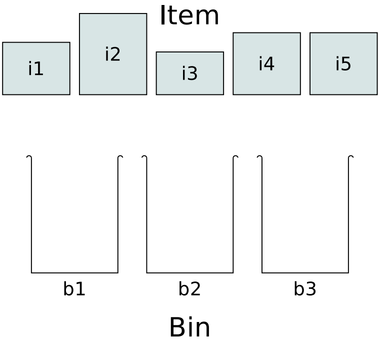
[2] Partially funded by Regione Lombardia – Fondazione Cariplo, grant agreement no. 2015-0717, project `REDNEAT`.

Bin Packing Problems

Classical Bin Packing Problem (BPP):

- NP-HARD
- well studied in the OR community
- models many problems in logistics ...

BPP with item fragmentation (BPP-IF): items can be split at a price.

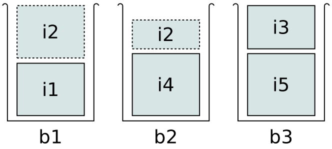


Bin Packing Problems

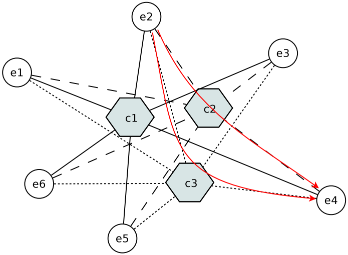
Classical Bin Packing Problem (BPP):

- NP-HARD
- well studied in the OR community
- models many problems in logistics ...

BPP with item fragmentation (BPP-IF): items can be split at a price.



BPP-IF in telecommunications

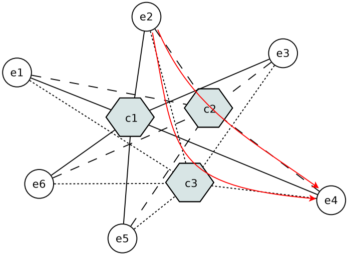


Tactical issues in (consolidated) routing:

- channels: (path, frequency, timeslot) tuples $\Rightarrow$ bin
- data transfer requests of different carriers $\Rightarrow$ items
- splitting requests consumes energy $\Rightarrow$ split items as few as possible

Initial attempts with CPLEX: timeout on instances with 10 or 20 items.
Spoiler:

BPP-IF in telecommunications



Tactical issues in (consolidated) routing:

- channels: (path, frequency, timeslot) tuples $\Rightarrow$ bin
- data transfer requests of different carriers $\Rightarrow$ items
- splitting requests consumes energy $\Rightarrow$ split items as few as possible

Initial attempts with CPLEX: timeout on instances with 10 or 20 items.
Spoiler: branch-and-price could tackle instances with up to 1000 items.

BPP-IF in transportation

	Split Forbidden	Split Allowed
Routing costs	VRP	Split Delivery VRP
No routing costs	BPP	BPP with Item Fragm.

Literature Review

Main BPP-IF variants

- Bin Minimization; Fragmentations minimization;
- Size Preserving; Size Increasing (weight overhead for each split)

Approximation algorithms and applications:

- N. Menakerman, R. Rom., *Bin packing with item fragmentation*. LNCS, proc. of the 7th WADS (2001)
- H. Shachnai, T. Tamir, O. Yehezkely, *Approximation schemes for packing with item fragmentation*. Theory of Comp. Sys. 43 (2008)
- L. Epstein, A. Levin, R. van Stee, *Approximation schemes for packing splittable items with cardinality constraints*. Algorithmica 62 (2012).
- B. Lecun, T. Mautor, F. Quessette, M.A. Weisser *Bin packing with fragmentable items: presentation and approximations*. Theoretical Computer Science 602 (2015)

Literature Review

Main BPP-IF variants

- Bin Minimization; Fragmentations minimization;
- Size Preserving; Size Increasing (weight overhead for each split)

Approximation algorithms and applications:

- N. Menakerman, R. Rom., *Bin packing with item fragmentation*. LNCS, proc. of the 7th WADS (2001)
- H. Shachnai, T. Tamir, O. Yehezkely, *Approximation schemes for packing with item fragmentation*. Theory of Comp. Sys. 43 (2008)
- L. Epstein, A. Levin, R. van Stee, *Approximation schemes for packing splittable items with cardinality constraints*. Algorithmica 62 (2012).
- B. Lecun, T. Mautor, F. Quessette, M.A. Weisser *Bin packing with fragmentable items: presentation and approximations*. Theoretical Computer Science 602 (2015)

Results overview:

Our approach:

- try to understand which features make the problem so difficult
- design *math. prog.* algorithms for solving BPPIFs

Main results

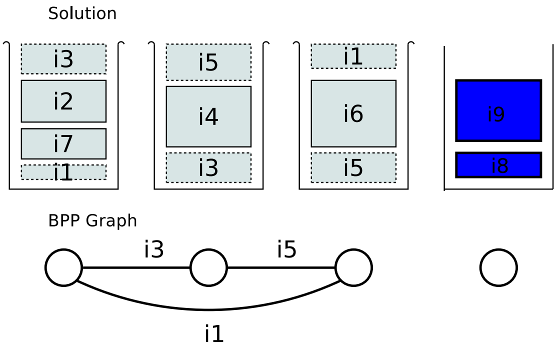
- A common framework for modeling and solving BPPIFs (bin or fragmentation minimization, size preserving or increasing)
- A characterization of particular subsets of optimal solutions
- Exact algorithms whose computing time scales very well in practice

Structure of a solution

BPP-IF Graph

Given a BPP-IF solution, build a *BPP-IF graph*:

- one vertex for each bin
- one edge for each pair of bins with fragments of the same item

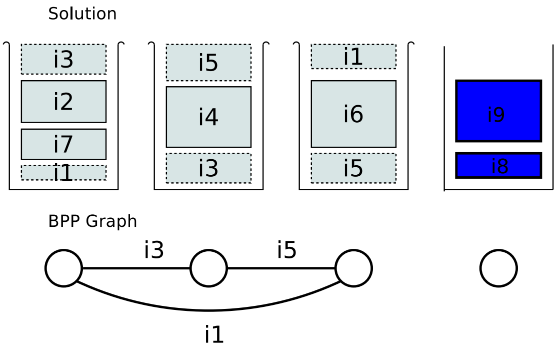


Structure of a solution

Primitive solutions

Def. *primitive solution* (MR '01):

- each item is split in at most two fragments
- each bin contains at most two fragmented items
- $\rightarrow$ the BPP-IF graph is a set of paths
- **def:** items beloging to bins of the same path form a *chain*.

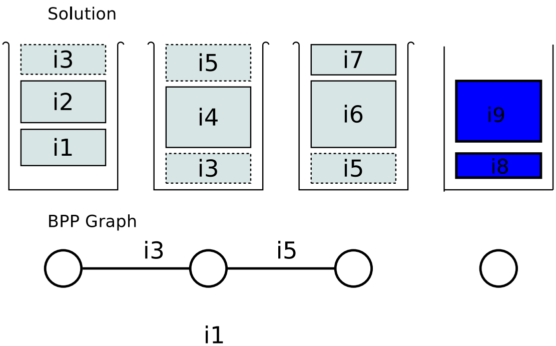


Structure of a solution

Primitive solutions

Def. *primitive solution* (MR '01):

- each item is split in at most two fragments
- each bin contains at most two fragmented items
- $\rightarrow$ the BPP-IF graph is a set of paths
- **def:** items beloging to bins of the same path form a *chain*.



Structure of a BPP-IF solution

Theorem (MR '02)

There always exists a b.m. BPP-IF optimal solution which is primitive.

Theorem (CC '13)

There always exists a f.m. BPP-IF optimal solution which is primitive.

Theorem (MR '01 - CC '13)

given the set of items belonging to each chain in a primitive solution, a full BPP-IF solution can be found by running Next Fit procedures.

Theorem (CC '13)

There always exists an (optimal) primitive dominant solution, in which the split items of each chain of k bins are the $k-1$ items of maximum weight belonging to that chain;

Structure of a BPP-IF solution

Theorem (MR '02)

There always exists a b.m. BPP-IF optimal solution which is primitive.

Theorem (CC '13)

There always exists a f.m. BPP-IF optimal solution which is primitive.

Theorem (MR '01 - CC '13)

given the set of items belonging to each chain in a primitive solution, a full BPP-IF solution can be found by running Next Fit procedures.

Theorem (CC '13)

There always exists an (optimal) primitive dominant solution, in which the split items of each chain of k bins are the $k-1$ items of maximum weight belonging to that chain;

Overall algorithm (N)

Branch-and-bound, exploiting:

- **bounding: compute valid lower bounds**
- approximation: compute upper bounds
- branching: (a) fix split / non split items; (b) fix *pairs* of split items for the same bin; (c) assign items to bins whose pairs of split items are fixed
- pruning: cutoff partial solutions, retaining only primitive dominant ones
- feasibility checks: nec. conditions and constraint programming
- acceleration techniques: dual cuts (specialization of Irnich and Gschwind '16)

Feasibility checks:

- residual capacity after chain fixing
- "large bin" capacity check
- multiple-subset-sum chain capacity check
- no split-items subset-sum capacity check

Overall algorithm (N)

Branch-and-bound, exploiting:

- bounding: compute valid lower bounds
- **approximation: compute upper bounds**
- branching: (a) fix split / non split items; (b) fix *pairs* of split items for the same bin; (c) assign items to bins whose pairs of split items are fixed
- pruning: cutoff partial solutions, retaining only primitive dominant ones
- feasibility checks: nec. conditions and constraint programming
- acceleration techniques: dual cuts (specialization of Irnich and Gschwind '16)

Feasibility checks:

- residual capacity after chain fixing
- "large bin" capacity check
- multiple-subset-sum chain capacity check
- no split-items subset-sum capacity check

Overall algorithm (N)

Branch-and-bound, exploiting:

- bounding: compute valid lower bounds
- approximation: compute upper bounds
- **branching: (a) fix split / non split items; (b) fix *pairs* of split items for the same bin; (c) assign items to bins whose pairs of split items are fixed**
- pruning: cutoff partial solutions, retaining only primitive dominant ones
- feasibility checks: nec. conditions and constraint programming
- acceleration techniques: dual cuts (specialization of Irnich and Gschwind '16)

Feasibility checks:

- residual capacity after chain fixing
- "large bin" capacity check
- multiple-subset-sum chain capacity check
- no split-items subset-sum capacity check

Overall algorithm (N)

Branch-and-bound, exploiting:

- bounding: compute valid lower bounds
- approximation: compute upper bounds
- branching: (a) fix split / non split items; (b) fix *pairs* of split items for the same bin; (c) assign items to bins whose pairs of split items are fixed
- **pruning: cutoff partial solutions, retaining only primitive dominant ones**
- feasibility checks: nec. conditions and constraint programming
- acceleration techniques: dual cuts (specialization of Irnich and Gschwind '16)

Feasibility checks:

- residual capacity after chain fixing
- "large bin" capacity check
- multiple-subset-sum chain capacity check
- no split-items subset-sum capacity check

Overall algorithm (N)

Branch-and-bound, exploiting:

- bounding: compute valid lower bounds
- approximation: compute upper bounds
- branching: (a) fix split / non split items; (b) fix *pairs* of split items for the same bin; (c) assign items to bins whose pairs of split items are fixed
- pruning: cutoff partial solutions, retaining only primitive dominant ones
- **feasibility checks: nec. conditions and constraint programming**
- acceleration techniques: dual cuts (specialization of Irnich and Gschwind '16)

Feasibility checks:

- residual capacity after chain fixing
- "large bin" capacity check
- multiple-subset-sum chain capacity check
- no split-items subset-sum capacity check

Overall algorithm (N)

Branch-and-bound, exploiting:

- bounding: compute valid lower bounds
- approximation: compute upper bounds
- branching: (a) fix split / non split items; (b) fix *pairs* of split items for the same bin; (c) assign items to bins whose pairs of split items are fixed
- pruning: cutoff partial solutions, retaining only primitive dominant ones
- feasibility checks: nec. conditions and constraint programming
- **acceleration techniques: dual cuts (specialization of Irnich and Gschwind '16)**

Feasibility checks:

- residual capacity after chain fixing
- "large bin" capacity check
- multiple-subset-sum chain capacity check
- no split-items subset-sum capacity check

Overall algorithm (N)

Branch-and-bound, exploiting:

- bounding: compute valid lower bounds
- approximation: compute upper bounds
- branching: (a) fix split / non split items; (b) fix *pairs* of split items for the same bin; (c) assign items to bins whose pairs of split items are fixed
- pruning: cutoff partial solutions, retaining only primitive dominant ones
- feasibility checks: nec. conditions and constraint programming
- acceleration techniques: dual cuts (specialization of Irnich and Gschwind '16)

Feasibility checks:

- residual capacity after chain fixing
- "large bin" capacity check
- multiple-subset-sum chain capacity check
- no split-items subset-sum capacity check

Focus: computing valid lower bounds (N)

Observation: **CC '14** In primitive solutions, minimizing the number of fragments, fragmentations or fragmented items is equivalent.

Compact model

$$\begin{aligned}
\min \quad & \sum_{i \in I, j \in B} z_{ij} - |I| & \\
\text{s.t.} \quad & \sum_{j \in B} x_{ij} = 1 & \forall i \in I \\
& \sum_{i \in I} d_i x_{ij} \leq C & \forall j \in B \\
& x_{ij} \leq z_{ij} & \forall i \in I, \forall j \in B \\
& 0 \leq x_{ij} \leq 1 & \forall i \in I, \forall j \in B \\
& z_{ij} \in \{0, 1\} & \forall i \in I, \forall j \in B
\end{aligned}$$

Extended formulation (MP)

$$\begin{aligned}
\min \quad & \sum_{k \in K, i \in I} \bar{z}_i^k y^k - |I| & & \\
\text{s.t.} \quad & \sum_{k \in K} \bar{x}_i^k y^k = 1 & \forall i \in I & \quad \lambda_i \\
& \sum_{k \in K} \bar{z}_i^k y^k \leq 2 & \forall i \in I & \quad \mu_i \\
& \sum_{k \in K} y^k \leq |B| & & \quad \eta \\
& y^k \in \mathbb{B} & \forall k \in K &
\end{aligned}$$

Focus: computing valid lower bounds (N)

Observation: CC '14 In primitive solutions, minimizing the number of fragments, fragmentations or fragmented items is equivalent.

Compact model

$$
\begin{aligned}
\min \quad & \sum_{i \in I, j \in B} z_{ij} - |I| & \\
\text{s.t.} \quad & \sum_{j \in B} x_{ij} = 1 & \forall i \in I \\
& \sum_{i \in I} d_i x_{ij} \leq C & \forall j \in B \\
& x_{ij} \leq z_{ij} & \forall i \in I, \forall j \in B \\
& 0 \leq x_{ij} \leq 1 & \forall i \in I, \forall j \in B \\
& z_{ij} \in \{0, 1\} & \forall i \in I, \forall j \in B
\end{aligned}
$$

Extended formulation (MP)

$$
\begin{aligned}
\min \quad & \sum_{k \in K, i \in I} \bar{z}_i^k y^k - |I| & & \\
\text{s.t.} \quad & \sum_{k \in K} \bar{x}_i^k y^k = 1 & \forall i \in I & \quad \lambda_i \\
& \sum_{k \in K} \bar{z}_i^k y^k \leq 2 & \forall i \in I & \quad \mu_i \\
& \sum_{k \in K} y^k \leq |B| \quad \eta & & \\
& 0 \leq y^k \leq 1 & \forall k \in K &
\end{aligned}
$$

Column generation (N)

The reduced cost of a variable y^k is:

$$\bar{c}^k = \sum_{i \in I} \bar{z}_i^k - \sum_{i \in I} \lambda_i \cdot \bar{x}_i^k - \sum_{i \in I} \mu_i \cdot \bar{z}_i^k - \eta$$

$$\rightarrow \bar{c}^k = \sum_{i \in I} (1 - \mu_i) \bar{z}_i^k - \sum_{i \in I} \lambda_i \cdot \bar{x}_i^k - \eta$$

and a pattern is feasible if

$$\begin{aligned} &\sum_{i \in I} d_i \cdot \bar{x}_i^k \leq C && \\ &0 \leq \bar{x}_i^k \leq z_i^k && \forall i \in I \\ &\bar{z}_i^k \in \{0, 1\} && \forall i \in I \end{aligned}$$

A Fractional Knapsack Problem with Penalties (FKPP)

Theorem (CC '14)

An optimal FKPP solution always exists, containing at most one fragmented item.

Pricing Algorithm (N) - Idea

To solve the FKPPs:

- consider 0 fragmented items $\rightarrow$ a 0–1 KP
- consider 1 fragmented item, and choose it explicitly $\rightarrow$ use KP recursion, and then fill optimally.

Pricing Algorithm (N) - Details

$$\min \bar{c}^k = \sum_{i \in I} (1 - \mu_i) \bar{z}_i^k - \sum_{i \in I} \lambda_i \cdot \bar{x}_i^k - \eta$$

$$s.t. \sum_{i \in I} d_i \cdot \bar{x}_i^k \leq C$$

$$0 \leq \bar{x}_i^k \leq \bar{z}_i^k \qquad \forall i \in I$$

$$\bar{z}_i^k \in \{0, 1\} \qquad \forall i \in I$$

Pricing algorithm:

(a) no fragmented items: $\bar{x}_i^k = \bar{z}_i^k$

$$\min \bar{c}^k = \sum_{i \in I} (1 - \mu_i - \lambda_i) \bar{z}_i^k - \eta$$

$$s.t. \sum_{i \in I} d_i \cdot \bar{z}_i^k \leq C$$

$$\bar{z}_i^k \in \{0, 1\} \qquad \forall i \in I$$

(b) one fragmented item: for each $i \in I$

Pricing Algorithm (N) - Details

$$
\begin{aligned}
\min \bar{c}^k &= \sum_{i \in I} (1 - \mu_i) \bar{z}_i^k - \sum_{i \in I} \lambda_i \cdot \bar{x}_i^k - \eta \\
s.t. & \sum_{i \in I} d_i \cdot \bar{x}_i^k \leq C \\
& 0 \leq \bar{x}_i^k \leq \bar{z}_i^k & \forall i \in I \\
& \bar{z}_i^k \in \{0, 1\} & \forall i \in I
\end{aligned}
$$

Pricing algorithm:

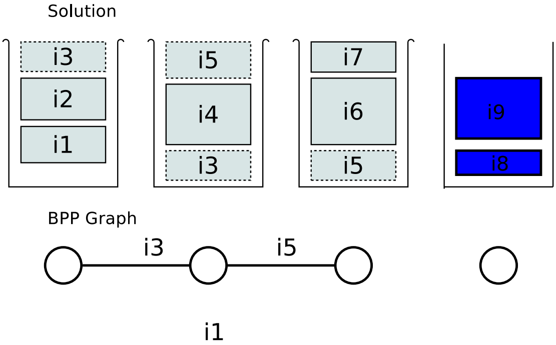
(a) no fragmented items: $\bar{x}_i^k = \bar{z}_i^k \rightarrow \bar{c}^k = \sum_{i \in I} (1 - \mu_i - \lambda_i) \bar{z}_i^k - \eta$

(b) one fragmented item: for each $i \in I$

- assume i is the (unique) fragmented item ($\bar{x}_j^k = \bar{z}_j^k \quad \forall j \in I : j \neq i$)
- solve a binary KP using traditional DP recursion, skipping i, and obtain $f_c \forall c = 0 \ldots C$
- take

$$
\min_{c = 1 \ldots d_i - 1} \left\{ f_{C-c} - \lambda_i \cdot c \cdot \frac{1}{d_i} \right\}
$$

Structure of a solution



Observation: **CC '15** Once items are organized in chains, a feasible fm-BPP-IF solution (if any exists) can be found by solving a Bin Packing feasibility problem ⇒ search for optimal packing in *chains* instead of optimal packing in *bins*.

BPP-IF models

Compact model

$$
\begin{aligned}
\min \quad & \sum_{i \in I, j \in B} z_{ij} - |I| & \\
\text{s.t.} \quad & \sum_{j \in B} x_{ij} = 1 & \forall i \in I \\
& \sum_{i \in I} d_i x_{ij} \leq C & \forall j \in B \\
& x_{ij} \leq z_{ij} & \forall i \in I, \forall j \in B \\
& 0 \leq x_{ij} \leq 1 & \forall i \in I, \forall j \in B \\
& z_{ij} \in \{0,1\} & \forall i \in I, \forall j \in B
\end{aligned}
$$

Chain based model

$$
\begin{aligned}
\min \quad & \sum_{k \in K} l_k - b_k & \\
\text{s.t.} \quad & \sum_{k \in K} z_{ik} = 1 & \forall i \in I \\
& \sum_{i \in I} d_i z_{ik} \leq C \cdot l_k & \forall k \in K \\
& \sum_{k \in K} l_k \leq |B| & \\
& b_k \leq l_k & \forall k \in K \\
& b_k \in \{0,1\}, l_k \in \mathbb{Z}_+ & \forall k \in K \\
& z_{ij} \in \{0,1\} \quad \forall i \in I, & \forall k \in K
\end{aligned}
$$

BPP-IF models

Chain-based model

$$
\begin{aligned}
\min \quad & \sum_{k \in K} l_k - b_k \\
\text{s.t.} \quad & \sum_{k \in K} z_{ik} = 1 && \forall i \in I \\
& \sum_{i \in I} d_i z_{ik} \leq C \cdot l_k && \forall k \in K \\
& \sum_{k \in K} l_k \leq |B| \\
& b_k \leq l_k && \forall k \in K \\
& b_k, l_k \in \mathbb{Z}_+ && \forall k \in K \\
& z_{ij} \in \{0, 1\} && \forall i \in I, \forall k \in K
\end{aligned}
$$

Chain-based extended formulation (MP)

$$
\begin{aligned}
\min \quad & \sum_{p \in \Omega, i \in I} (\bar{l}^p - 1) \cdot y^p \\
\text{s.t.} \quad & \sum_{p \in \Omega} \bar{z}_i^p \cdot y^p = 1 && \forall i \in I \quad \lambda_i \\
& \sum_{p \in \Omega} \bar{l}^p \cdot y^p \leq |B| \quad \eta \\
& y^p \in \mathbb{B} && \forall p \in \Omega
\end{aligned}
$$

BPP-IF models

Chain-based model

$$
\begin{aligned}
\min \quad & \sum_{k \in K} l_k - b_k & \\
\text{s.t.} \quad & \sum_{k \in K} z_{ik} = 1 & \forall i \in I \\
& \sum_{i \in I} d_i z_{ik} \leq C \cdot l_k & \forall k \in K \\
& \sum_{k \in K} l_k \leq |B| & \\
& b_k \leq l_k & \forall k \in K \\
& b_k, l_k \in \mathbb{Z}_+ & \forall k \in K \\
& z_{ij} \in \{0,1\} & \forall i \in I, \forall k \in K
\end{aligned}
$$

Chain-based extended formulation (MP)

$$
\begin{aligned}
\min \quad & \sum_{p \in \Omega, i \in I} (\bar{l}^p - 1) \cdot y^p & & \\
\text{s.t.} \quad & \sum_{p \in \Omega} \bar{z}_i^p \cdot y^p = 1 & \forall i \in I & \quad \lambda_i \\
& \sum_{p \in \Omega} \bar{l}^p \cdot y^p \leq |B| & & \quad \eta \\
& 0 \leq y^p \leq 1 & \forall p \in \Omega &
\end{aligned}
$$

Column generation (C)

The reduced cost of a variable y^p is:

$$\bar{c}^p = (\bar{l}^p - 1) - \sum_{i \in I} \lambda_i \cdot \bar{z}_i^p + \bar{l}^p \cdot \eta$$

$$\rightarrow \bar{c}^p = -1 + (1 + \eta) \cdot \bar{l}^p - \sum_{i \in I} \lambda_i \cdot \bar{z}_i^p$$

and a pattern is feasible if

$$\sum_{i \in I} d_i \cdot \bar{z}_i^p \leq C \cdot \bar{l}^p$$

$$\bar{z}_i^p \in \{0, 1\} \qquad \forall i \in I$$

$$\bar{l}^p \in \mathbb{Z}_+$$

Observation

When $\bar{l}^p$ is fixed, the pricing problem is a 0–1 Knapsack Problem (KP)

$\rightarrow$ a *Variable Size* 0–1 Knapsack Problem (VSKP)

Pricing algorithm

Observation

The VSKP can be solved in pseudo-linear time.

Exact algorithm:

- let U (resp. L) be an upper (resp. lower) bound on the length of a chain;
- sort I;
- let $f(n, c)$ be opt. KP subproblem solution;
- compute $f(|I|, U \cdot C)$ using KP Dynamic Programming recursion;
- for $\ell = L \dots U$ set $\pi_\ell = (1 + \eta) \cdot \bar{l}^p - f(|I|, C \cdot \ell)$;
- select $\bar{l}^p = \text{argmin}_{\ell = L \dots U} \pi_\ell$;
- select $\bar{z}_i^p$ accordingly.

Exact Algortihm (C)

Tree search simplifies as well:

- elect *leading items*, one for each chain;
- fix/forbid assignments to specific chains;
- leading items make chains asymmetric.

Experimental setting

Implementation:

- Algorithms coded in C++
- SCIP 3 as framework for branch-and-price
- IBM ILOG CPLEX 12.3 for LP subproblems
- Hardware: 3.0GHz CPU, 4GB RAM

Dataset 0: industrial instances.
Competitors:

- IBM ILOG CPLEX 12.3 ILP Solver
- Exact Algo (Nat)
- Exact Algo (Chain)

Overall comparison

- Dataset 1 (180 instances, different size distrib., different available cap.)
- Number of inst. solved to proven optimality within 1h CPU time (fm):

$\|I\|$	CPLEX (N)	CPLEX (C)	Exact Algo (N)	Exact Algo (C)
10	56	60	60	60
50	37	58	57	60
100	20	33	36	60

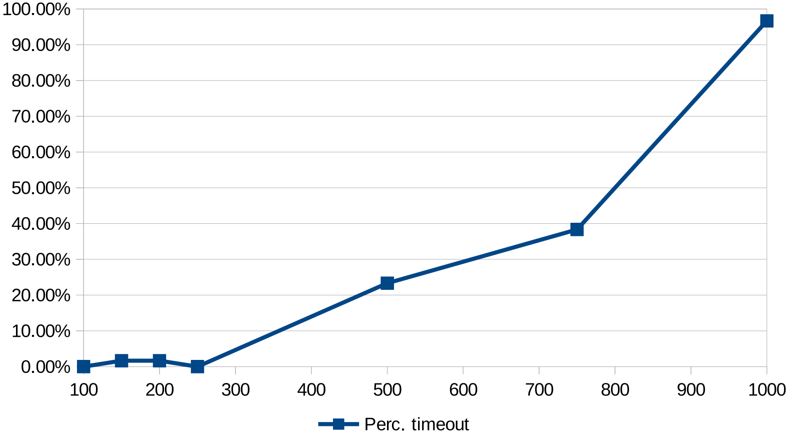
- Number of inst. solved to proven optimality within 1h CPU time (bm):

$\|I\|$	CPLEX (N)	CPLEX (C)	Exact Algo (C)
10	11	15	60
50	10	10	60
100	0	4	60

- Exact Algo (C) computing time: always less than 1 minute.

Stress test

Dataset 2 (360 instances, different size distrib., different available cap.)
Bin Minimization



Conclusions

Main conclusions:

- branch-and-price much more effective than CPLEX
- fragmentation minimization harder than bin minimization
- 1%C size increasing yields up to 4% cost increase in our (bm) instances (and scales linearly)
- for branch-and-price size increasing is not substantially harder to handle

In BPPIFs, be ready to pay an additional computing effort at a pricing stage, but avoid fractional decisions at a master stage.

Follow-up

Lately we:

- extended to variable size and cost packing
- applied similar ideas to **bike sharing systems**

Currently

Research on shortest path problems for hybrid vehicles.