Exact Methods for Recursive Circle Packing: "price-and-verify" with nonlinear subproblems

Ambros Gleixner, Benjamin Müller, Stephen J. Maher, João P. Pedroso

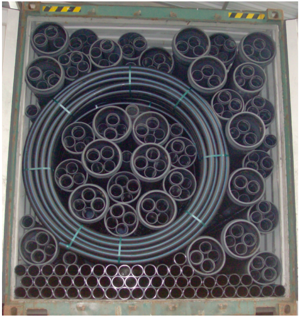
Zuse Institute Berlin, gleixner@zib.de

SCIP Workshop 2018 · RWTH Aachen · March 8

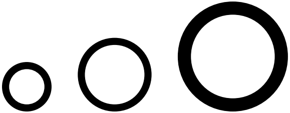
SPONSORED BY THE
Federal Ministry of Education and Research

Motivation

- reduce large shipping costs in tube production industry
- tubes are cut to the length of the container $\rightarrow$ 2-dim problem
- recursive packing of tubes into containers



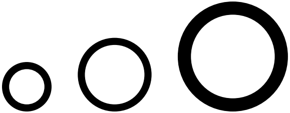
Variants of Recursive Circle Packing

ext. / int. radii 

demands 13 5 2

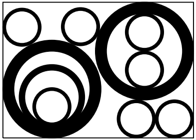
Variants of Recursive Circle Packing

ext. / int. radii



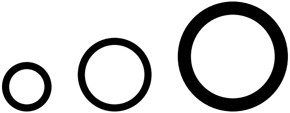
demands 13 5 2

maximize load of a single container



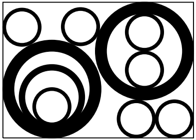
Variants of Recursive Circle Packing

ext. / int. radii

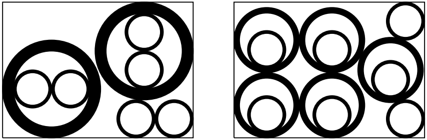


demands 13 5 2

maximize load of a single container

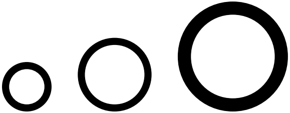


minimize total number of containers



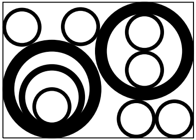
Variants of Recursive Circle Packing

ext. / int. radii

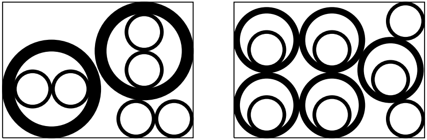


demands 13 5 2

maximize load of a single container



minimize total number of containers



Recursive Circle Packing Problem (RCPP)

input . . .

- set of ring types $\mathcal{T} := \{1, \ldots, T\}$
- external radius r_t^{ext} and internal radius r_t^{int} for each $t \in \mathcal{T}$
- demand $D_t \in \mathbb{N}$ for each $t \in \mathcal{T}$
- infinitely many rectangles with width W and height H

Recursive Circle Packing Problem (RCPP)

input . . .

- set of ring types $\mathcal{T} := \{1, \ldots, T\}$
- external radius r_t^{ext} and internal radius r_t^{int} for each $t \in \mathcal{T}$
- demand $D_t \in \mathbb{N}$ for each $t \in \mathcal{T}$
- infinitely many rectangles with width W and height H

objective . . .

- minimize $z^* =$ number of rectangles
- s.t. $\bigcup_t \{D_t$ many rings of type $t\}$ are packable into z^* rectangles

Related Work

CPP extensively studied ...

- sphere packing [Kepler 1611]
- packing ...
  - identical circles [Goldberg 1970, Locatelli, Raber 2002, ...]
  - different circles [George, George, Lamaer 1995, Huang et al. 2004, Kallrath 2007, ...]
  - ellipsoids [Kallrath 2015]
- recent surveys [Castillo, Kampas, and Pintér 2008, Hifi and M'Hallah 2009]

RCPP has been introduced by [Pedroso, Cunha, Tavares 2013]

- compact nonconvex MINLP formulation
- randomized greedy heuristic (GRASP)

Compact MINLP Formulation

- consider each ring individually
- (x_i, y_i) center of ring i
- $u_{i,j} \in \{0, 1\}$ if ring i is directly placed inside ring j
- $w_{i,r} \in \{0, 1\}$ if ring i is directly placed inside rectangle r
- $\phi_r \in \{0, 1\}$ if rectangle r is used

Compact MINLP Formulation

- consider each ring individually
- (x_i, y_i) center of ring i
- $u_{i,j} \in \{0,1\}$ if ring i is directly placed inside ring j
- $w_{i,r} \in \{0,1\}$ if ring i is directly placed inside rectangle r
- $\phi_r \in \{0,1\}$ if rectangle r is used

$$\min \sum_r \phi_r$$

$$\left\| \begin{pmatrix} x_i \\ y_i \end{pmatrix} - \begin{pmatrix} x_j \\ y_j \end{pmatrix} \right\|^2 \geq (r_i^{ext} + r_j^{ext})^2 (w_{i,r} + w_{j,r} - 1) \qquad \forall i \neq j \; \forall r$$

$$\left\| \begin{pmatrix} x_i \\ y_i \end{pmatrix} - \begin{pmatrix} x_j \\ y_j \end{pmatrix} \right\|^2 \geq (r_i^{ext} + r_j^{ext})^2 (u_{i,k} + u_{j,k} - 1) \qquad \forall i \neq j \neq k$$

$$\vdots$$

Compact MINLP Formulation

- consider each ring individually
- (x_i, y_i) center of ring i
- $u_{i,j} \in \{0,1\}$ if ring i is directly placed inside ring j
- $w_{i,r} \in \{0,1\}$ if ring i is directly placed inside rectangle r
- $\phi_r \in \{0,1\}$ if rectangle r is used

$$\min \sum_r \phi_r$$

$$\left\| \begin{pmatrix} x_i \\ y_i \end{pmatrix} - \begin{pmatrix} x_j \\ y_j \end{pmatrix} \right\|^2 \geq (r_i^{ext} + r_j^{ext})^2 (w_{i,r} + w_{j,r} - 1) \qquad \forall i \neq j \, \forall r$$

$$\left\| \begin{pmatrix} x_i \\ y_i \end{pmatrix} - \begin{pmatrix} x_j \\ y_j \end{pmatrix} \right\|^2 \geq (r_i^{ext} + r_j^{ext})^2 (u_{i,k} + u_{j,k} - 1) \qquad \forall i \neq j \neq k$$

$$\vdots$$

Hopeless to solve with a general MINLP solver!

Outline

Classic Dantzig-Wolfe Decomposition

A Revised Dantzig-Wolfe Decomposition
- 1-Level Packings
- Combining Column Enumeration & Generation
- Primal and Dual Bounds: Dealing with Subproblem Timeouts

Implementation & Experimental Results

Conclusion

Outline

Classic Dantzig-Wolfe Decomposition

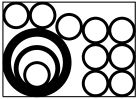
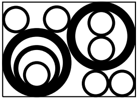
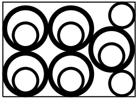
A Revised Dantzig-Wolfe Decomposition

- 1-Level Packings
- Combining Column Enumeration & Generation
- Primal and Dual Bounds: Dealing with Subproblem Timeouts

Implementation & Experimental Results

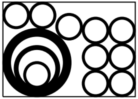
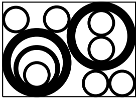
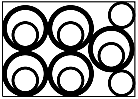
Conclusion

Classic Dantzig-Wolfe Decomposition

$\mathcal{P} = \{$  ,  ,  $, \ldots \}$

- $z_P \in \mathbb{Z}_+$ amount of packing $P \in \mathcal{P}$ used
- $\alpha_{t,P}$ number of rings of type $t \in \mathcal{T}$ in $P \in \mathcal{P}$

Classic Dantzig-Wolfe Decomposition

$\mathcal{P} = \{$  ,  ,  $, \dots \}$

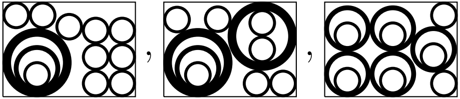
- $z_P \in \mathbb{Z}_+$ amount of packing $P \in \mathcal{P}$ used
- $\alpha_{t,P}$ number of rings of type $t \in \mathcal{T}$ in $P \in \mathcal{P}$

$$\min \sum_{P \in \mathcal{P}} z_P$$

$$\sum_{P \in \mathcal{P}} \alpha_{t,P} \cdot z_P \geq D_t \qquad \forall t \in \mathcal{T}$$

$$z_p \in \mathbb{Z}_+ \qquad \forall P \in \mathcal{P}$$

Classic Dantzig-Wolfe Decomposition

$\mathcal{P} = \{$  $, \ldots \}$

- $z_P \in \mathbb{Z}_+$ amount of packing $P \in \mathcal{P}$ used
- $\alpha_{t,P}$ number of rings of type $t \in \mathcal{T}$ in $P \in \mathcal{P}$

$$\min \sum_{P \in \mathcal{P}} z_P$$

$$\sum_{P \in \mathcal{P}} \alpha_{t,P} \cdot z_P \geq D_t \qquad \forall t \in \mathcal{T}$$

$$z_p \in \mathbb{Z}_+ \qquad \forall P \in \mathcal{P}$$

- similar to bin-packing reformulation: $|\mathcal{P}|$ exponentially large in $T = |\mathcal{T}|$

Column Generation

$$\min \sum_{P \in \mathcal{P}'} z_P$$

$$\sum_{P \in \mathcal{P}'} \alpha_{t,P} \cdot z_P \geq D_t \qquad \forall t \in \mathcal{T} \qquad \text{(demand)}$$

$$z_p \in \mathbb{Z}_+ \qquad \forall P \in \mathcal{P}'$$

Column Generation

$$\min \sum_{P \in \mathcal{P}'} z_P$$

$$\sum_{P \in \mathcal{P}'} \alpha_{t,P} \cdot z_P \geq D_t \qquad \forall t \in \mathcal{T} \qquad \text{(demand)}$$

$$z_p \in \mathbb{Z}_+ \qquad \forall P \in \mathcal{P}'$$

Pricing Problem (PP)

- $(\lambda_1, \ldots, \lambda_T)$ dual multipliers of demand constraints
- find $P \in \mathcal{P}$ with $1 - \sum_t \alpha_{t,P} \lambda_t < 0$

$\rightsquigarrow$ maximization version of RCPP

Column Generation

$$\min \sum_{P \in \mathcal{P}'} z_P$$

$$\sum_{P \in \mathcal{P}'} \alpha_{t,P} \cdot z_P \geq D_t \qquad \forall t \in \mathcal{T} \qquad \text{(demand)}$$

$$z_p \in \mathbb{Z}_+ \qquad \forall P \in \mathcal{P}'$$

Pricing Problem (PP)

- $(\lambda_1, \ldots, \lambda_T)$ dual multipliers of demand constraints
- find $P \in \mathcal{P}$ with $1 - \sum_t \alpha_{t,P} \lambda_t < 0$

$\rightsquigarrow$ maximization version of RCPP

Drawbacks

- recursive structure of RCPP is completely in PP
- hopeless to solve as nonconvex MINLP

Outline

Classic Dantzig-Wolfe Decomposition

A Revised Dantzig-Wolfe Decomposition
- 1-Level Packings
- Combining Column Enumeration & Generation
- Primal and Dual Bounds: Dealing with Subproblem Timeouts

Implementation & Experimental Results

Conclusion

1-Level Packings

Definition

A tuple $(d, t) \in \mathbb{Z}_+^T \times \mathcal{T}$ is a circular pattern :$\Leftrightarrow$

- $\bigcup_i \{d_i$ many **circles** with radius $r_i^{ext}\}$ packable inside a ring of type t

1-Level Packings

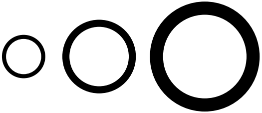
Definition

A tuple $(d, t) \in \mathbb{Z}_+^T \times \mathcal{T}$ is a circular pattern :⇔

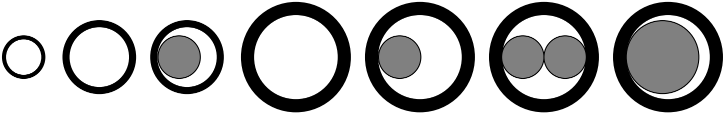
- $\bigcup_i \{d_i$ many **circles** with radius $r_i^{ext}\}$ packable inside a ring of type t

Example

input



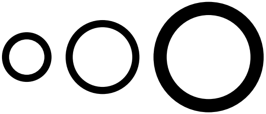
all circular patterns



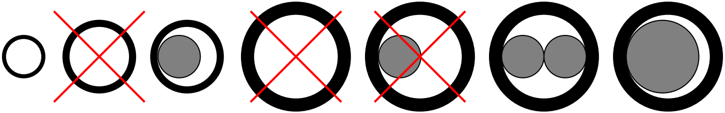
1-level Packings

Example

input



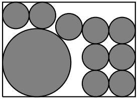
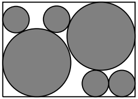
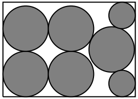
all circular patterns



- $\mathcal{CP}$:= set of non-dominated patterns
- $|\mathcal{CP}|$ depends on T, r^{ext}, r^{int}

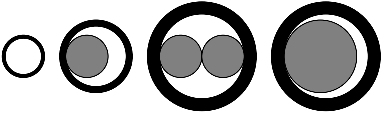
1-level Packings

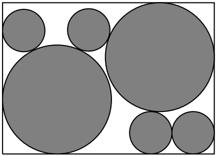
Analogous: $\mathcal{RP}$ the set of all rectangular patterns

$\mathcal{RP} = \{$  ,  ,  $, \dots\}$

- $\mathcal{RP}$ consists of vectors $d = (d_1, \dots, d_T) \in \mathbb{Z}_+^T$
- $|\mathcal{RP}|$ depends on T, r^{ext}, W, H

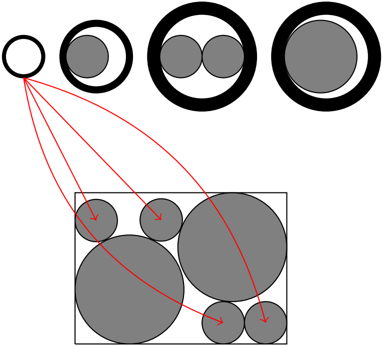
From 1-Level to Recursive Packings





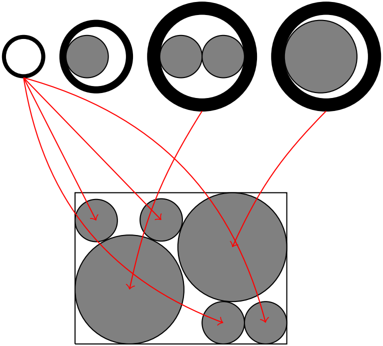
- idea: recursion on 1-level packings instead of rings
- inspired by multi-stage cutting stock formulation [Muter, Birbil, Bülbï 2012]

From 1-Level to Recursive Packings



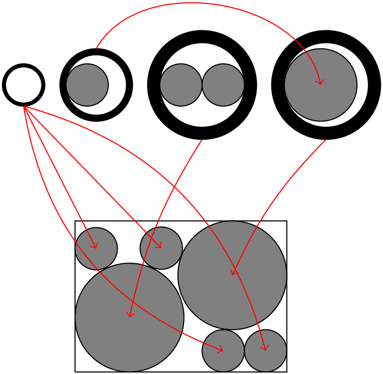
- idea: recursion on 1-level packings instead of rings
- inspired by multi-stage cutting stock formulation [Muter, Birbil, Bülbï 2012]

From 1-Level to Recursive Packings



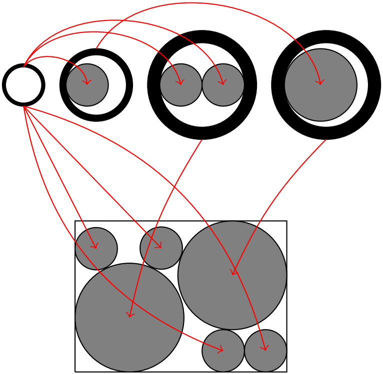
- idea: recursion on 1-level packings instead of rings
- inspired by multi-stage cutting stock formulation [Muter, Birbil, Bülbï 2012]

From 1-Level to Recursive Packings



- idea: recursion on 1-level packings instead of rings
- inspired by multi-stage cutting stock formulation [Muter, Birbil, Bülbï 2012]

From 1-Level to Recursive Packings



- idea: recursion on 1-level packings instead of rings
- inspired by multi-stage cutting stock formulation [Muter, Birbil, Bülbï 2012]

A Revised Dantzig-Wolfe Decomposition

- $z_C \in \mathbb{Z}_+$ for $C \in \mathcal{CP}$
- $z_R \in \mathbb{Z}_+$ for $R \in \mathcal{RP}$
- $\alpha_{t,R}$ / $\alpha_{t,C}$ number of circles of type t in pattern R / C

A Revised Dantzig-Wolfe Decomposition

- $z_C \in \mathbb{Z}_+$ for $C \in \mathcal{CP}$
- $z_R \in \mathbb{Z}_+$ for $R \in \mathcal{RP}$
- $\alpha_{t,R}$ / $\alpha_{t,C}$ number of circles of type t in pattern R / C

$$\min \sum_{R \in \mathcal{RP}} z_R$$

$$\sum_{C=(d,t) \in \mathcal{CP}} z_C \geq D_t \qquad \forall t \in \mathcal{T} \qquad \text{(demand)}$$

$$\sum_{C=(d,t) \in \mathcal{CP}} z_C \leq \sum_{R \in \mathcal{RP}} \alpha_{t,R} z_R + \sum_{C \in \mathcal{CP}} \alpha_{t,C} z_C \qquad \forall t \in \mathcal{T} \qquad \text{(packing)}$$

$$z_R \in \mathbb{Z}_+ \qquad \forall R \in \mathcal{RP}$$

$$z_C \in \mathbb{Z}_+ \qquad \forall C \in \mathcal{CP}$$

A Revised Dantzig-Wolfe Decomposition

- $z_C \in \mathbb{Z}_+$ for $C \in \mathcal{CP}$
- $z_R \in \mathbb{Z}_+$ for $R \in \mathcal{RP}$
- $\alpha_{t,R}$ / $\alpha_{t,C}$ number of circles of type t in pattern R / C

$$
\begin{aligned}
&\min \sum_{R \in \mathcal{RP}} z_R \\
&\sum_{C=(d,t) \in \mathcal{CP}} z_C \geq D_t && \forall t \in \mathcal{T} && \text{(demand)} \\
&\sum_{C=(d,t) \in \mathcal{CP}} z_C \leq \sum_{R \in \mathcal{RP}} \alpha_{t,R} z_R + \sum_{C \in \mathcal{CP}} \alpha_{t,C} z_C && \forall t \in \mathcal{T} && \text{(packing)} \\
&z_R \in \mathbb{Z}_+ && \forall R \in \mathcal{RP} \\
&z_C \in \mathbb{Z}_+ && \forall C \in \mathcal{CP}
\end{aligned}
$$

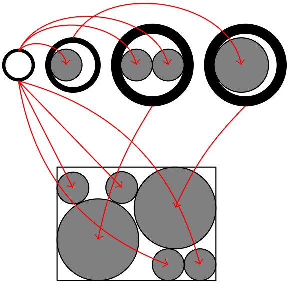
In practice: $\mathcal{CP}$ smaller than $\mathcal{RP}$

1. Column enumeration of $\mathcal{CP}$
2. Column generation over $\mathcal{RP}$

Combining Column Enumeration & Generation

Restricted Master Problem (RMP')

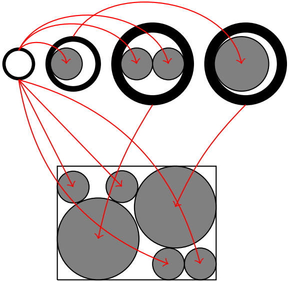
- circular patterns enumerated
- rectangular patterns generated
- recursiveness reduced to counting
  $\rightsquigarrow$ implicit symmetry breaking



Combining Column Enumeration & Generation

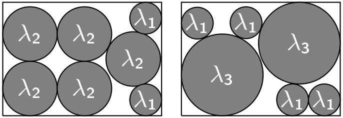
Restricted Master Problem (RMP')

- circular patterns enumerated
- rectangular patterns generated
- recursiveness reduced to counting
  $\rightsquigarrow$ implicit symmetry breaking



Pricing Problem (PP')

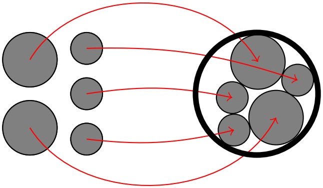
- maximum weight circle packing
- solve nonconvex MINLP formulation via spatial branch-and-bound
- Farley bound gives a proven dual bound even if not solved to optimality [Farley 1990]



Enumeration of $\mathcal{CP}$

Given a candidate $(d, t) \in \mathbb{Z}_+^T \times \mathcal{T}$, decide whether

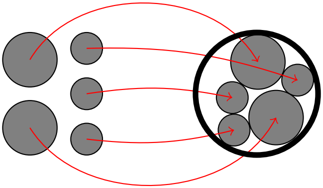
- pattern is packable: $(d, t) \in \mathcal{CP}_{feas}$
- pattern is infeasible: $(d, t) \in \mathcal{CP}_{infeas}$



Enumeration of $\mathcal{CP}$

Given a candidate $(d, t) \in \mathbb{Z}_+^T \times \mathcal{T}$, decide whether

- pattern is packable: $(d, t) \in \mathcal{CP}_{feas}$
- pattern is infeasible: $(d, t) \in \mathcal{CP}_{infeas}$



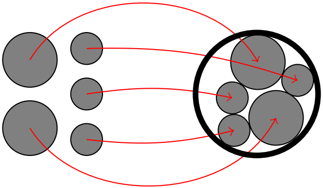
Solve nonconvex verification NLP: candidates with ...

- too few circles $\rightarrow$ feasible (heuristic) $\rightarrow$ easy
- too many circles $\rightarrow$ infeasible (NLP) $\rightarrow$ easy

Enumeration of $\mathcal{CP}$

Given a candidate $(d, t) \in \mathbb{Z}_+^T \times \mathcal{T}$, decide whether

- pattern is packable: $(d, t) \in \mathcal{CP}_{feas}$
- pattern is infeasible: $(d, t) \in \mathcal{CP}_{infeas}$



Solve nonconvex verification NLP: candidates with ...

- too few circles $\rightarrow$ feasible (heuristic) $\rightarrow$ easy
- too many circles $\rightarrow$ infeasible (NLP) $\rightarrow$ easy

Other candidates hard to verify: $(d, t) \in \mathcal{CP}_{unknown}$

- optimistic enumeration $\rightarrow$ dual bound via $\mathcal{CP}_{feas} \cup \mathcal{CP}_{unknown}$
- pessimistic enumeration $\rightarrow$ primal bound via $\mathcal{CP}_{feas}$

Outline

Classic Dantzig-Wolfe Decomposition

A Revised Dantzig-Wolfe Decomposition

- 1-Level Packings
- Combining Column Enumeration & Generation
- Primal and Dual Bounds: Dealing with Subproblem Timeouts

Implementation & Experimental Results

Conclusion

Implementation

- `cmain.c` $\leftarrow$ include plugins, add parameters, start shell
- `pattern.{h,c}`
- `reader_rpa.{h,c}`
- `probdata_rpa.{h,c}`
- `pricer_rpa.{h,c}`

Implementation

- `cmain.c` ← include plugins, add parameters, start shell
- `pattern.{h,c}` ← helpers for storing patterns

```
...
struct SCIP_Pattern
{
   BMS_BLKMEM*          blkmem;             /**< block memory */
   SCIP_PATTERNTYPE     patterntype;        /**< pattern type: circular, rectangular */
   SCIP_PACKABLE        packable;           /**< packable status: yes, no, unknown */
   SCIP_Real*           xs;                 /**< array containing the x-coordinate of eac
   SCIP_Real*           ys;                 /**< array containing the y-coordinate of eac
   int*                 types;              /**< array storing the type of each element *
   int                  size;               /**< size of types, xs, and ys arrays */
   int                  nelems;             /**< number of elements stored */
   int                  nlocks;             /**< number of locks */
   int                  type;               /**< type of the boundary circle */
};
typedef struct SCIP_Pattern SCIP_PATTERN;
...
```

- `reader_rpa.{h,c}`
- `probdata_rpa.{h,c}`
- `pricer_rpa.{h,c}`

Implementation

- `cmain.c` ← include plugins, add parameters, start shell
- `pattern.{h,c}` ← helpers for storing patterns
- `reader_rpa.{h,c}` ← read simple instance encoding

```
s03i1.rpa
3   10   11.4468
66   0.295319   0.342553
28   0.517447   0.599362
36   0.739574   0.85617
```

- `probdata_rpa.{h,c}`
- `pricer_rpa.{h,c}`

Implementation

- `cmain.c` ← include plugins, add parameters, start shell
- `pattern.{h,c}` ← helpers for storing patterns
- `reader_rpa.{h,c}` ← read simple instance encoding
- `probdata_rpa.{h,c}` ← create and store restricted master

```
struct SCIP_ProbData
{
   int*                demands;          /**< array of demands */
   SCIP_Real*          rints;            /**< internal radii of each ring */
   SCIP_Real*          rexts;            /**< external radii of each ring */
   int                 ntypes;           /**< number of different types */
   SCIP_Real           width;            /**< height of each rectangle */
   SCIP_Real           height;           /**< width of each rectangle */
   SCIP_CONS**         patternconss;     /**< packing constraints for each type */

   SCIP_PATTERN**      cpatterns;        /**< array containing all circular patterns *
   SCIP_VAR**          cvars;            /**< variables corresponding to circular patt
   int                 ncpatterns;       /**< total number of circular patterns */
   ...
   SCIP_PATTERN**      rpatterns;        /**< array containing all rectangular pattern
   SCIP_VAR**          rvars;            /**< variables corresponding to rectangular p
   int                 nrpatterns;       /**< total number of rectangular patterns */
   ...
   SCIP_Bool           isdualinvalid;    /**< whether the following reported dual boun
   SCIP_Real           dualbound;        /**< valid dual bound for RCPP instance */
   SCIP_RANDNUMGEN*    randnumgen;       /**< random number generator */
};
```

- `pricer_rpa.{h,c}`

Implementation

- `cmain.c` ← include plugins, add parameters, start shell
- `pattern.{h,c}` ← helpers for storing patterns
- `reader_rpa.{h,c}` ← read simple instance encoding
- `probdata_rpa.{h,c}` ← create and store restricted master

```
SCIP_RETCODE SCIPprobdataSetupProblem()

SCIP_RETCODE SCIPprobdataItemizePatterns()

void SCIPpackCirclesGreedy()

SCIP_RETCODE SCIPverifyCircularPatternHeuristic()

SCIP_RETCODE SCIPverifyCircularPatternNLP()

void SCIPprobdataUpdateDualbound()

void SCIPprobdataInvalidateDualbound()

...
```

- `pricer_rpa.{h,c}`

Implementation

- `cmain.c` $\leftarrow$ include plugins, add parameters, start shell
- `pattern.{h,c}` $\leftarrow$ helpers for storing patterns
- `reader_rpa.{h,c}` $\leftarrow$ read simple instance encoding
- `probdata_rpa.{h,c}` $\leftarrow$ create and store restricted master
- `pricer_rpa.{h,c}` $\leftarrow$ implement redcost pricing:

  a) randomized greedy packing heuristic

  b) exact pricing MINLP

  c) compute Farley bound

Implementation

- `cmain.c` $\leftarrow$ include plugins, add parameters, start shell
- `pattern.{h,c}` $\leftarrow$ helpers for storing patterns
- `reader_rpa.{h,c}` $\leftarrow$ read simple instance encoding
- `probdata_rpa.{h,c}` $\leftarrow$ create and store restricted master
- `pricer_rpa.{h,c}` $\leftarrow$ implement redcost pricing:

  a) randomized greedy packing heuristic

  b) exact pricing MINLP

  c) compute Farley bound

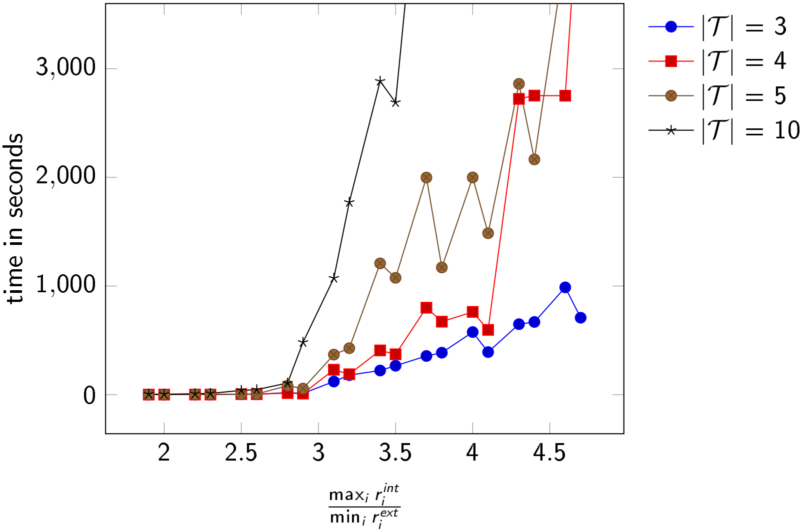
3 Phases

1. Column enumeration:
   create non-dominated circular patterns
2. Column generation for optimistic case:
   exact dual bound via $\mathcal{CP}_{feas} \cup \mathcal{CP}_{unknown}$
3. Price-and-branch for pessimistic case:
   exact primal bound via $\mathcal{CP}_{feas}$

Test Set

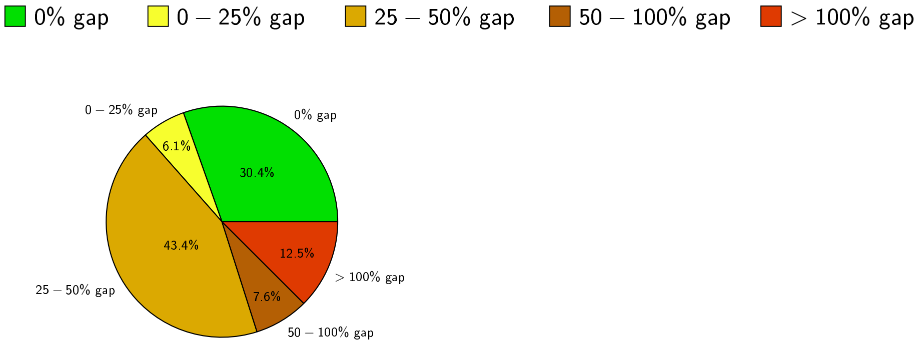
- contains 800 random generated instances
- $|\mathcal{T}| \in \{3, 4, 5, 10\}$
- $\frac{\max_i r_i^{int}}{\min_i r_i^{ext}} \in \{2.0, 2.3, \ldots, 5.0\}$
- $\frac{\max\{W,H\}}{\max_i r_i^{ext}} \in \{2.0, 2.3, \ldots, 5.0\}$
- $D_i \in \left[\frac{0.8 \cdot W \cdot H}{\pi (r_i^{ext})^2}, \frac{1.2 \cdot W \cdot H}{\pi (r_i^{ext})^2}\right] \cdot \gamma$ for $\gamma \in \{5, 10\}$

Results: Enumeration of $\mathcal{CP}$



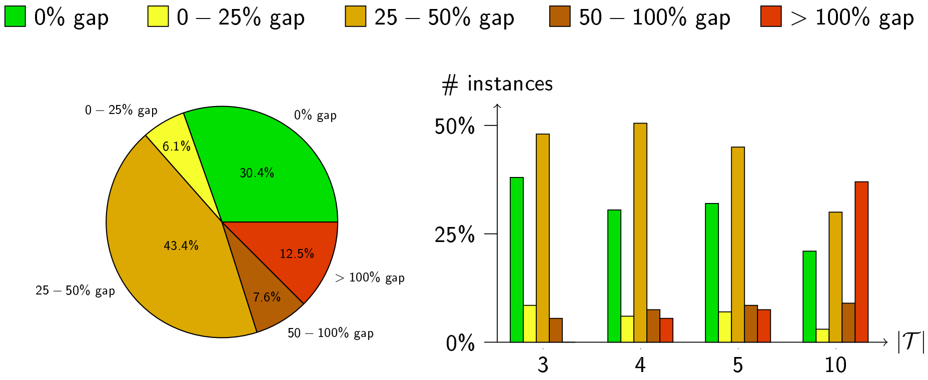
Results: Reached Gaps

- $gap = \frac{|primal - dual|}{\min\{primal, dual\}}$
- $dual$ = dual bound for $\mathcal{CP}_{LB}$
- $primal$ = primal bound for $\mathcal{CP}_{UB}$



Results: Reached Gaps

- $gap = \frac{|primal - dual|}{\min\{primal, dual\}}$
- $dual$ = dual bound for $\mathcal{CP}_{LB}$
- $primal$ = primal bound for $\mathcal{CP}_{UB}$



Results: Primal Bounds

- compare to randomized greedy heuristic GRASP [Pedroso, Cunha, Tavares 2013]
- time limit 3600s

$\lvert\mathcal{T}\rvert$	all (%)	better (#)	better (%)	worse (#)	worse (%)	no sol (#)
all	98.2	286	93.5	43	117.9	8
3	101.0	54	94.0	12	147.2	0
4	97.7	74	93.7	14	106.6	0
5	97.7	81	94.2	11	109.5	1
10	95.4	77	92.2	6	106.2	7

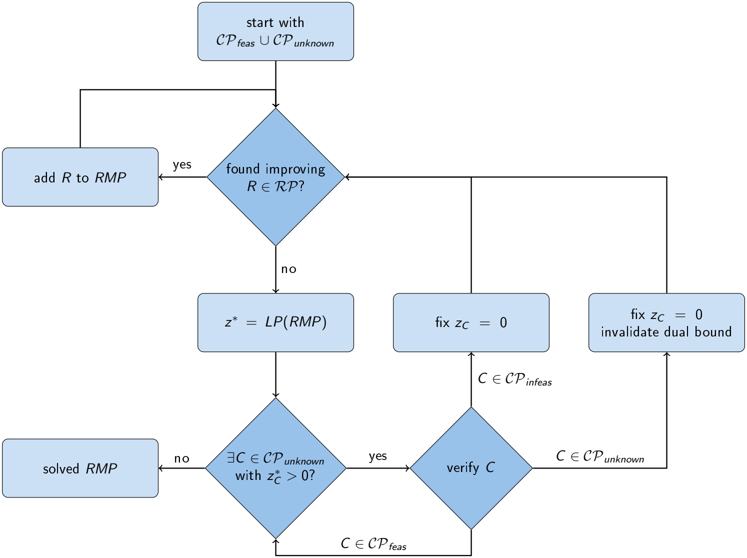
Results: Primal Bounds

- compare to randomized greedy heuristic GRASP [Pedroso, Cunha, Tavares 2013]
- time limit 3600s

$\lvert\mathcal{T}\rvert$	all (%)	better (#)	better (%)	worse (#)	worse (%)	no sol (#)
all	98.2	286	93.5	43	117.9	8
3	101.0	54	94.0	12	147.2	0
4	97.7	74	93.7	14	106.6	0
5	97.7	81	94.2	11	109.5	1
10	95.4	77	92.2	6	106.2	7

- better solutions on 35.8% on all instances
- worse solutions on 6.4% on all instances
- performs better if recursive part becomes more complicated

Outlook: "Price-and-Verify" with cons_rpa



Outline

Classic Dantzig-Wolfe Decomposition

A Revised Dantzig-Wolfe Decomposition

- 1-Level Packings
- Combining Column Enumeration & Generation
- Primal and Dual Bounds: Dealing with Subproblem Timeouts

Implementation & Experimental Results

Conclusion

Recursive Circle Packing Problem

- pack set of rings; minimize the number of rectangles
- generalization of Circle Packing Problem

Recursive Circle Packing Problem

- pack set of rings; minimize the number of rectangles
- generalization of Circle Packing Problem

Specialized Dantzig-Wolfe Reformulation

- recursive master problem
- nonconvex MINLP pricing

Recursive Circle Packing Problem

- pack set of rings; minimize the number of rectangles
- generalization of Circle Packing Problem

Specialized Dantzig-Wolfe Reformulation

- recursive master problem
- nonconvex MINLP pricing

First Exact Algorithm

- optimal solutions for many small and medium-sized instances
- improved primal solutions compared to existing heuristic

Recursive Circle Packing Problem

- pack set of rings; minimize the number of rectangles
- generalization of Circle Packing Problem

Specialized Dantzig-Wolfe Reformulation

- recursive master problem
- nonconvex MINLP pricing

First Exact Algorithm

- optimal solutions for many small and medium-sized instances
- improved primal solutions compared to existing heuristic

Further Details

- Gleixner, Müller, Maher, Pedroso, **Exact methods for recursive circle packing**, submitted to Annals of Operations Research, available as ZIB-Report 17-07, `http://nbn-resolving.de/urn:nbn:de:0297-zib-62039`