Exact methods for rectangle placement problems

François Claustiaux

Université des Sciences et Technologies de Lille
LIFL, UMR USTL/CNRS 8022
Equipe projet INRIA DOLPHIN

May 22, 2008

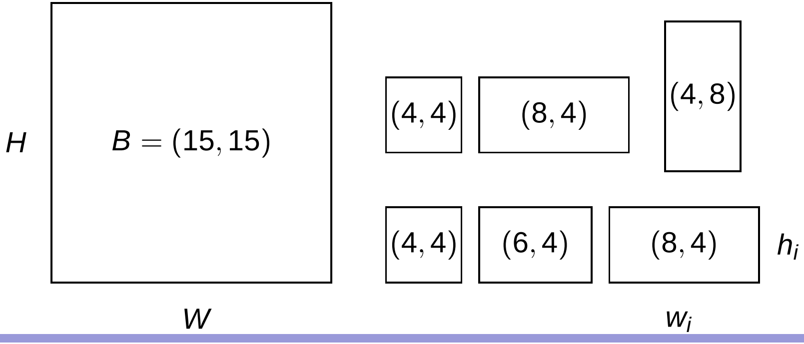
1. Introduction
2. Heuristics / feasibility tests
3. Exact methods
4. Guillotine-cutting problem
5. Summary and some issues

1. Introduction
2. Heuristics / feasibility tests
3. Exact methods
4. Guillotine-cutting problem
5. Summary and some issues

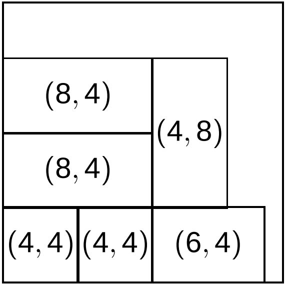
Rectangle placement problems

- Finding a placement in a large rectangle (bin) for a list of small rectangles (items)
- Cutting iron, wood, paper, packing containers in a cargo, ...
- A crucial subproblem in many two-dimensional cutting and packing problems (knapsack, bin- or strip-packing, ...)
- This problem is NP-complete

Rectangle placement



Rectangle placement



Presentation content

- Heuristic and initial feasibility tests
  - Heuristics and links with exact methods
  - Bin packing lower bounds as feasibility tests (links with LP)
- Exact methods for the rectangle placement problem
  - Several models and paradigms used
  - We will focus on a constraint-based scheduling model
  - Some feasibility tests
- The guillotine case
  - Several models
  - A new graph-theoretical model
  - A constraint-programming model

1. Introduction
2. Heuristics / feasibility tests
   - Heuristics
   - Initial feasibility tests
3. Exact methods
4. Guillotine-cutting problem
5. Summary and some issues

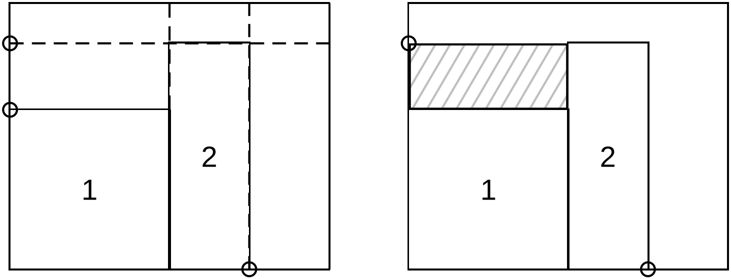
Simple heuristics

- for large instances, only heuristics can be used
- simple heuristics are efficient for many cases
- heuristics based on the bottom-left rule
- improving results using metaheuristics
- $\Longrightarrow$ for difficult instances, difficult to avoid an enumerative phase

Several types of methods

- direct placement methods with local search [Boschetti and Mingozzi, 2002, El Hayek et al., 2007, Neveu et al., 2008]
  - placement considering all remaining surfaces
  - placement with "masked" surfaces
- using the graph-theoretical model of [Fekete and Schepers, 1997]
  - a tabu-search determining the relative placement of the items in the bin [Crainic et al., 2003]

Two ways of packing items



Bin packing lower bounds as feasibility tests

- determining *a priori* in polynomial time cases where the rectangles cannot fit in the bin
- trivial rules (incompatible pairs, ...)
- lower bounds defined for 2D bin packing problems

Continuous bound

Formulation

$$\left\lceil \sum_{i \in I} \frac{w_i h_i}{WH} \right\rceil \leq 1$$

- The total surface of the rectangles has to be smaller than the surface of the bin
- Transformation of the instance that conserves the validity of a pattern
- The goal is to increase the continuous bound
- Using one-dimensional *dual-feasible functions*

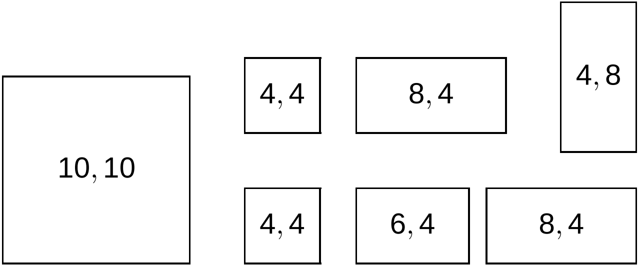
Dual feasible functions

A discrete dual-feasible function f is such that :

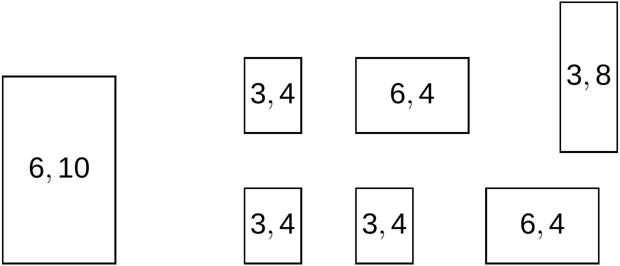
$$x_1 + x_2 + \ldots + x_k \leq X \Rightarrow f(x_1) + f(x_2) + \ldots + f(x_k) \leq f(X)$$

- Applying DFF on both dimensions of the instance [Fekete and Schepers, 1997]
- The continuous bound is still valid for the initial instance
- Many DFF are known and can be applied in turn (see [FC, Alves, Carvalho, 2008] for a survey on DFF)

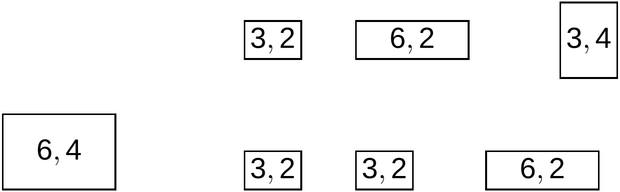
DFF and 2*BP*



DFF and 2BP



DFF and 2BP



Weakness of the feasibility tests known

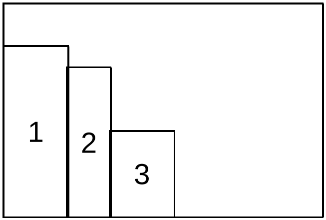
- in the end, only one-dimensional reasoning are used
- difficult to apply to a partial pattern (filtering)
- weaker when the number of bins in an optimal solution is small (here, one or two !)

1. Introduction
2. Heuristics / feasibility tests
3. Exact methods
   - State of the art
   - A scheduling-based algorithm
   - Feasibility tests
4. Guillotine-cutting problem
5. Summary and some issues

Three types of models

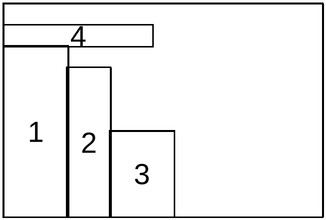
- Iterative placement of the rectangles
  - using the bottom-left rule [Hadjiconstantinou and Christofides, 1995], [Martello and Vigo, 1998]
  - using a discretization of the large rectangle (all positions tested in turn for each rectangle) [Beldiceanu and Carlsson, 2001]
- Relations between the rectangles
  - above/under/left/right [Pisinger and Sigurd, 2007]
  - graph-theoretical model [Fekete and Schepers, 1997]
- Scheduling-based algorithms
  - using a scheduling relaxation [FC, Carlier, Moukrim, 2006], [FC, Jouglet, Carlier, Moukrim, 2008]

Iterative placement of rectangles : the bottom-left rule [Hadjiconstantinou and Christofides,1995], [Martello and Vigo, 1998]



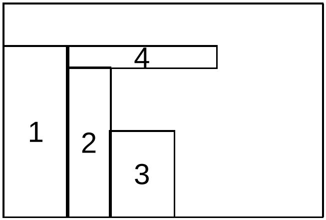
- **Pros** : does not depend on the size of the bin, a small computing time at each node
- **Cons** : symmetries, deductions during the search, difficulty to focus on "important" items

Iterative placement of rectangles : the bottom-left rule [Hadjiconstantinou and Christofides,1995], [Martello and Vigo, 1998]



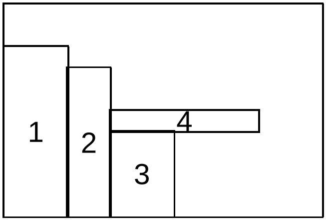
- **Pros** : does not depend on the size of the bin, a small computing time at each node
- **Cons** : symmetries, deductions during the search, difficulty to focus on "important" items

Iterative placement of rectangles : the bottom-left rule [Hadjiconstantinou and Christofides,1995], [Martello and Vigo, 1998]



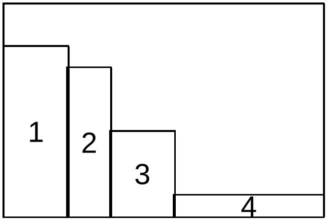
- **Pros** : does not depend on the size of the bin, a small computing time at each node
- **Cons** : symmetries, deductions during the search, difficulty to focus on "important" items

Iterative placement of rectangles : the bottom-left rule [Hadjiconstantinou and Christofides,1995], [Martello and Vigo, 1998]



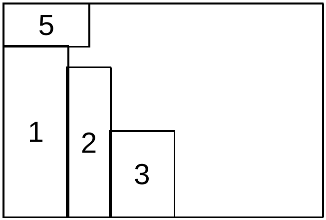
- **Pros** : does not depend on the size of the bin, a small computing time at each node
- **Cons** : symmetries, deductions during the search, difficulty to focus on "important" items

Iterative placement of rectangles : the bottom-left rule [Hadjiconstantinou and Christofides,1995], [Martello and Vigo, 1998]



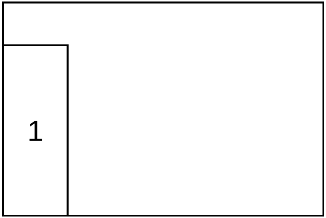
- **Pros** : does not depend on the size of the bin, a small computing time at each node
- **Cons** : symmetries, deductions during the search, difficulty to focus on "important" items

Iterative placement of rectangles : the bottom-left rule [Hadjiconstantinou and Christofides,1995], [Martello and Vigo, 1998]



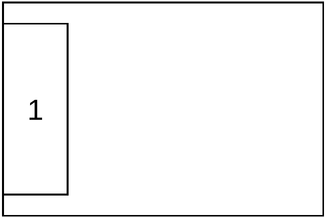
- **Pros** : does not depend on the size of the bin, a small computing time at each node
- **Cons** : symmetries, deductions during the search, difficulty to focus on "important" items

Iterative placement of rectangle : discretization of the bin [Beldiceanu and Carlsson, 2001, Korf, 2004]



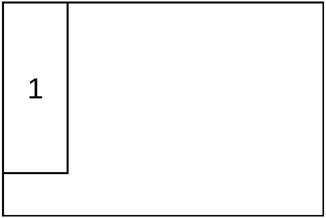
- **Pros** : deductions during the search, possibility to focus on "important" items
- **Cons** : depends on the size of the bin, symmetries

Iterative placement of rectangle : discretization of the bin [Beldiceanu and Carlsson, 2001, Korf, 2004]



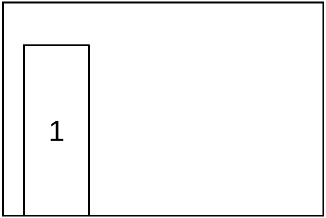
- **Pros** : deductions during the search, possibility to focus on "important" items
- **Cons** : depends on the size of the bin, symmetries

Iterative placement of rectangle : discretization of the bin [Beldiceanu and Carlsson, 2001, Korf, 2004]



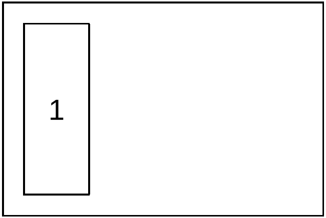
- **Pros** : deductions during the search, possibility to focus on "important" items
- **Cons** : depends on the size of the bin, symmetries

Iterative placement of rectangle : discretization of the bin [Beldiceanu and Carlsson, 2001, Korf, 2004]



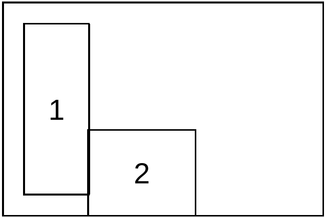
- **Pros** : deductions during the search, possibility to focus on "important" items
- **Cons** : depends on the size of the bin, symmetries

Iterative placement of rectangle : discretization of the bin [Beldiceanu and Carlsson, 2001, Korf, 2004]



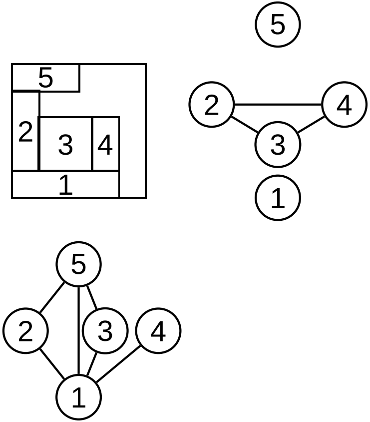
- **Pros** : deductions during the search, possibility to focus on "important" items
- **Cons** : depends on the size of the bin, symmetries

Iterative placement of rectangle : discretization of the bin [Beldiceanu and Carlsson, 2001, Korf, 2004]



- **Pros** : deductions during the search, possibility to focus on "important" items
- **Cons** : depends on the size of the bin, symmetries

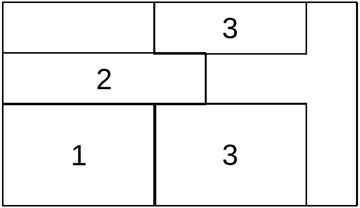
Graph-theoretical model [Fekete and Schepers, 1997]



- **Pros** : does not depend on the size of the bin, possibility to focus on "important" items, few symmetries
- **Cons** : deductions during the search, the remaining symmetries are hard to handle, a large computing time for each node

A 1.5packing relaxation [FC, Carlier, Moukrim, 2004]

- Replacing the vertical non-overlapping constraints by cumulative constraints
  - rectangles are *cut into horizontal strips*
  - they have to be packed at the same x-ordinate
  - *1.5packing* or *cut-packing* [Hoyland, 1988]

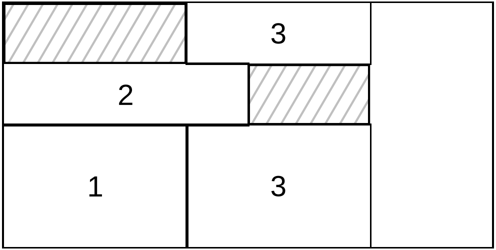


Using the relaxation in a two-phase method

- Seeking a solution for the cutpacking problem
- For each solution, checking if it corresponds with a feasible solution for the rectangle placement problem

First phase

- At each step j, we seek the set of rectangles to be packed at the current x-ordinate
- the quantity of lost area is recorded



The two-phase method : pros and cons

The method is efficient, since the relaxed problem is often unfeasible when the original problem is not feasible.
Better than bottom-left placement methods, and competitive with the graph-theoretical based method.

- **Pros** : strong relaxation : an easier problem, independent of the bin size, less symmetries
- **Cons** : some symmetries remain, difficulty to focus on the "important" rectangles, difficulty to make deductions at each step of the method

Toward a constraint-scheduling based method

- The two latter issues can be taken into account using a constraint-programming model.
- The 1.5packing problem is strictly equivalent to the cumulative scheduling problem.
- Many researchers have worked on constraint-based scheduling methods => using their work to improve the method

Basic model in constraint programming

- Variables :
  $\forall i \in I : X_i$ and Y_i coordinates of item i in the rectangle
- Domains :
  $\forall i \in I : D(X_i) = [X_i^{min}, X_i^{max}]$
  $\forall i \in I : D(X_i) = [X_i^{min}, X_i^{max}]$
- Constraints :
  $\forall i, k \in I, i \neq k$, $[X_i + w_i \leq X_k]$ or $[X_k + w_k \leq X_i]$ or $[Y_i + h_i \leq Y_k]$ or $[Y_k + h_k \leq Y_i]$
  (non-overlap constraint)

Two cumulative scheduling problems

The "horizontal" problem

- A resource R^H of capacity H
- n jobs A_i^w to schedule on R^H
  - computing time : w_i
  - demand on R^W : h_i

The "vertical" problem

- A resource R^W of capacity W
- n jobs A_i^h to schedule on R^W
  - computing time : h_i
  - demand on R^H : w_i

Scheduling-based model

Resource constraints

- $\forall t \in [0, W), \sum_{A_i^w \text{ s.t. } start(A_i^w) \leq t \leq start(A_i^w) + w_i} h_i \leq H$
- $\forall t \in [0, H), \sum_{A_i^h \text{ s.t. } start(A_i^h) \leq t \leq start(A_i^h) + h_i} w_i \leq W$

Link with the basic model

- $start(A_i^w) = X_i$
- $start(A_i^h) = Y_i$

Why using this model ?

- using the methods that already exist in constraint solvers (here we use ILOG scheduler)
- using propagation rules and operations research techniques that have been used for years on cumulative scheduling problems (see [Baptiste, Le Pape and Nuijten, 2001])

Branching scheme

We use depth-first search (schedule-or-postpone branching scheme)
At each step :

- we choose a non-scheduled job A and we fix its variable $start(A)$ to the smallest possible value t
- if the start time t leads to a fail, A will not be reconsidered before $D(start(A))$ is modified by another choice

Computational results

	OPP		TSBP		SWEEP		Sch. Model	
instance	nodes	cpu	nodes	cpu	nodes	cpu	nodes	cpu
00, N, 23	–	–	9057985	289	–	–	3675002	160
00, N, 23	–	–	5968406	86	–	–	1496331	70
05, N, 15	18369	2	1	0	111815	17	1	0
05, F, 20	547708	491	39387	2	78	0	27515	1
04, F, 20	22796	22	5876	3	238252	11	59795	3
10, N, 15	77	0	1	0	–	–	1	0
03, N, 16	9891	2	1592400	32	308003	62	14897	1
05, N, 17	1	0	993	1	175651	11	1673	0
03, F, 18	574	0	2605815	22	3205	0	1111	0
04, N, 18	24593	10	434824	7	3529193	329	1032	0
02, F, 20	–	–	487230	12	951640	42	2245	0
04, F, 17	20270	13	1942682	26	66	0	474	0
00, N, 15	127	0	96920	2	813351	136	610	0
20, F, 15	36	0	4355492	44	58	0	96	0
02, F, 22	190617	167	174943	4	4110	0	31	0
04, F, 19	786057	560	1075159	7	700	0	29	0
05, F, 18	262	0	20245458	126	149	0	25	0
08, F, 15	433	0	22658934	117	68	0	21	0
13, N, 15	1	0	91	0	102913	9	1	0
15, N, 15	1	0	1117	0	99905	6	1	0
Avg.	43521	33.53	1764380	19.73	179262.8	18.11	129527	6.09

Constraint-based scheduling method : pros and cons

- **Pros** : a strong relaxation, which leads to an easier problem, does not depend on the size of the bin, removes some symmetries, ability to focus on the "important" rectangles, some deductions at each step
- **Cons** : some symmetries remain, difficulty to make real two-dimensional deductions during the search

Feasibility tests

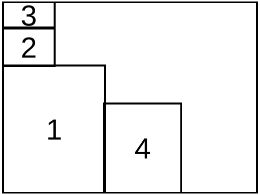
Algorithms to determine that a partial solution cannot lead to any valid solution.

- classical constraint-propagation strategies
- using DFF on a partial pattern
- two-dimensional energetic reasoning
- tests based on a knapsack model

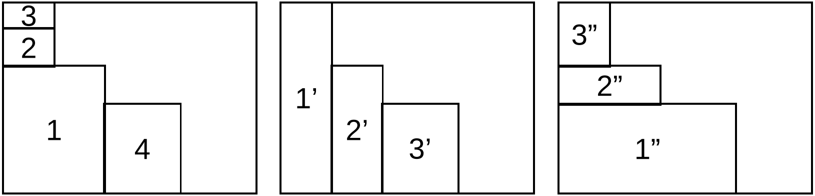
Applying DFF to a partial solution

- When rectangles are packed, their dimensions are not modified
- DFF will produce the same value as initially
- How can we exploit the additional information ?
  - aggregation of the rectangles into large rectangles

Applying DFF to a partial solution



Applying DFF to a partial solution



If one of the obtained instance is non-feasible, then the current pattern does not lead to any feasible solution.

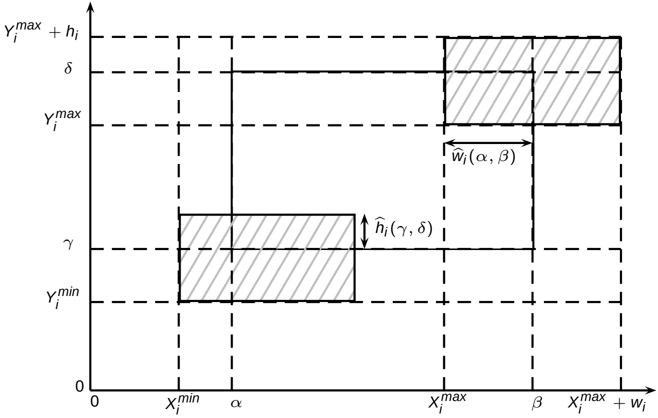
Energetic reasoning

Initially developped for cumulative scheduling problems [Erschler et al., 1991], [Lopez et al., 1992]
For a given time interval $[\alpha, \beta)$, the energy is produced by the resources and consumed by the jobs

- energy produced by a resource of capacity C : $(\beta - \alpha)C$
- minimum energy consumed by a task A : part of A that is scheduled in $[\alpha, \beta)$

Quantity of energy produced and consumed in several intervals
$\Longrightarrow$ feasibility tests and deductions

Two-dimensional energetic reasoning



Other feasibility tests for the mandatory parts

- The considered area is rectangular
- The mandatory parts of the rectangles are rectangular

$\Longrightarrow$ a new rectangle placement subproblem (similarly to [Néron *et al.*, 2006])

- Initial feasibility tests (DFF)
- Exact method ?

Solving a knapsack problem to perform feasibility tests

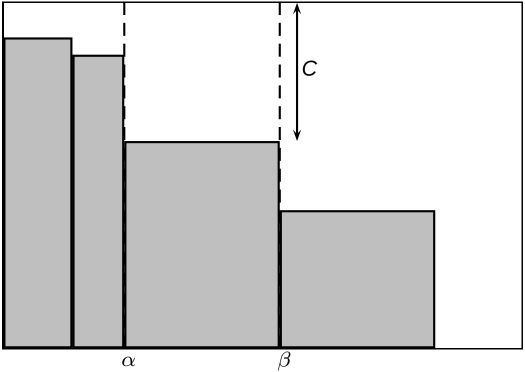
A specific knapsack problem :
Data : a set of values $c_1, \ldots, c_z$ and a value C
Determining the largest sum of values less than or equal to C

$$\max \left\{ \sum_{i=1}^{z} c_i \xi_i : \sum_{i=1}^{z} c_i \xi_i \leq C, \xi_i \in \{0, 1\} \right\}$$

Particular (and difficult) case of the knapsack problem, which can be solved in pseudo-polynomial time : $0(zC)$

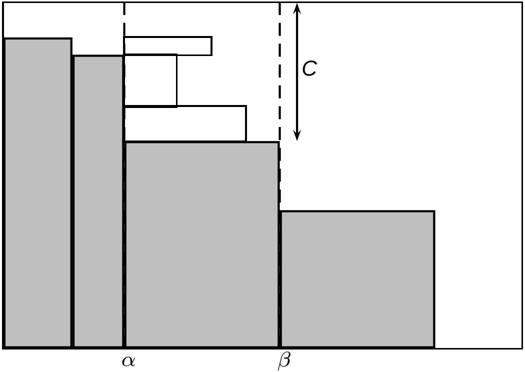
Solving a knapsack problem to perform feasibility tests

Computing an estimation of the lost area in the interval $[\alpha, \beta)$



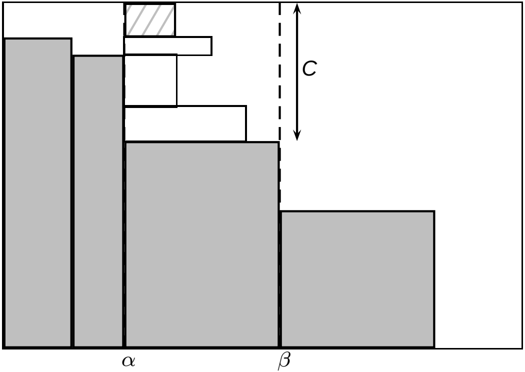
Solving a knapsack problem to perform feasibility tests

Computing an estimation of the lost area in the interval $[\alpha, \beta)$



Solving a knapsack problem to perform feasibility tests

Computing an estimation of the lost area in the interval $[\alpha, \beta)$



Computational experiments

			ER		KP		ER+KP		LB+ER(LB)+KP	
instance	nodes	cpu	nodes	cpu	nodes	cpu	nodes	cpu	nodes	cpu
00, N, 23	3675002	160	88446	6	50095	6	42207	6	32926	26
00, N, 23	1496331	70	7134	1	3208	1	3283	1	1731	1
05, F, 20	27515	1	27513	1	27515	1	27513	1	27513	1
04, F, 20	59795	3	9002	1	8336	1	5580	1	657	0
03, N, 16	14897	1	5183	1	8008	1	4743	1	3905	4
20, F, 15	7229	0	7229	0	6863	0	6863	0	27	0
04, N, 15	10181	0	4945	0	8002	1	4508	1	3286	3
03, N, 15	3562	0	3084	0	3085	0	2863	0	2217	2
10, N, 15	2622	0	2622	0	2437	0	2437	0	1844	1
07, N, 15	2149	0	2143	0	2133	0	2128	0	1966	1
05, N, 17	1673	0	1661	0	1649	0	1637	0	87	0
08, N, 15	1420	0	1420	0	1420	0	1420	0	1	0
03, F, 18	1111	0	1056	0	1060	0	1015	0	838	0
04, N, 18	1032	0	812	0	697	0	579	0	149	0
02, F, 20	2245	0	275	0	236	0	165	0	30	0
04, F, 15	287	0	286	0	285	0	285	0	284	0
03, N, 17	260	0	212	0	163	0	163	0	163	0
00, N, 15	610	0	102	0	96	0	89	0	80	0
05, F, 18	25	0	25	0	25	0	25	0	25	0
Avg.	129527	6.09	4029	0.61	3115	0.65	2670	0.65	1924	1.32

1 Introduction

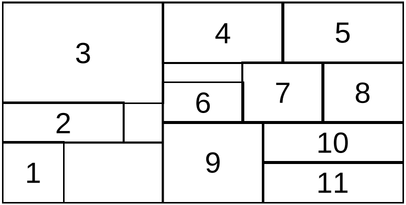
2 Heuristics / feasibility tests

3 Exact methods

4 Guillotine-cutting problem
- Problem description
- A new graph-theoretical model
- A constraint-programming based method

5 Summary and some issues

Guillotine-cutting problem



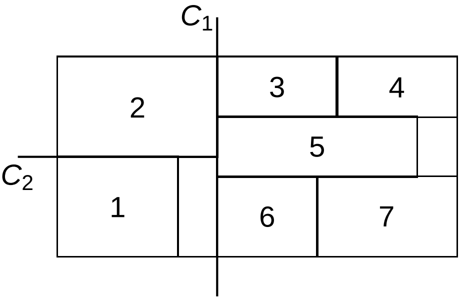
Guillotine-cutting problem

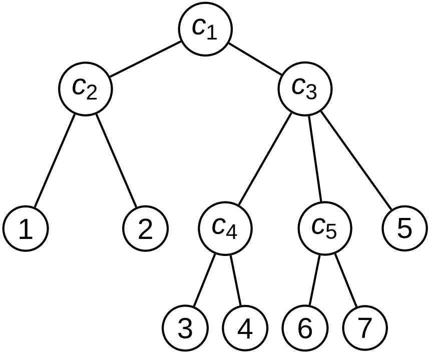
- Only guillotine cuts can be applied
- a dichotomic cutting pattern

Branch-and-bound algorithms

- Top-down algorithm [Christophides and Whitlock, 1995]
  - cuts are iteratively applied
  - the algorithm stops when all items are cut
- Bottom-up algorithm [Viswanathan and Bagchi, 1993]
  - items are gathered into horizontal and vertical *builds* [Wang, 1983]
  - the algorithm stops when there is only one remaining large item

A tree representation



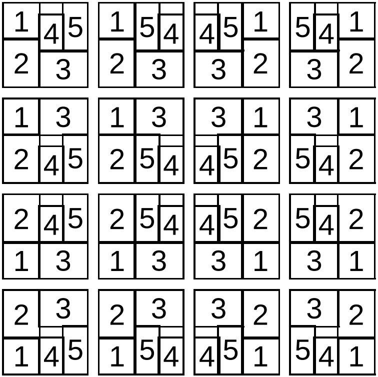


A tree representation

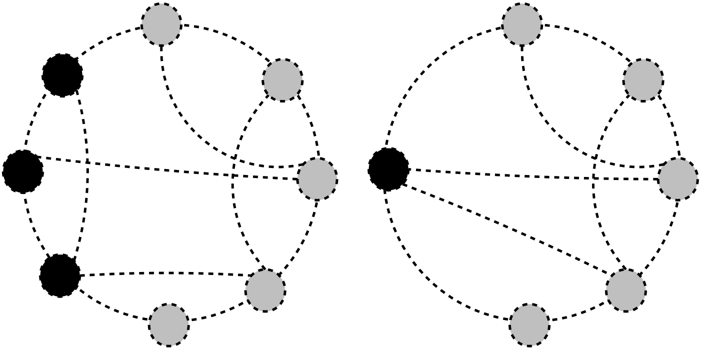
- Vertices are related to either cuts or items
- No interesting additional information compared to the previous methods
- This representation does not lead to a different branching scheme

Guillotine-cutting classes

Two patterns pertain to the same guillotine-cutting class if they can be obtained by the same sequence of builds



Cycle-contraction



(a) initial graph (b) after cycle-contraction

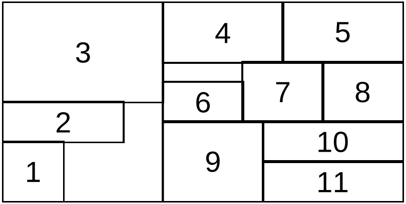
Guillotine graphs

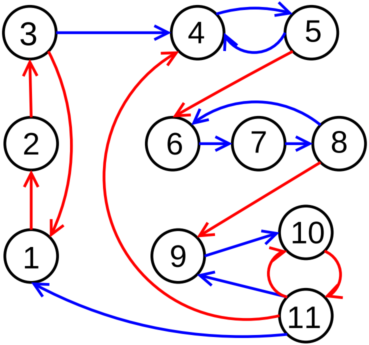
Definition

Let G be an arc-colored graph. G is a *guillotine graph* if the two following conditions hold :

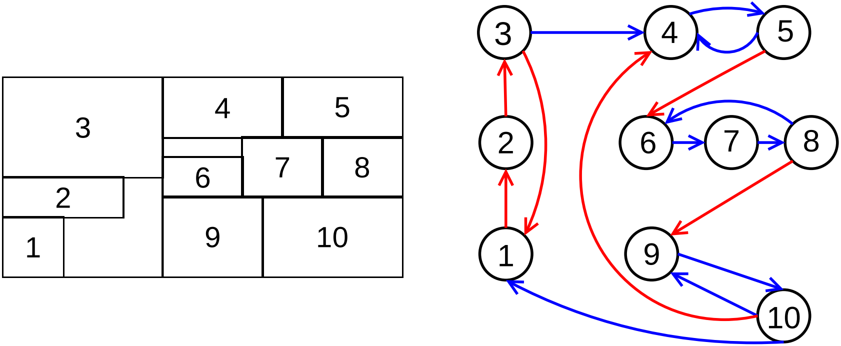
1. G can be reduced to a single vertex x by iterative contractions of monochromatic circuits
2. no steps are encountered where a vertex pertains to two monochromatic circuits

Guillotine graphs : illustration

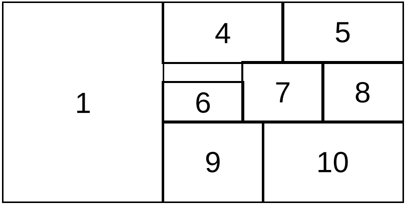


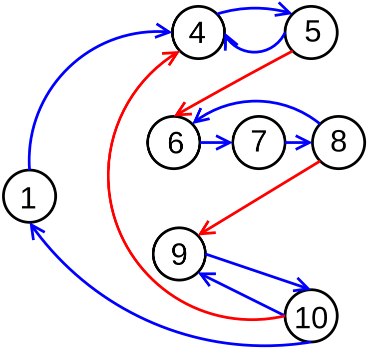


Guillotine graphs : illustration

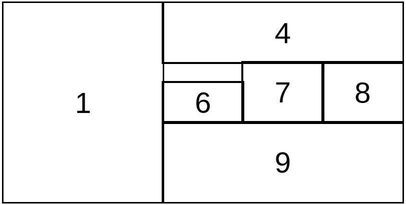


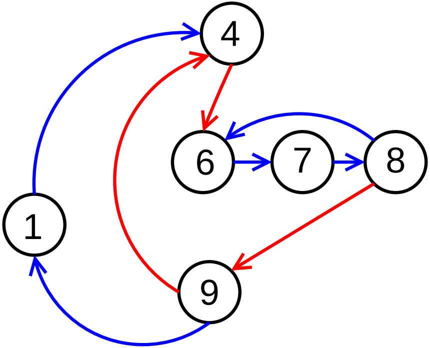
Guillotine graphs : illustration



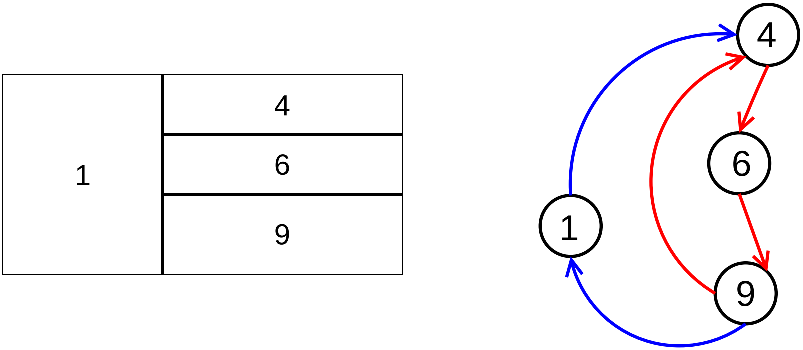


Guillotine graphs : illustration

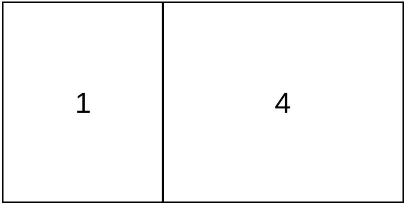


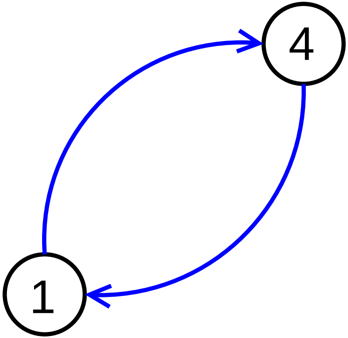


Guillotine graphs : illustration

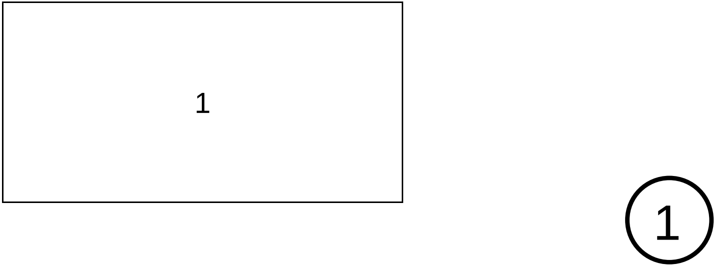


Guillotine graphs : illustration





Guillotine graphs : illustration



Dominant guillotine graphs

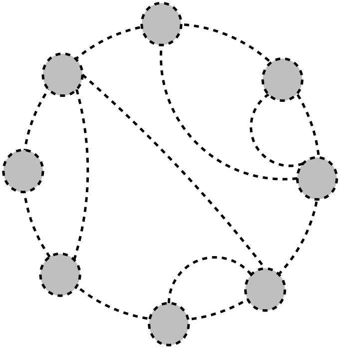
- Two configurations can be equivalent
- The order of the vertices in a circuit does not change the configuration

Definition

A guillotine graph G is dominant if in all graphs that can be obtained from G by iterative circuit contractions, vertices in all monochromatic circuits are ordered by increasing value of label.

- A guillotine-cutting class is related to an unique dominant guillotine graph.

The combinatorial structure of a dominant guillotine graph



Properties of a dominant guillotine graph G

- They contain an Hamiltonian circuit and "*backward* edges"
- $n \leq m \leq 2n - 2$
- They can be recognized in $O(n)$ time
- There is only one way of orienting and two ways of coloring an uncolored guillotine graph
- A dominant guillotine graph is associated with a unique guillotine-cutting class
- Computing the corresponding guillotine pattern takes $O(n)$ time

Computational experiments

	IMVB			IGG		
instance	nodes	cpu (s)	cpu x MIPS	nodes	cpu (s)	cpu x MIPS
SCP2	1797	3.4	857	80	0.2	2683
SCP3	3427	6.8	1714	61	0.3	3714
SCP4	6356	78.6	19807	759	0.7	9699
SCP6	8012	54.6	13759	22	0.1	1651
SCP7	1195	1.8	454	46	0.2	2683
SCP11	11036	221.5	55818	25	0.2	2889
SCP13	4359	39.5	9954	73	0.1	1651
SCP14	4782	41.7	10508	63	0.4	4746
SCP16	85627	654.8	165010	2253	2.7	35081
SCP17	13668	227.3	57280	1361	1.7	22493
SCP18	22087	321.5	81018	3419	4.2	54892
SCP19	39550	1794.3	452164	2733	2.3	30954
SCP20	36577	874.3	220324	1909	1.7	22493
SCP21	26748	1757.6	442915	8624	8.0	105450
SCP22	40909	606.0	152712	640	2.6	34875
SCP23	29512	691.9	174359	1754	1.4	18779
SCP24	117013	6265.0	1578780	12402	10.7	140738
SCP25	69434	3735.8	941422	6485	10.7	141563
Avg.	20972	695.2	175188	1721	2.0	26786

Future work

Future work

- A global "guillotine" constraint in a CP solver ?
- Lower bounds/feasibility tests based on the model ?
- Designing heuristics based on the new model

1. Introduction
2. Heuristics / feasibility tests
3. Exact methods
4. Guillotine-cutting problem
5. Summary and some issues

Summary

- a good heuristic and feasibility tests are mandatory
- constraint-programming techniques are well suited to rectangle placement problems
- packing and scheduling problems are tightly linked
- the guillotine constraint leads to totally different models

Some issues

- "real" two-dimensional feasibility tests
- a better link with mathematical programming
- taking into account the information during the search
- embedding the guillotine condition as a global constraint