Packing with biased random-key genetic algorithms

Mauricio G. C. Resende
AT&T Labs Research
Middletown, New Jersey

mgcr@research.att.com

Talk given at U. of Florida
Gainesville, Florida ✤ March 19, 2014

Summary

Part 1: Packing with BRKGA (~35 minutes)

- Combinatorial optimization
- Random-key genetic algorithm of Bean (1994)
- Biased random-key genetic algorithms (BRKGA)
- BRKGA for 2-dim and 3-dim packing
- BRKGA for 3-dim bin packing

Part 2: My vision for the Department of Industrial & Systems Engineering at the University of Florida (~20 minutes)

Combinatorial optimization

Optimization problems are commonly classified into two groups:

1) Continuous optimization: Those with continuous variables, that in principle may take any real value.

2) Combinatorial optimization: Those represented by discrete variables, that may take only a finite or countably infinite set of values.

Combinatorial optimization

Combinatorial optimization problems reduce to the search for a best solution (or object) in a finite (or countably infinite) set.

The solution set may typically be formed by binary or integer variables, permutations, paths, trees, cycles, or graphs.

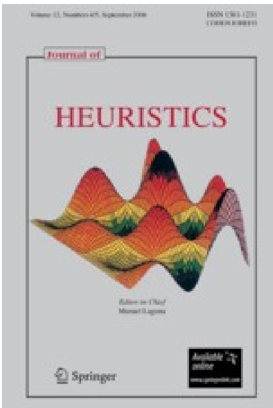
Applications abound: e.g. scheduling, routing, design, ...

Genetic algorithms (GA)

Genetic algorithms are stochastic search methods for combinatorial optimization that apply Darwin's principle of survival of the fittest to evolve a population of solutions towards the optimal solution.

In this talk we describe a special class of genetic algorithm, the Biased Random-Key GA (BRKGA) and apply it to solve hard 2D and 3D packing problems.

Reference



J.F. Gonçalves and M.G.C.R., "Biased random-key genetic algorithms for combinatorial optimization," J. of Heuristics, vol.17, pp. 487-525, 2011.

Tech report version:

http://www.research.att.com/~mgcr/doc/srkga.pdf

Random-key genetic algorithms

Random key – random number in [0,1)

0.2384

Random-key genetic algorithms

Random key – random number in [0,1)

0.8394

Random-key genetic algorithms

Random key – random number in [0,1)

0.9234

Random-key genetic algorithms

Random key – random number in [0,1)

0.0143

Random-key genetic algorithms

Solutions can be encoded by a vector of random keys

0.4756	0.9903	0.0012	0.7675

Random-key genetic algorithms

Solutions can be encoded by a vector of random keys

0.1102	0.5887	0.3929	0.3992

Random-key genetic algorithms

Solutions can be encoded by a vector of random keys

0.2384	0.8394	0.9234	0.0143

Random-key genetic algorithms

Solutions can be encoded by a vector of random keys

0.2324	0.1029	0.3428	0.3233

Random-key genetic algorithms

Decoder takes vector of random keys as input and outputs solution to optimization problem

0.1102	0.5887	0.3929	0.3992

Random-key genetic algorithms

Decoder takes vector of random keys as input and outputs solution to optimization problem

0.1102	0.5887	0.3929	0.3992

Sorting, for example, … gives us a permutation

0.1102	0.3929	0.3992	0.5887
1	3	4	2

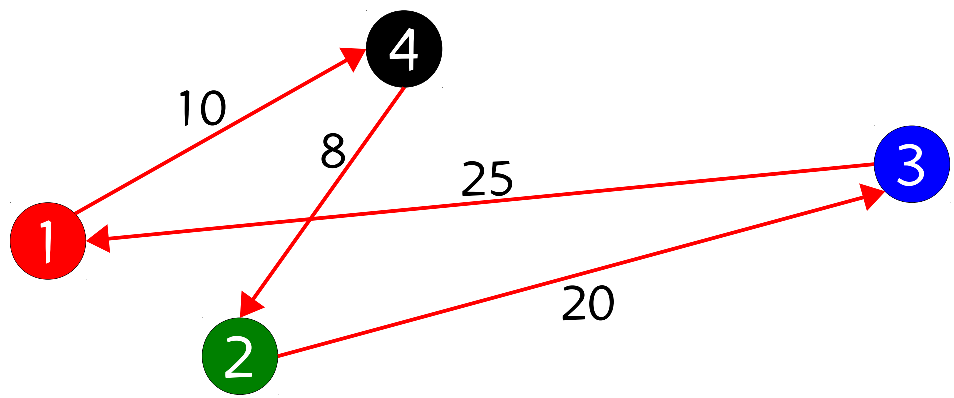
Random-key genetic algorithms

Bean (1994) – proposed a GA based on random keys for problems whose solutions can be encoded as permutations (e.g. sequencing, assignment, TSP)

Random-key genetic algorithms

0.0012	0.4756	0.7675	0.9903
3	1	4	2

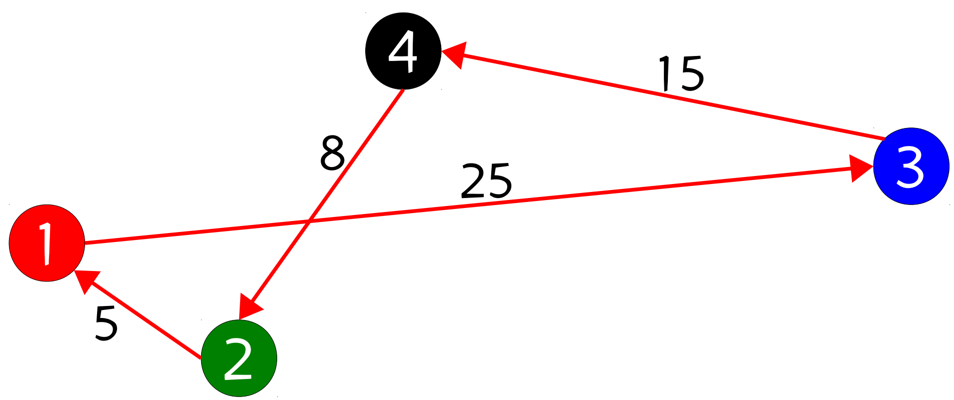
TSP tour length: 63



Random-key genetic algorithms

0.1102	0.3929	0.3992	0.5887
1	3	4	2

TSP tour length: 53

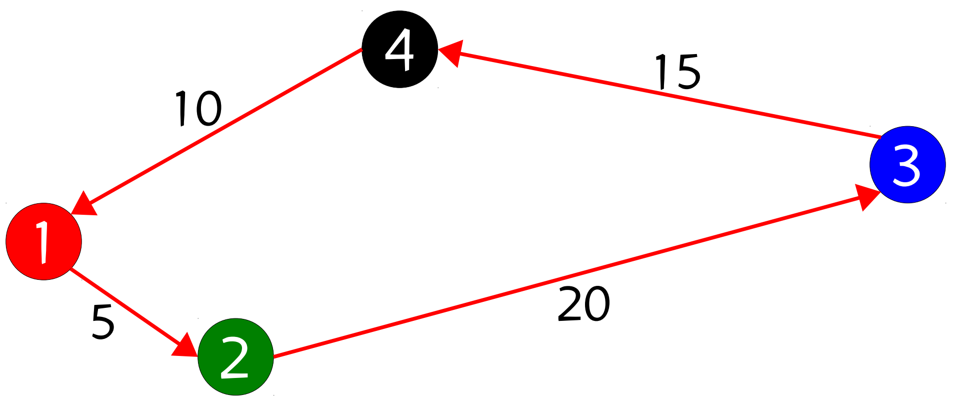


Random-key genetic algorithms

0.0143	0.2384	0.8394	0.9234
4	1	2	3

TSP tour length: 50

Optimal tour!

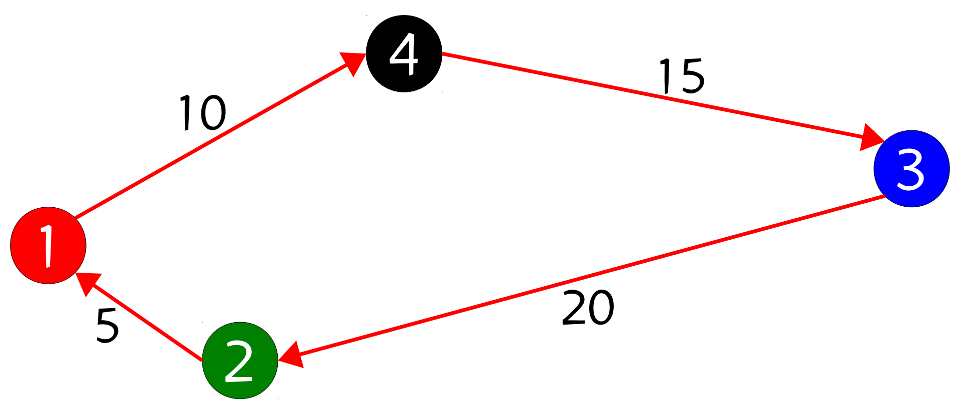


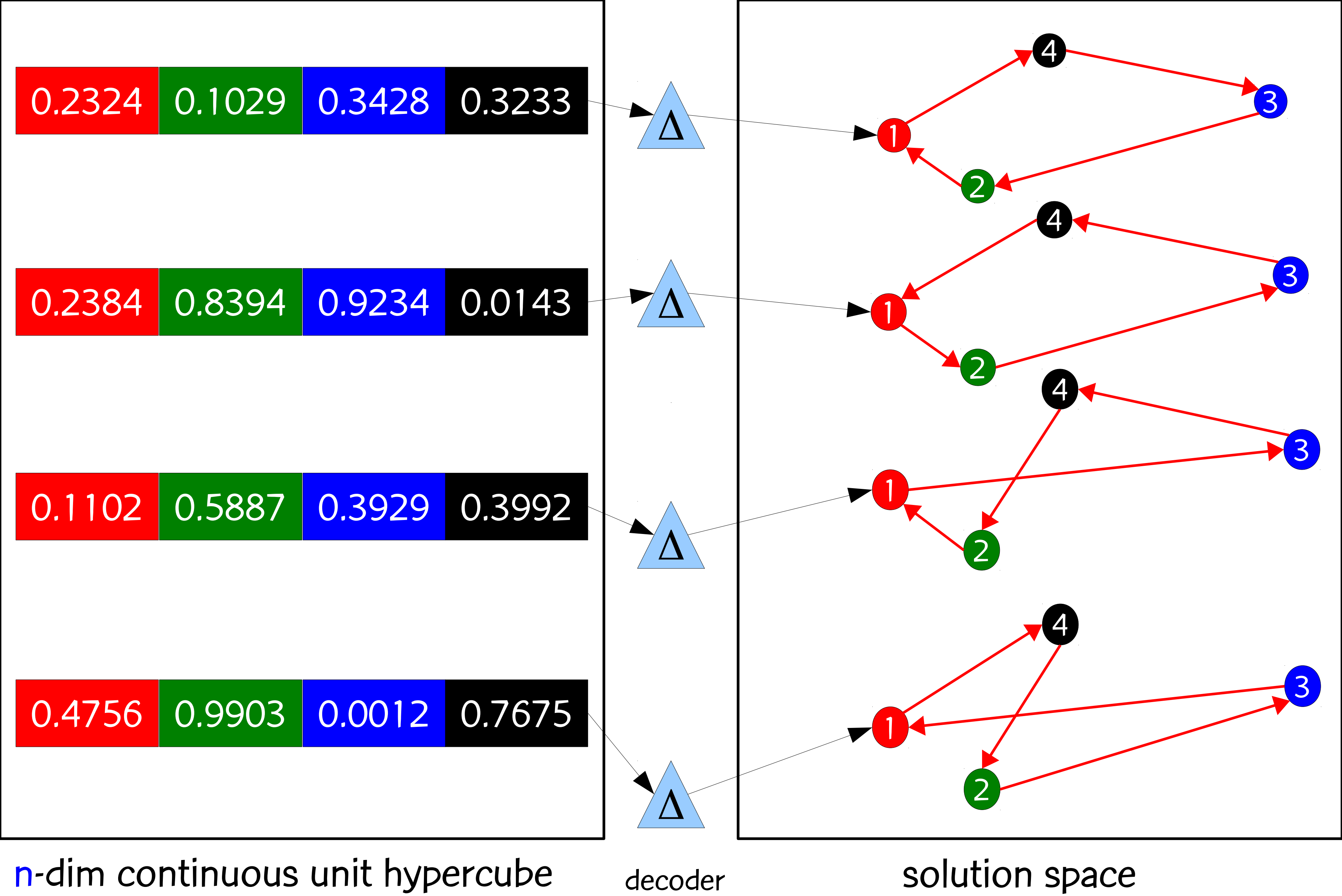
Random-key genetic algorithms

0.1029	0.2324	0.3233	0.3428
2	1	4	3

TSP tour length: 50

Another optimal tour!




0.2324
0.1029
0.3428
0.3233
0.2384
0.8394
0.9234
0.0143
0.1102
0.5887
0.3929
0.3992
0.4756
0.9903
0.0012
0.7675
Δ
1
2
3
4
n-dim continuous unit hypercube
decoder
solution space

Feasibility-preserving property

Given a valid decoder and two random-key vectors, each corresponding to a feasible solution ...

tour length: 50

0.2324	0.1029	0.3428	0.3233
0.4756	0.9903	0.0012	0.7675

tour length: 63

Any combination (flipping coin) of the two

Feasibility-preserving property

Given a valid decoder and two random-key vectors, each corresponding to a feasible solution ...

tour length: 50

0.2324 √	0.1029	0.3428	0.3233
0.4756	0.9903	0.0012	0.7675

tour length: 63

Any combination (flipping coin) of the two

0.2324

Feasibility-preserving property

Given a valid decoder and two random-key vectors, each corresponding to a feasible solution ...

tour length: 50

0.2324 √	0.1029	0.3428	0.3233
0.4756	0.9903 √	0.0012	0.7675

tour length: 63

Any combination (flipping coin) of the two

0.2324	0.9903

Feasibility-preserving property

Given a valid decoder and two random-key vectors, each corresponding to a feasible solution ...

tour length: 50

0.2324 √	0.1029	0.3428 √	0.3233

0.4756	0.9903 √	0.0012	0.7675

tour length: 63

Any combination (flipping coin) of the two

0.2324	0.9903	0.3428

Feasibility-preserving property

Given a valid decoder and two random-key vectors, each corresponding to a feasible solution ...

tour length: 50

0.2324 √	0.1029	0.3428 √	0.3233
0.4756	0.9903 √	0.0012	0.7675 √

tour length: 63

Any combination (flipping coin) of the two

0.2324	0.9903	0.3428	0.7675

tour length: 53

also corresponds to a feasible solution.

Parameterized uniform crossover

[Spears & DeJong, 1991]

Flip biased coin (probability p of resulting in heads) n times for random-key vector of size n

tour length: 50

0.2324	0.1029	0.3428	0.3233
0.4756	0.9903	0.0012	0.7675

tour length: 63

$p = 0.7$

Parameterized uniform crossover

[Spears & DeJong, 1991]

Flip biased coin (probability p of resulting in heads) n times for random-key vector of size n

tour length: 50

0.2324 √	0.1029	0.3428	0.3233
0.4756	0.9903	0.0012	0.7675

tour length: 63

$p = 0.7$

0.6872 (H)
0.2324

Parameterized uniform crossover

[Spears & DeJong, 1991]

Flip biased coin (probability p of resulting in heads) n times for random-key vector of size n

tour length: 50

0.2324 √	0.1029	0.3428	0.3233
0.4756	0.9903 √	0.0012	0.7675

tour length: 63

$p = 0.7$

0.6872 (H)	0.7802 (T)
0.2324	0.9903

Parameterized uniform crossover

[Spears & DeJong, 1991]

Flip biased coin (probability p of resulting in heads) n times for random-key vector of size n

tour length: 50

0.2324 √	0.1029	0.3428 √	0.3233
0.4756	0.9903 √	0.0012	0.7675

tour length: 63

$p = 0.7$

0.6872 (H)	0.7802 (T)	0.1234 (H)
0.2324	0.9903	0.3428

Parameterized uniform crossover

[Spears & DeJong, 1991]

Flip biased coin (probability p of resulting in heads) n times for random-key vector of size n

tour length: 50

0.2324 √	0.1029	0.3428 √	0.3233
0.4756	0.9903 √	0.0012	0.7675 √

tour length: 63

$p = 0.7$

0.6872 (H)	0.7802 (T)	0.1234 (H)	0.9278 (T)
0.2324	0.9903	0.3428	0.7675

tour length: 53

Bean's algorithm

- Start with population of P vectors of n random keys

- Evolve population until stopping criterion is satisfied

- Best decoded solution from vectors in final population is output as solution of the algorithm

Evolution in Bean's algorithm

- Decode and evaluate all new vectors of random keys
- Partition population into a small set of P_E elite solutions and $P - P_E$ non-elite solutions
- Copy all P_E elite vectors to new population
- Generate P_M vectors of random keys (mutants) in new population
- Apply parameterized uniform crossover on $P - P_E - P_M$ pairs of vectors of random keys chosen at random from entire population and add each resulting vector to new population

Biased random key genetic algorithm

- A biased random key genetic algorithm (BRKGA) is a random key genetic algorithm (RKGA).

Biased random key genetic algorithm

- A biased random key genetic algorithm (BRKGA) is a random key genetic algorithm (RKGA).
- BRKGA and RKGA differ in how mates are chosen for crossover and how parametrized uniform crossover is applied.

How RKGA & BRKGA differ

RKGA	BRKGA
both parents chosen at random from entire population	

How RKGA & BRKGA differ

RKGA	**BRKGA**
both parents chosen at random from entire population	both parents chosen at random but one parent chosen from population of elite solutions

How RKGA & BRKGA differ

RKGA	BRKGA
both parents chosen at random from entire population	both parents chosen at random but one parent chosen from population of elite solutions
either parent can be associated with heads in parametrized uniform crossover coin flip	

How RKGA & BRKGA differ

RKGA	BRKGA
both parents chosen at random from entire population	both parents chosen at random but one parent chosen from population of elite solutions
either parent can be associated with heads in parametrized uniform crossover coin flip	best fit parent is associated with heads in parametrized uniform crossover coin flip

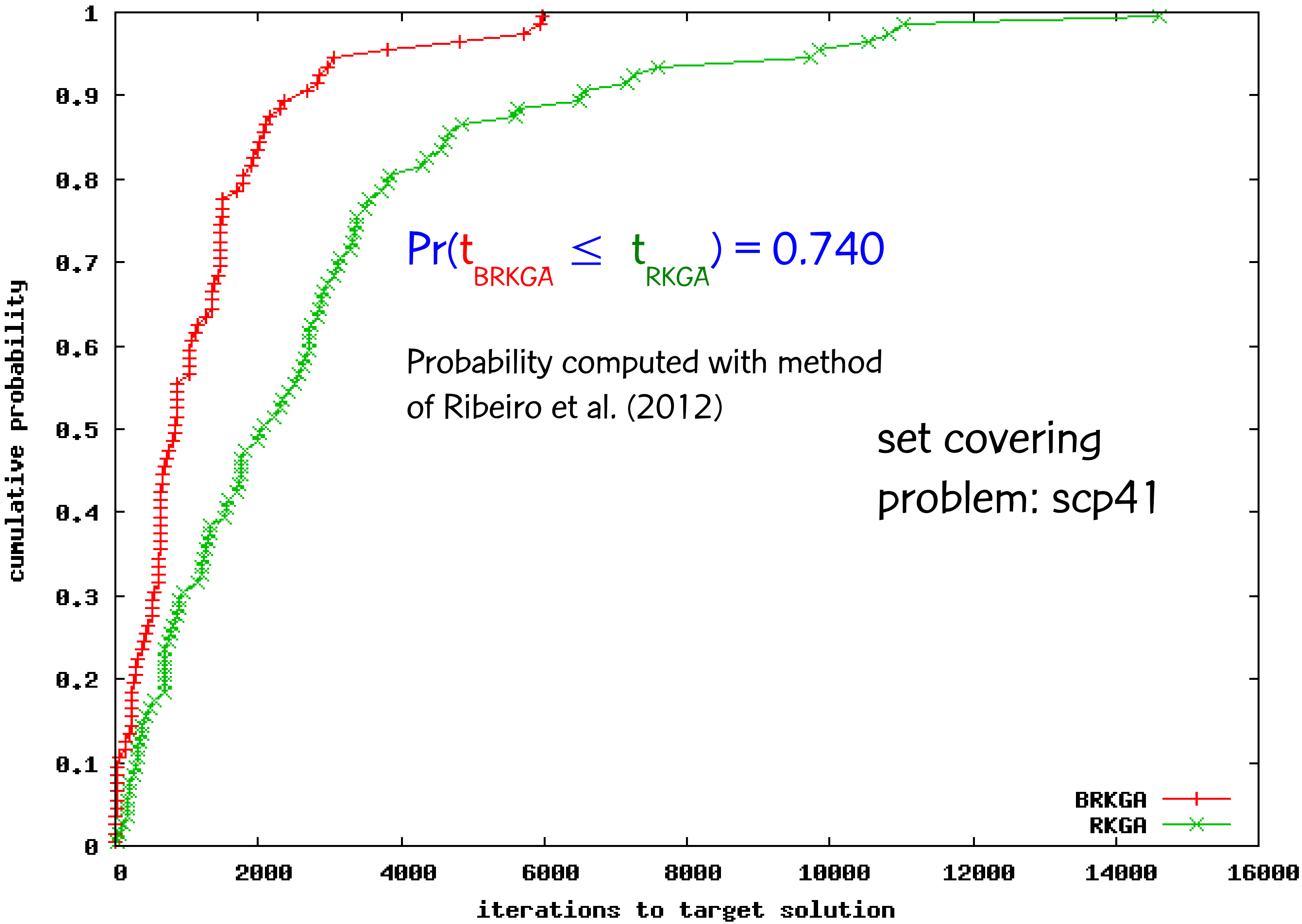
How RKGA & BRKGA differ

RKGA	BRKGA
both parents chosen at random from entire population	both parents chosen at random but one parent chosen from population of elite solutions
either parent can be associated with heads in parametrized uniform crossover coin flip	best fit parent is associated with heads in parametrized uniform crossover coin flip

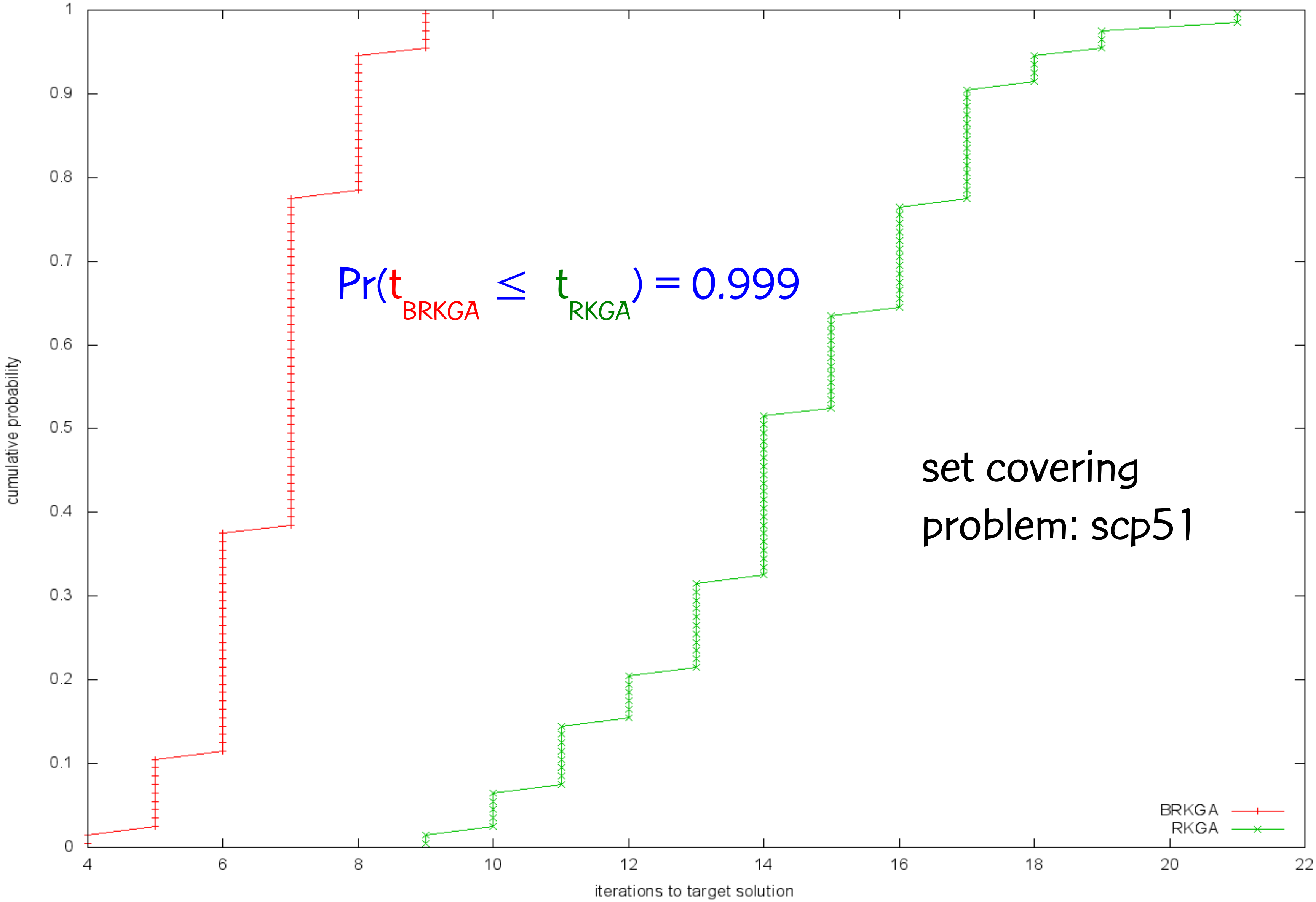
This adds "survival of the fittest" to RKGA

Paper comparing BRKGA and Bean's Method

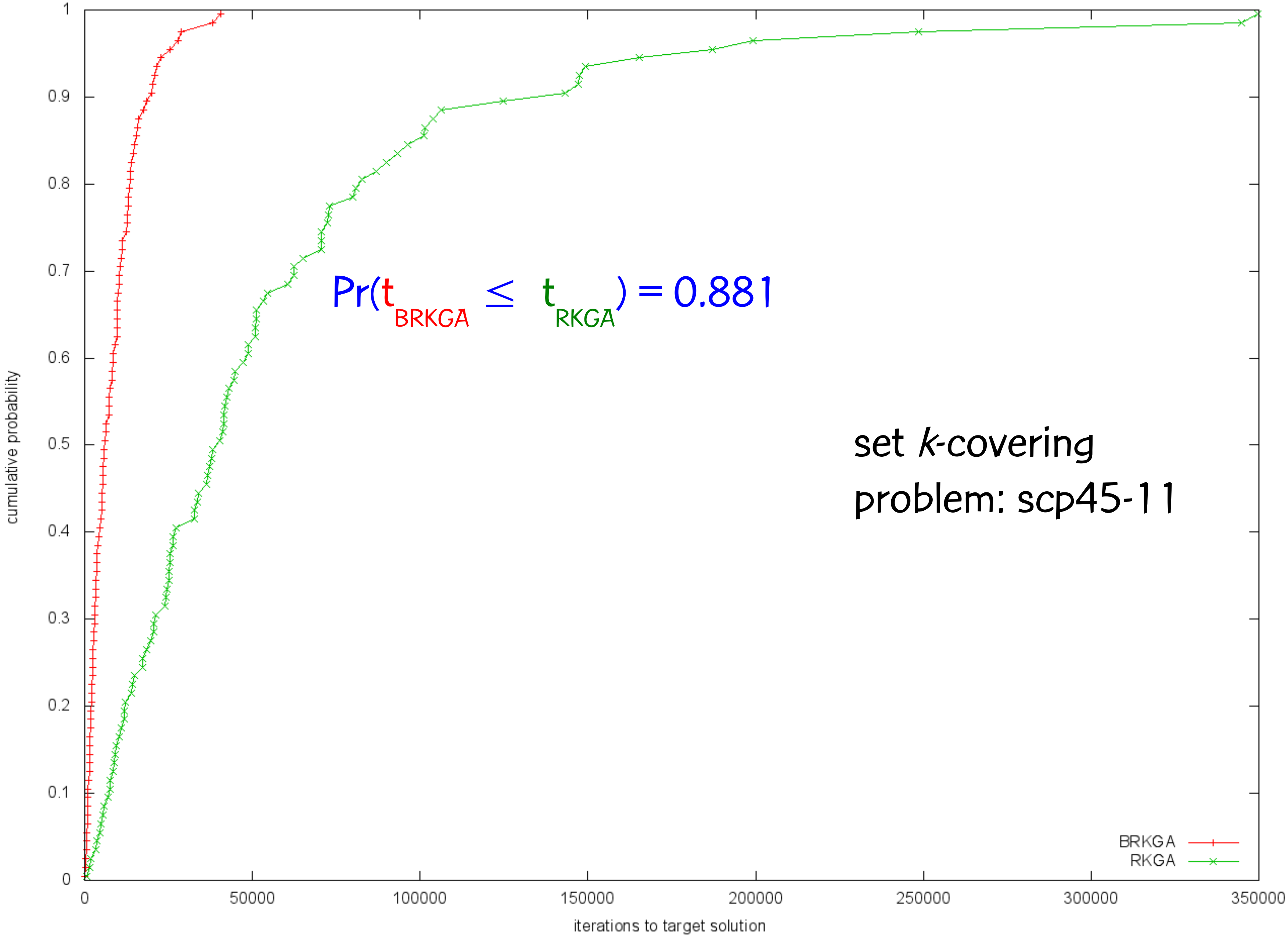
Gonçalves, M.G.C.R., and Toso, "Biased and unbiased random-key genetic algorithms: An experimental analysis", Proceedings of the 10th Metaheuristics International Conference, Singapore, August 2013.



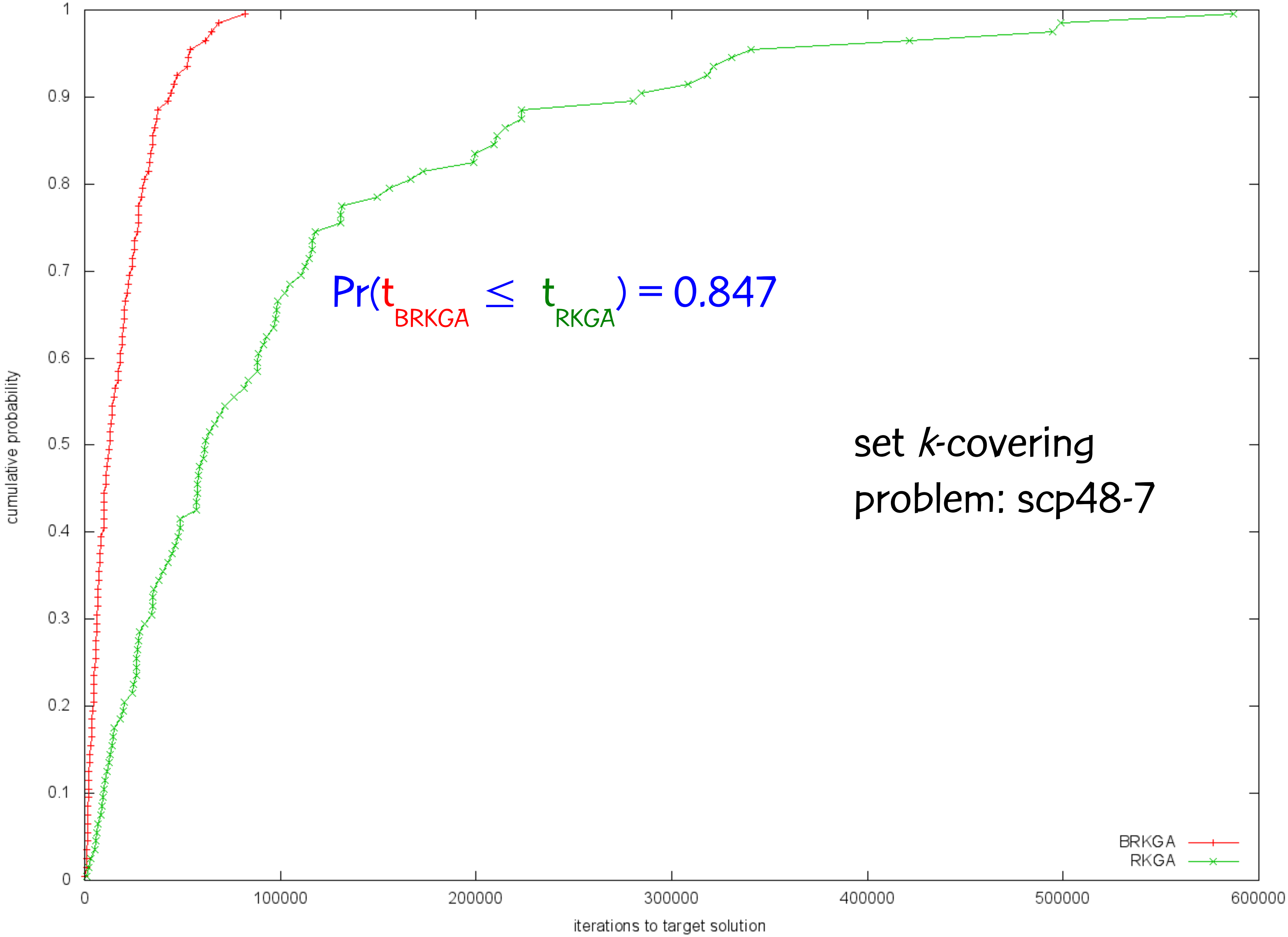
Pr(t_BRKGA ≤ t_RKGA) = 0.740
Probability computed with method
of Ribeiro et al. (2012)
set covering
problem: scp41
cumulative probability
iterations to target solution
BRKGA
RKGA
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
2000
4000
6000
8000
10000
12000
14000
16000



Pr(t_BRKGA ≤ t_RKGA) = 0.999
set covering
problem: scp51
cumulative probability
iterations to target solution
BRKGA
RKGA
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
4
6
8
10
12
14
16
18
20
22

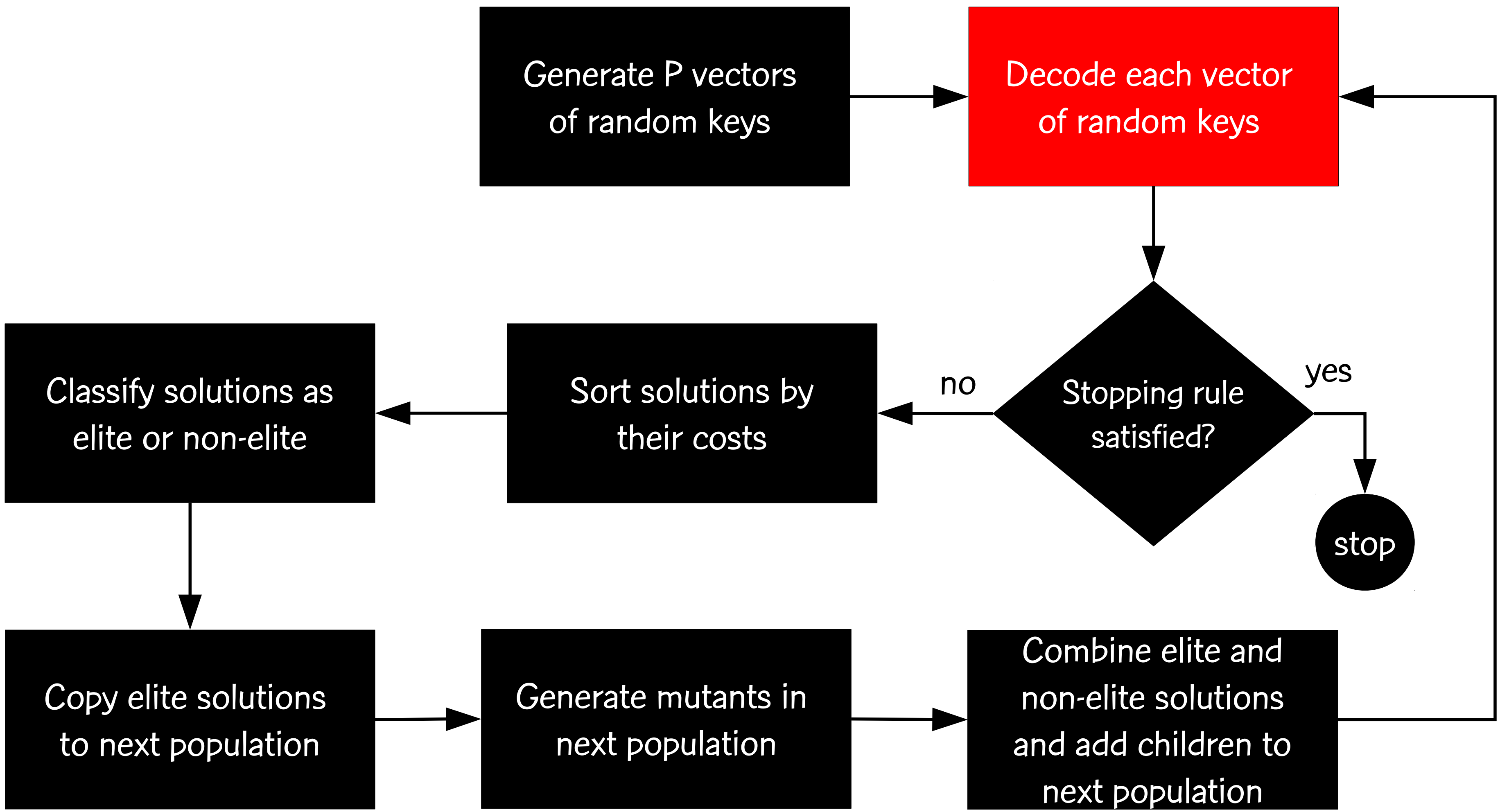


$\Pr(t_{BRKGA} \leq t_{RKGA}) = 0.881$
set *k*-covering
problem: scp45-11
cumulative probability
iterations to target solution
BRKGA
RKGA
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
50000
100000
150000
200000
250000
300000
350000

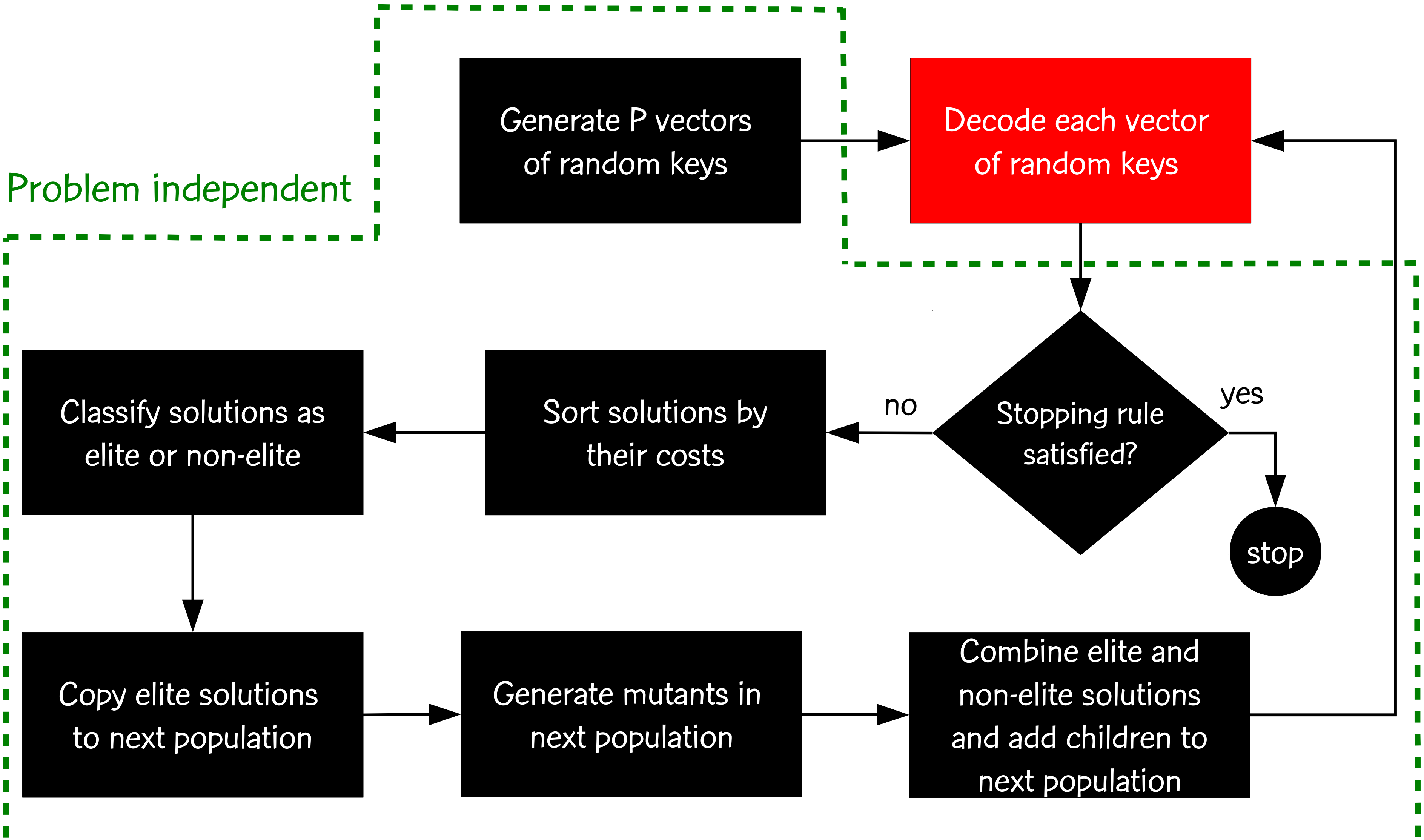


$\Pr(t_{BRKGA} \leq t_{RKGA}) = 0.847$
set *k*-covering
problem: scp48-7
cumulative probability
iterations to target solution
BRKGA
RKGA
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
100000
200000
300000
400000
500000
600000

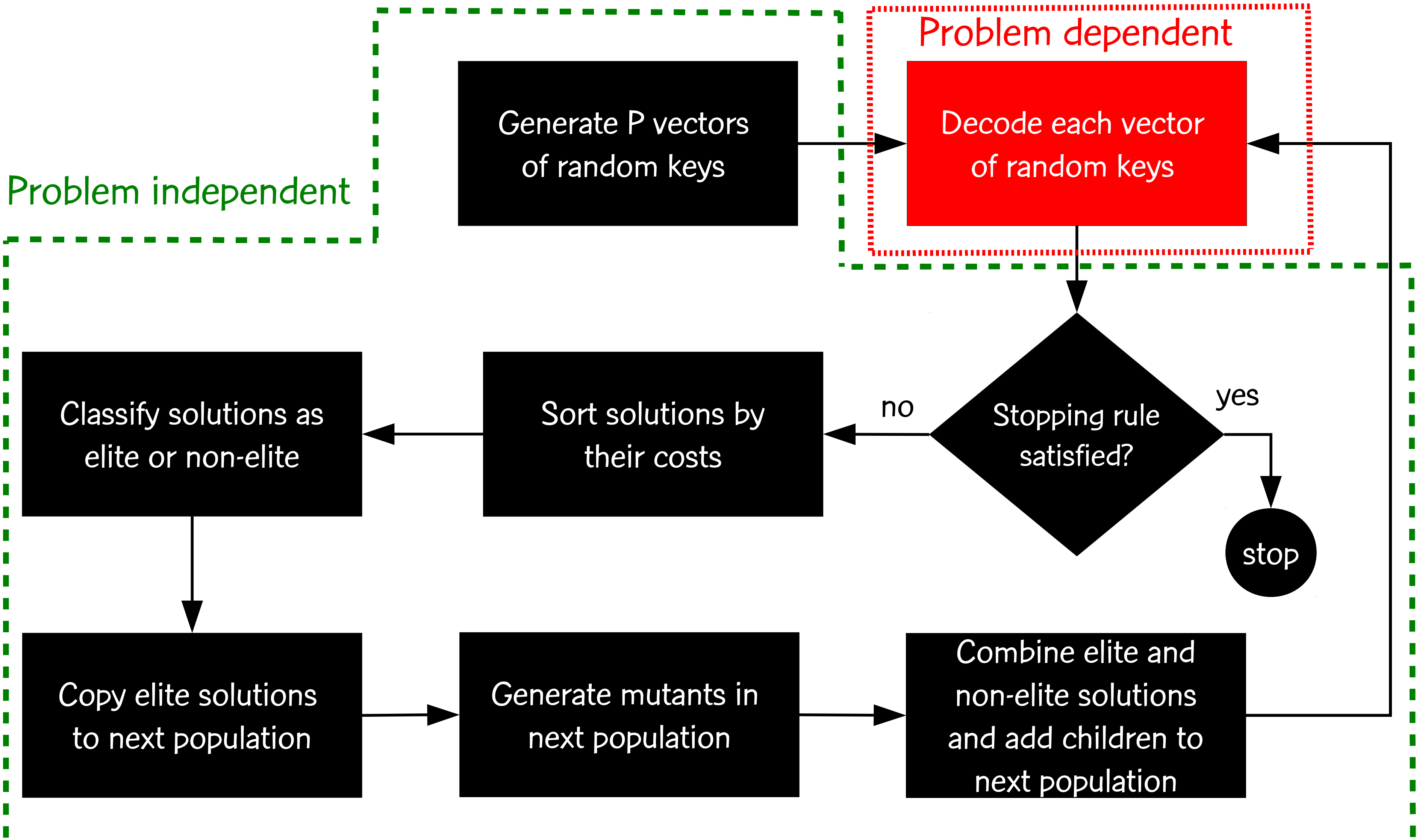
Framework for biased random-key genetic algorithms



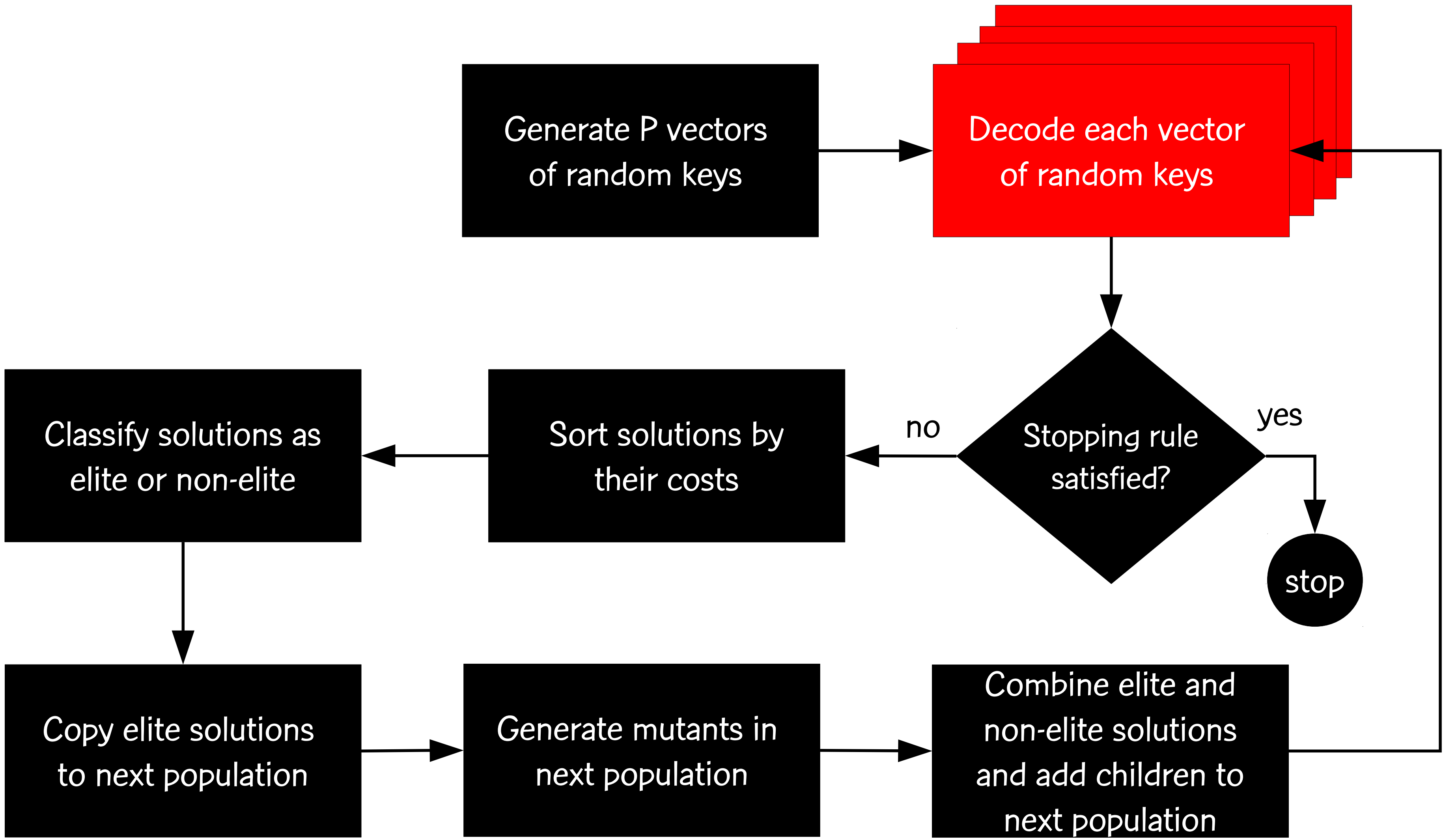
Framework for biased random-key genetic algorithms



Framework for biased random-key genetic algorithms



Decoding of random key vectors can be done in parallel



Specifying a BRKGA

Specifying a biased random-key GA

- Encoding is always done the same way, i.e. with a vector of N random-keys (parameter N must be specified)

Specifying a biased random-key GA

- Encoding is always done the same way, i.e. with a vector of N random-keys (parameter N must be specified)
- Decoder that takes as input a vector of N random-keys and outputs the corresponding solution of the combinatorial optimization problem and its cost (this is usually a heuristic)

Specifying a biased random-key GA

- Encoding is always done the same way, i.e. with a vector of N random-keys (parameter N must be specified)
- Decoder that takes as input a vector of N random-keys and outputs the corresponding solution of the combinatorial optimization problem and its cost (this is usually a heuristic)
- Parameters

Specifying a biased random-key GA

Parameters:

- Size of population
- Size of elite partition
- Size of mutant set
- Child inheritance probability
- Stopping criterion

Specifying a biased random-key GA

Parameters:

- Size of population: a function of N, say N or 2N
- Size of elite partition
- Size of mutant set
- Child inheritance probability
- Stopping criterion

Specifying a biased random-key GA

Parameters:

- Size of population: a function of N, say N or 2N
- Size of elite partition: 15-25% of population
- Size of mutant set
- Child inheritance probability
- Stopping criterion

Specifying a biased random-key GA

Parameters:

- Size of population: a function of N, say N or 2N
- Size of elite partition: 15-25% of population
- Size of mutant set: 5-15% of population
- Child inheritance probability
- Stopping criterion

Specifying a biased random-key GA

Parameters:

- Size of population: a function of N, say N or 2N
- Size of elite partition: 15-25% of population
- Size of mutant set: 5-15% of population
- Child inheritance probability: > 0.5, say 0.7
- Stopping criterion

Specifying a biased random-key GA

Parameters:

- Size of population: a function of N, say N or 2N
- Size of elite partition: 15-25% of population
- Size of mutant set: 5-15% of population
- Child inheritance probability: > 0.5, say 0.7
- Stopping criterion: e.g. time, # generations, solution quality, # generations without improvement

brkgaAPI: A C++ API for BRKGA

- brkgaAPI is an efficient and easy-to-use object oriented application programming interface (API) for the algorithmic framework of BRKGA.

brkgaAPI: A C++ API for BRKGA

- brkgaAPI is an efficient and easy-to-use object oriented application programming interface (API) for the algorithmic framework of BRKGA.
- Cross-platform library handles large portion of problem independent modules that make up the framework, e.g.
  - population management
  - evolutionary dynamics

brkgaAPI: A C++ API for BRKGA

- brkgaAPI is an efficient and easy-to-use object oriented application programming interface (API) for the algorithmic framework of BRKGA.
- Cross-platform library handles large portion of problem independent modules that make up the framework, e.g.
  - population management
  - evolutionary dynamics
- Implemented in C++ and may benefit from shared-memory parallelism if available.

brkgaAPI: A C++ API for BRKGA

- brkgaAPI is an efficient and easy-to-use object oriented application programming interface (API) for the algorithmic framework of BRKGA.
- Cross-platform library handles large portion of problem independent modules that make up the framework, e.g.
  - population management
  - evolutionary dynamics
- Implemented in C++ and may benefit from shared-memory parallelism if available.
- User only needs to implement problem-dependent decoder.

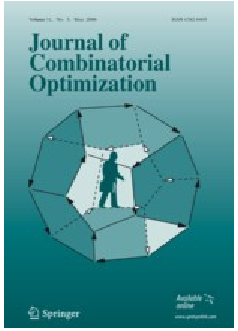
brkgaAPI: A C++ API for BRKGA

Paper: Rodrigo F. Toso and M.G.C.R., "A C++ Application Programming Interface for Biased Random-Key Genetic Algorithms," Optimization Methods & Software, published online 13 Mar 2014.

Software: http://github.com/rfrancotoso/brkgaAPI

Packing weighted rectangles with a BRKGA

Reference



J.F. Gonçalves and M.G.C.R., "A parallel multi-population genetic algorithm for a constrained two-dimensional orthogonal packing problem," Journal of Combinatorial Optimization, vol. 22, pp. 180-201, 2011.

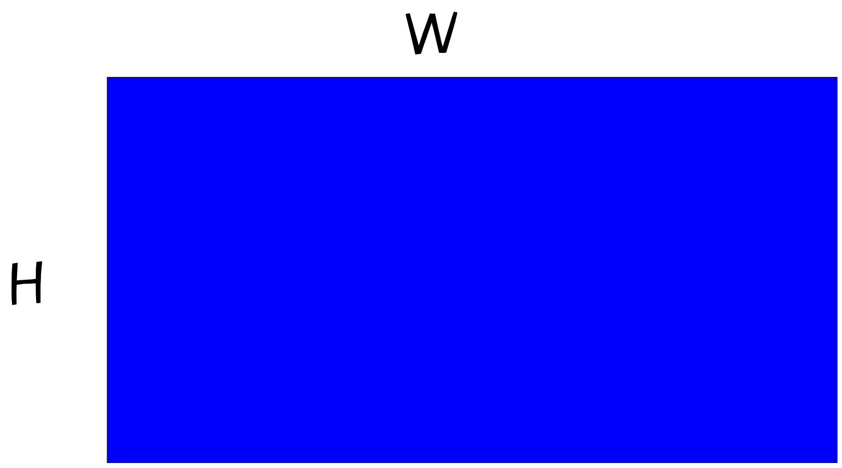
Tech report:
http://www.research.att.com/~mgcr/doc/pack2d.pdf

Constrained orthogonal packing

- Given a large planar stock rectangle (W, H) of width W and height H;

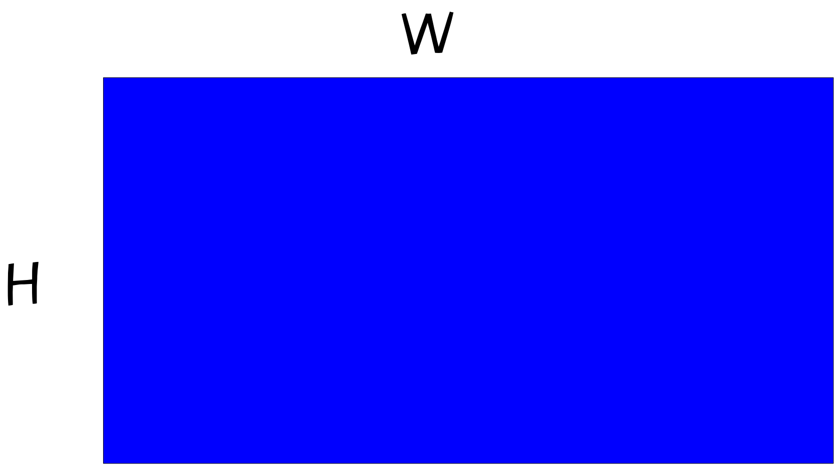
Constrained orthogonal packing

- Given a large planar stock rectangle (W, H) of width W and height H;



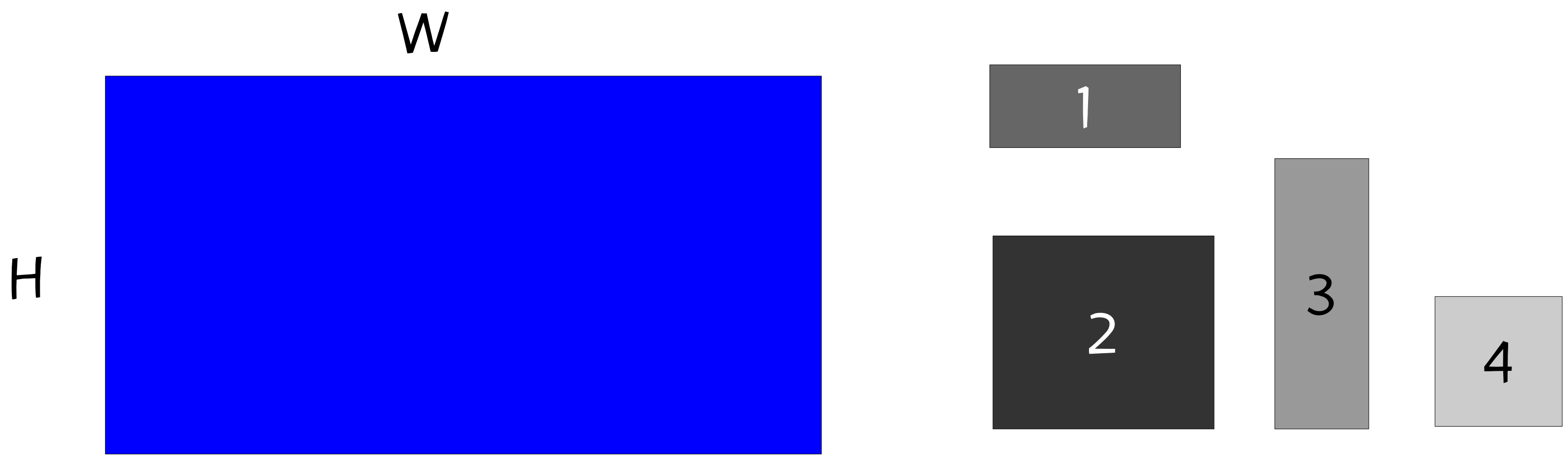
Constrained orthogonal packing

- Given a large planar stock rectangle (W, H) of width W and height H;
- Given N smaller rectangle types (w[i], h[i]), i = 1,...,N, each of width w[i], height h[i], and value v[i];



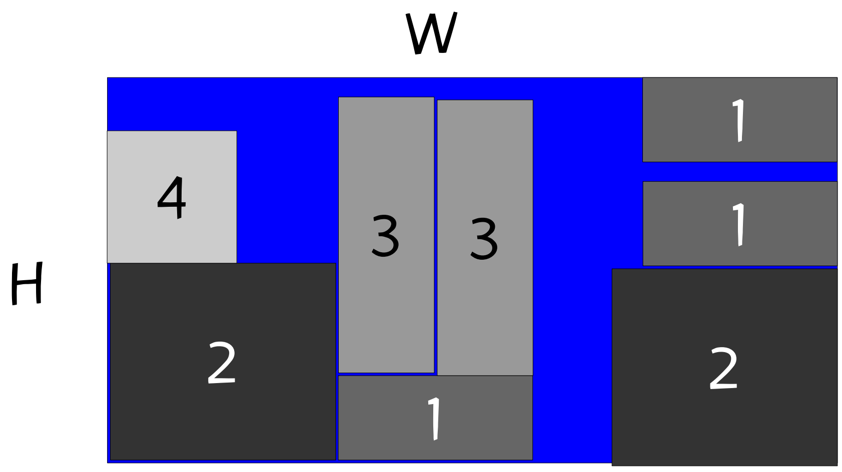
Constrained orthogonal packing

- Given a large planar stock rectangle (W, H) of width W and height H;
- Given N smaller rectangle types (w[i], h[i]), i = 1,...,N, each of width w[i], height h[i], and value v[i];



Constrained orthogonal packing

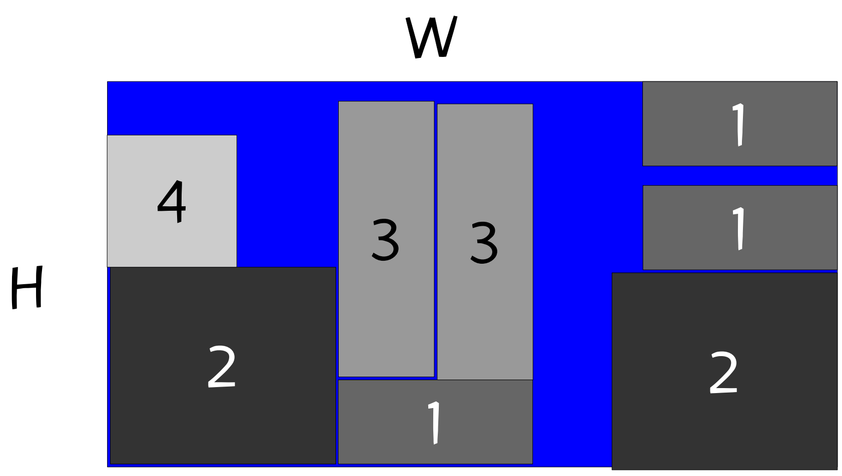
- r[i] rectangles of type i = 1, ..., N are to be packed in the large rectangle without overlap and such that their edges are parallel to the edges of the large rectangle;



Constrained orthogonal packing

- r[i] rectangles of type i = 1, ..., N are to be packed in the large rectangle without overlap and such that their edges are parallel to the edges of the large rectangle;
- For i = 1, ..., N, we require that:

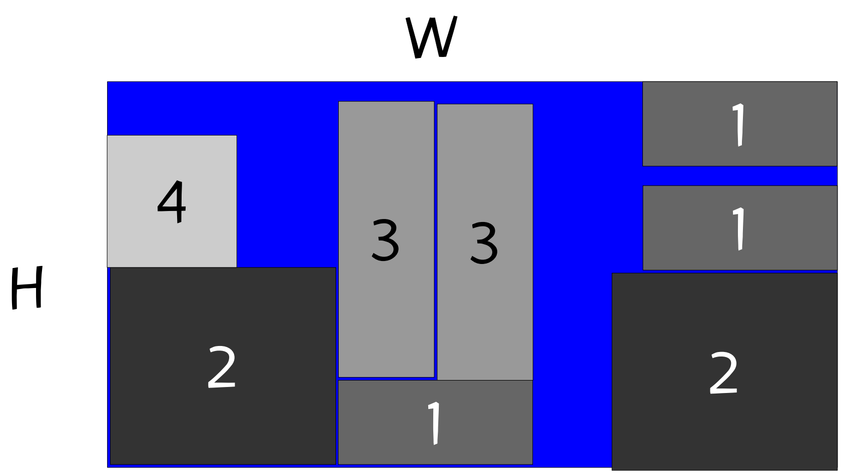
$$0 \leq P[i] \leq r[i] \leq Q[i]$$



Constrained orthogonal packing

- $r[i]$ rectangles of type $i = 1, \ldots, N$ are to be packed in the large rectangle without overlap and such that their edges are parallel to the edges of the large rectangle;
- For $i = 1, \ldots, N$, we require that:

$$0 \leq P[i] \leq r[i] \leq Q[i]$$

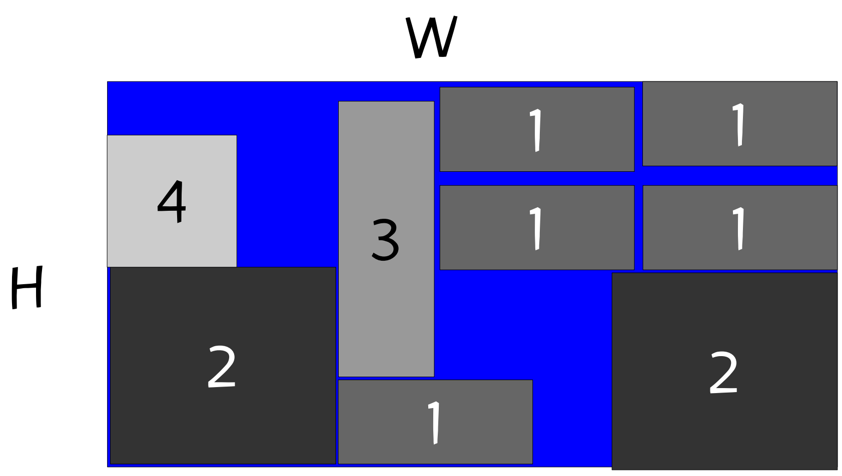


Suppose $5 \leq r[1] \leq 12$

Constrained orthogonal packing

- r[i] rectangles of type i = 1, ..., N are to be packed in the large rectangle without overlap and such that their edges are parallel to the edges of the large rectangle;
- For i = 1, ..., N, we require that:

$$0 \leq P[i] \leq r[i] \leq Q[i]$$

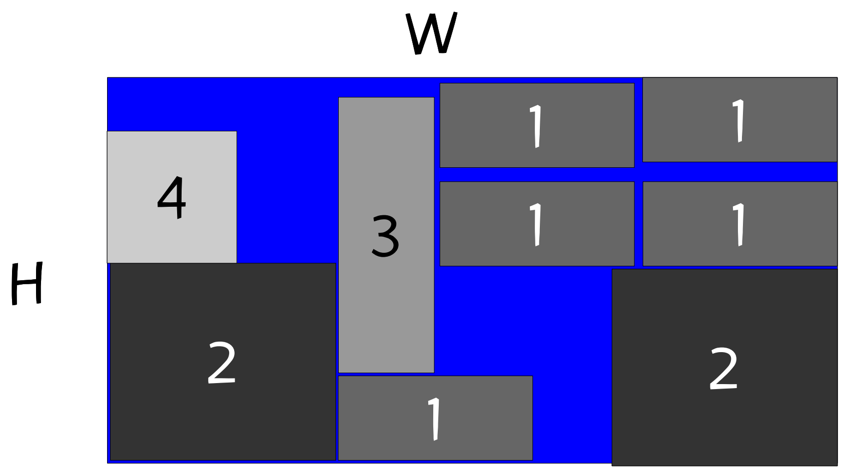


Suppose $5 \leq r[1] \leq 12$

Objective

Among the many feasible packings, we want to find one that maximizes total value of packed rectangles:

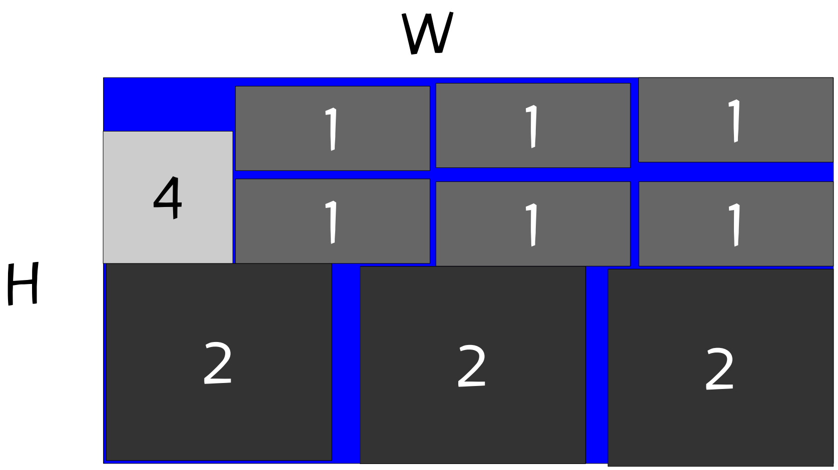
$$v[1]\ r[1] + v[2]\ r[2] + \cdots + v[N]\ r[N]$$



Objective

Among the many feasible packings, we want to find one that maximizes total value of packed rectangles:

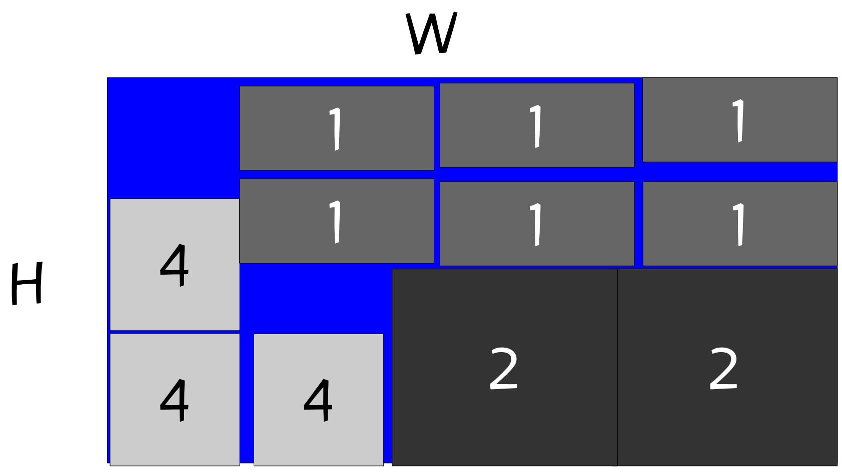
$$v[1]\, r[1] + v[2]\, r[2] + \cdots + v[N]\, r[N]$$



Objective

Among the many feasible packings, we want to find one that maximizes total value of packed rectangles:

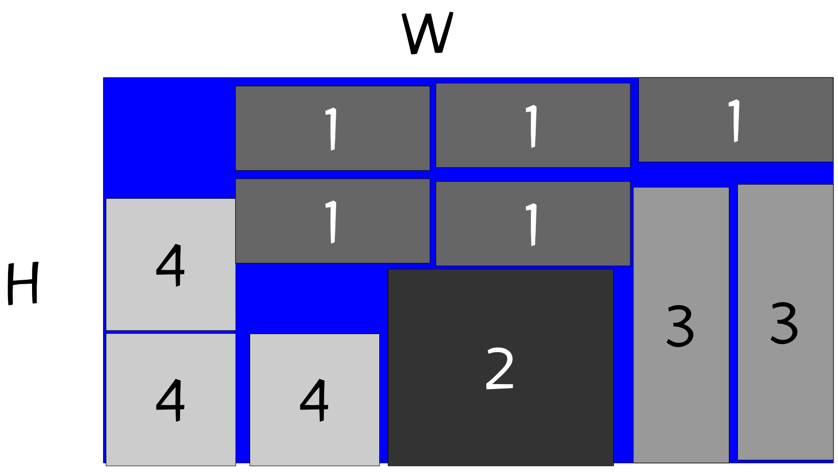
$$v[1]\ r[1] + v[2]\ r[2] + \cdots + v[N]\ r[N]$$



Objective

Among the many feasible packings, we want to find one that maximizes total value of packed rectangles:

$$v[1]\ r[1] + v[2]\ r[2] + \cdots + v[N]\ r[N]$$

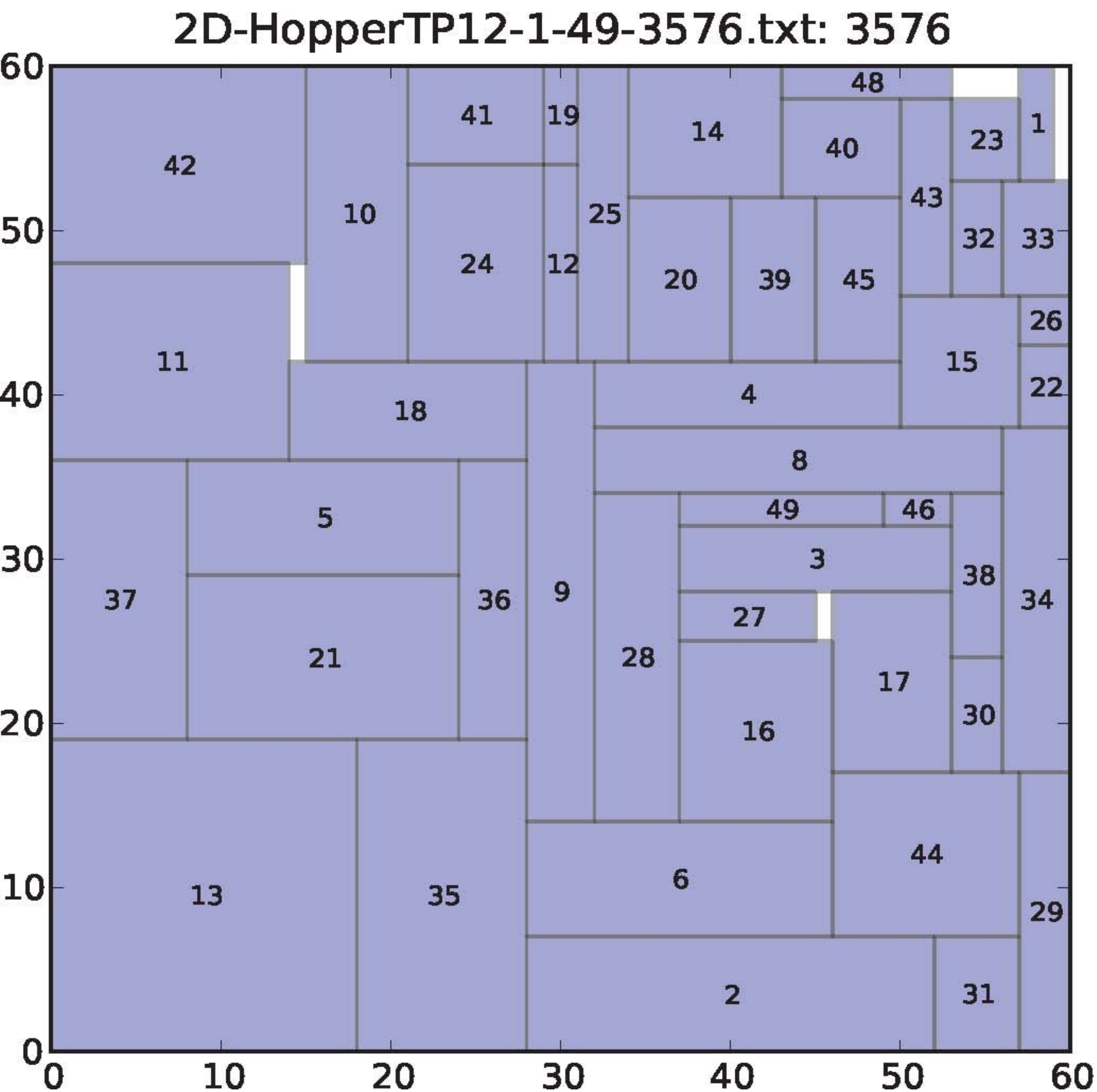


Applications

Problem arises in several production processes, e.g.

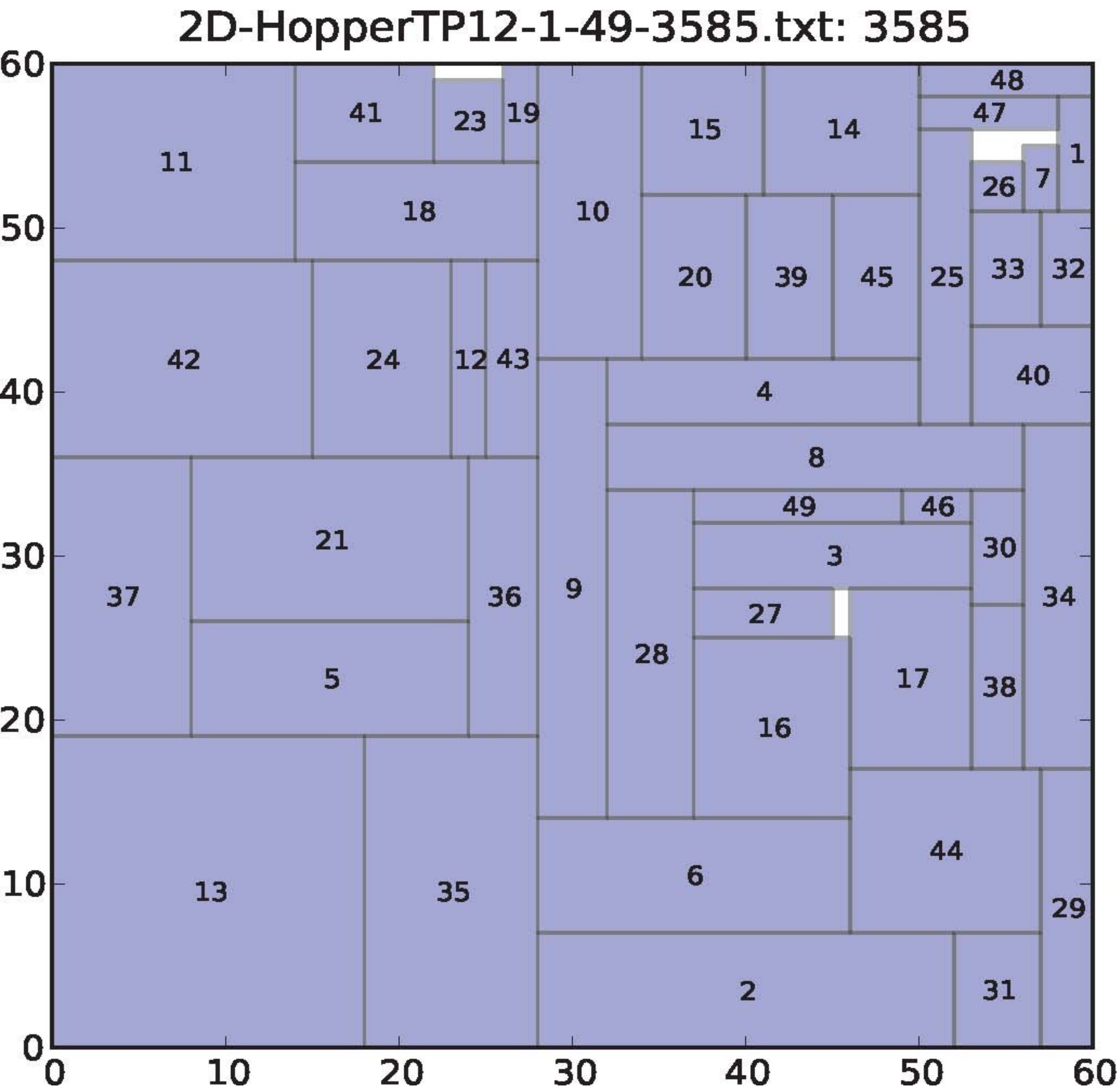
- Textile
- Glass
- Wood
- Paper

where rectangular figures are cut from large rectangular sheets of materials.



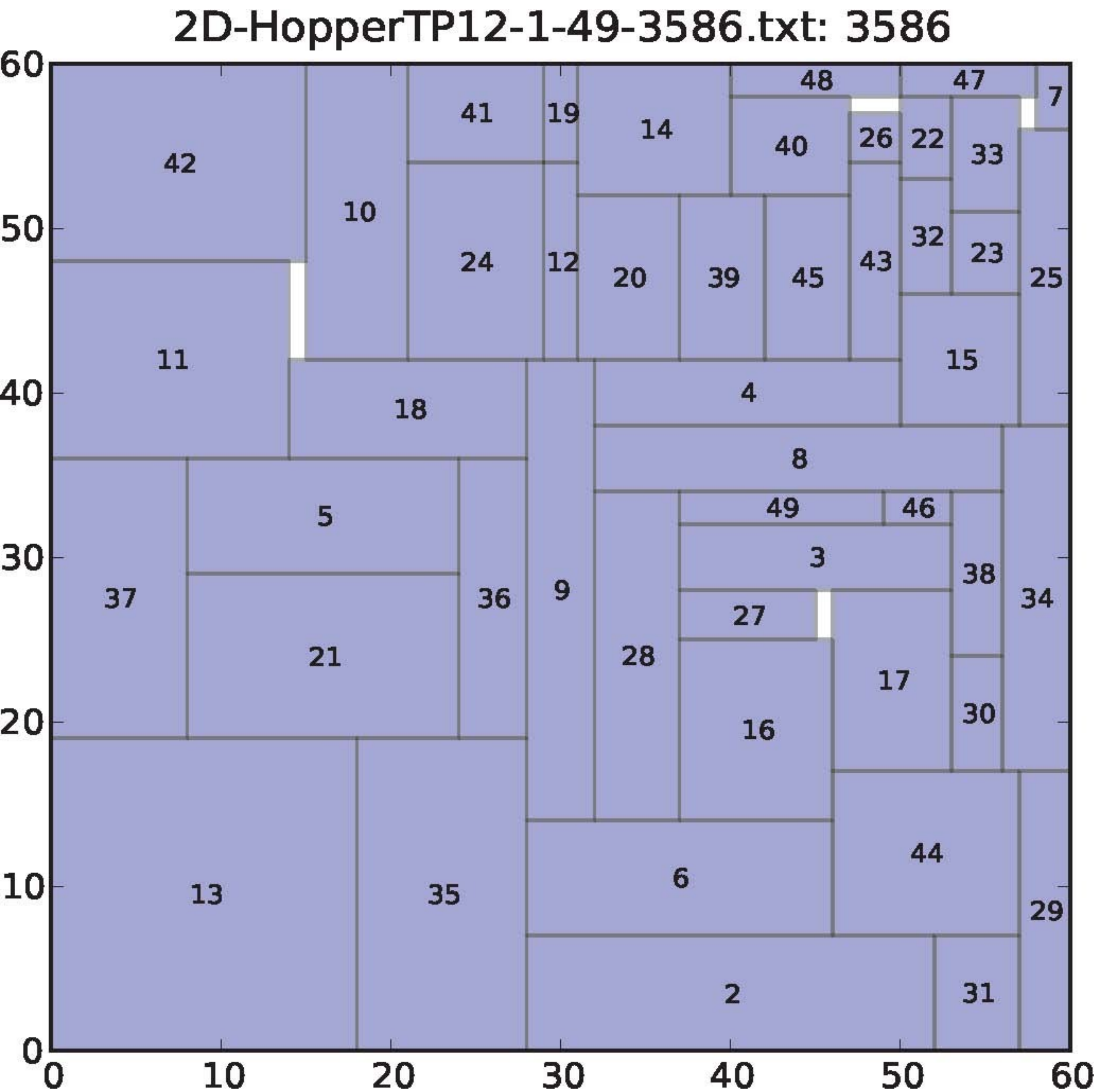
Hopper & Turton, 2001
Instance 4-1 60 x 60
Value: 3576

Previous best: 3580 by a Tabu Search heuristic
(Alvarez-Valdes et al., 2007)



Hopper & Turton, 2001
Instance 4-2 60 x 60
Value: 3585

Previous best: 3580 by a Tabu Search heuristic (Alvarez-Valdes et al., 2007)

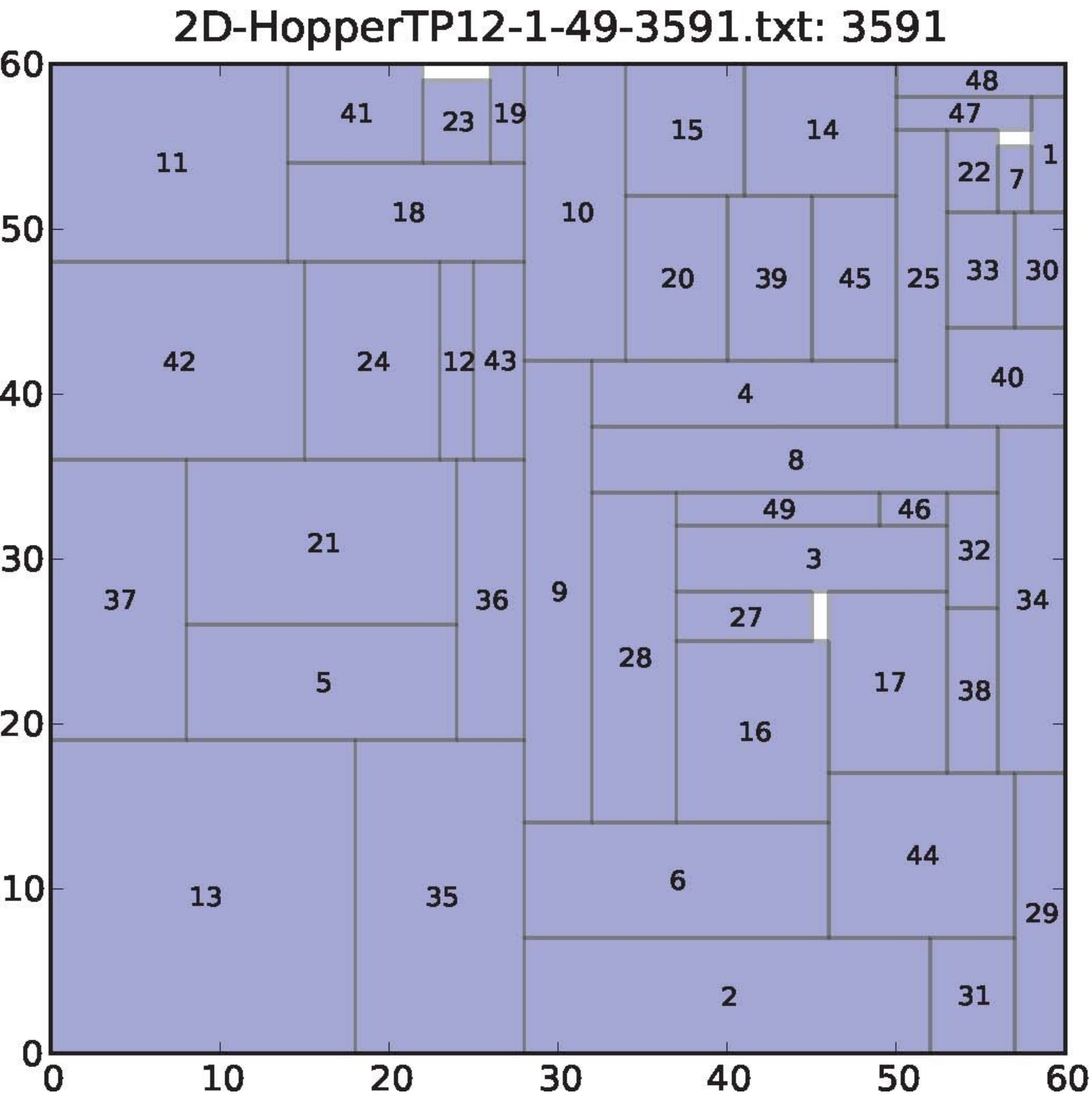


Hopper & Turton, 2001

Instance 4-2 60 x 60

Value: 3586

Previous best: 3580 by a Tabu Search heuristic (Alvarez-Valdes et al., 2007)

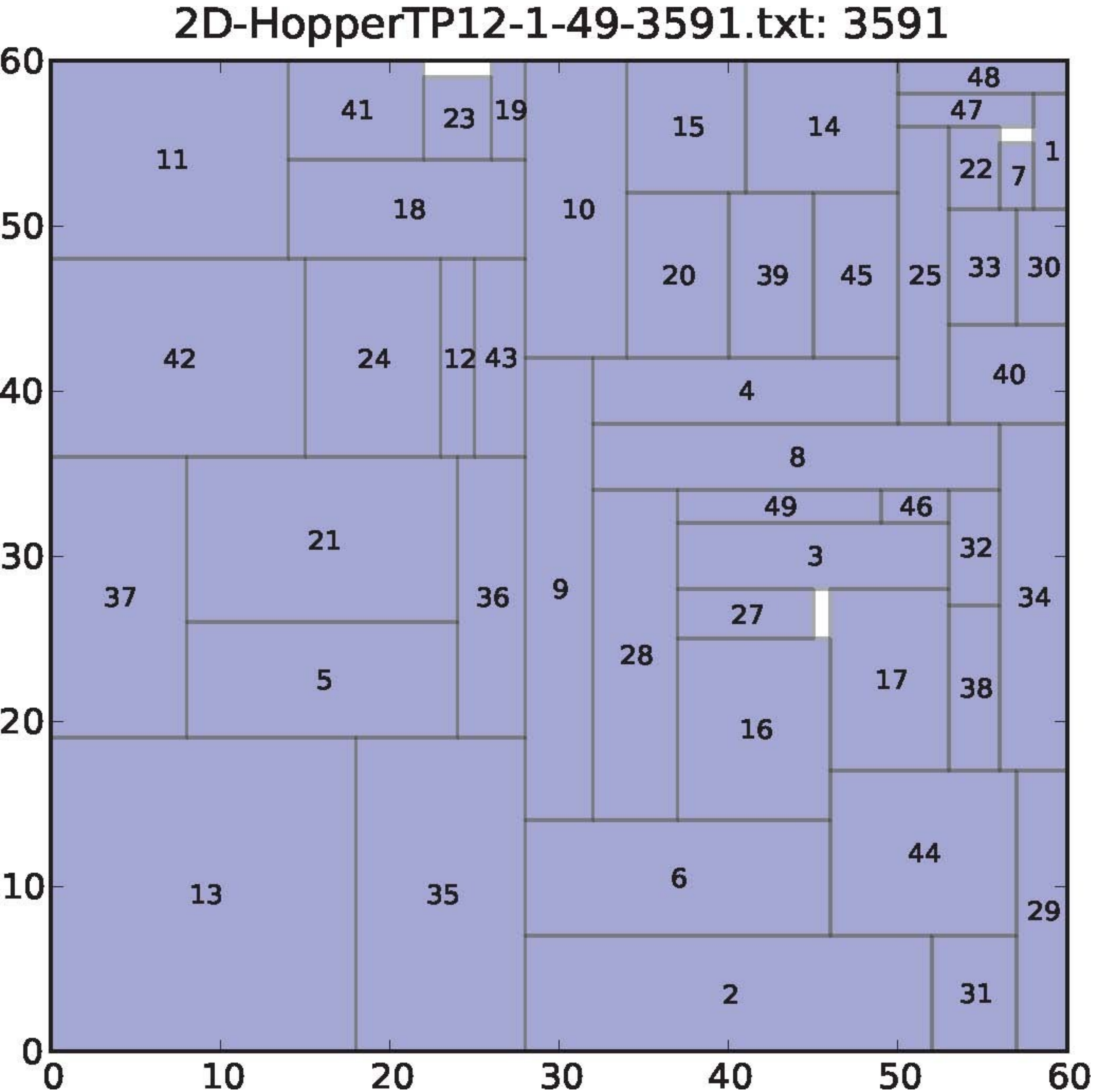


Hopper & Turton, 2001

Instance 4-2 60 x 60

Value: 3591

Previous best: 3580 by a Tabu Search heuristic (Alvarez-Valdes et al., 2007)



Hopper & Turton, 2001

Instance 4-2 60 x 60

Value: 3591

New best known solution!

Previous best: 3580 by a Tabu Search heuristic (Alvarez-Valdes et al., 2007)

BRKGA for constrained 2-dim orthogonal packing

Encoding

- Solutions are encoded as vectors X of

$$2N' = 2 \{ Q[1] + Q[2] + \cdots + Q[N] \}$$

random keys, where Q[i] is the maximum number of rectangles of type i (for i = 1, ..., N) that can be packed.

- X = (X[1], ..., X[N'], X[N'+1], ..., X[2N'])

Encoding

- Solutions are encoded as vectors X of
$$2N' = 2 \{ Q[1] + Q[2] + \cdots + Q[N] \}$$
random keys, where Q[i] is the maximum number of rectangles of type i (for i = 1, ..., N) that can be packed.

- $X = (\underbrace{X[1], \ldots, X[N'],}_{\text{Rectangle type packing sequence (RTPS)}} \quad X[N'+1], \ldots, X[2N'])$

Encoding

- Solutions are encoded as vectors X of

$$2N' = 2 \{ Q[1] + Q[2] + \cdots + Q[N] \}$$

  random keys, where Q[i] is the maximum number of rectangles of type i (for i = 1, ..., N) that can be packed.

- $X = (\underbrace{X[1], ..., X[N'],}_{\text{Rectangle type packing sequence (RTPS)}} \quad \underbrace{X[N'+1], ..., X[2N']}_{\text{Vector of placement procedures (VPP)}})$

Decoding

- Simple heuristic to pack rectangles:
  - Make Q[i] copies of rectangle i, for i = 1, ..., N.
  - Order the N' = Q[1] + Q[2] + $\cdots$ + Q[N] rectangles in some way.
  - Process the rectangles in the above order. Place the rectangle in the stock rectangle according to one of the following heuristics: bottom-left (BL) or left-bottom (LB). If rectangle cannot be positioned, discard it and go on to the next rectangle in the order.

Decoding

- Simple heuristic to pack rectangles:
  - Make Q[i] copies of rectangle i, for i = 1, ..., N.
  - Order the N' = Q[1] + Q[2] + ⋯ + Q[N] rectangles in some way. **Sort first N' keys to obtain order.**
  - Process the rectangles in the above order. Place the rectangle in the stock rectangle according to one of the following heuristics: bottom-left (BL) or left-bottom (LB). If rectangle cannot be positioned, discard it and go on to the next rectangle in the order.

Decoding

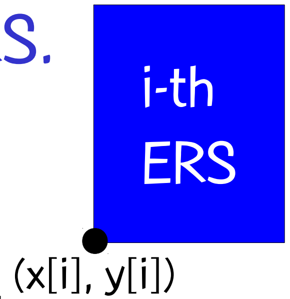
- Simple heuristic to pack rectangles:
  - Make Q[i] copies of rectangle i, for i = 1, ..., N.
  - Order the N' = Q[1] + Q[2] + ··· + Q[N] rectangles in some way. **Sort first N' keys to obtain order.**
  - Process the rectangles in the above order. Place the rectangle in the stock rectangle according to one of the following heuristics: bottom-left (BL) or left-bottom (LB). If rectangle cannot be positioned, discard it and go on to the next rectangle in the order. **Use the last N' keys to determine which heuristic to use. If X[N'+i] > 0.5 use LB, else use BL.**

Decoding

- A maximal empty rectangular space (ERS) is an empty rectangular space not contained in any other ERS.
- ERSs are generated and updated using the Difference Process of Lai and Chan (1997).
- When placing a rectangle, we limit ourselves only to maximal ERSs. We order all the maximal ERSs and place the rectangle in the first maximal ERS in which it fits.
- Let (x[i], y[i]) be the coordinates of the bottom left corner of the i-th ERS.

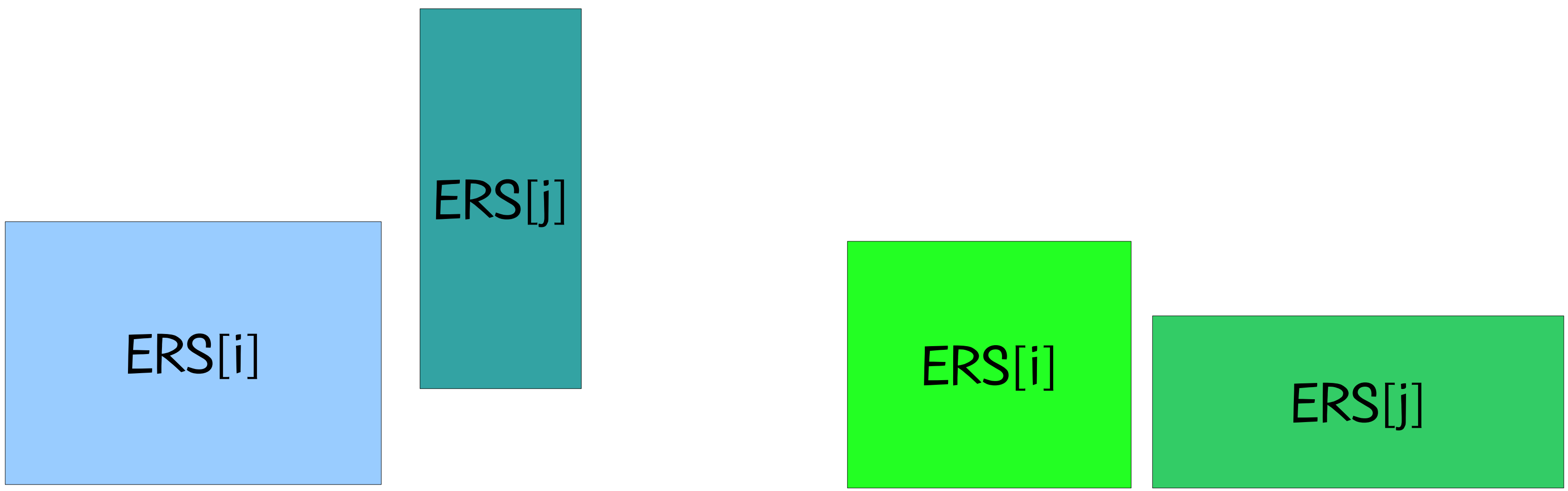
Decoding

- A maximal empty rectangular space (ERS) is an empty rectangular space not contained in any other ERS.
- ERSs are generated and updated using the Difference Process of Lai and Chan (1997).
- When placing a rectangle, we limit ourselves only to maximal ERSs. We order all the maximal ERSs and place the rectangle in the first maximal ERS in which it fits.
- Let (x[i], y[i]) be the coordinates of the bottom left corner of the i-th ERS.

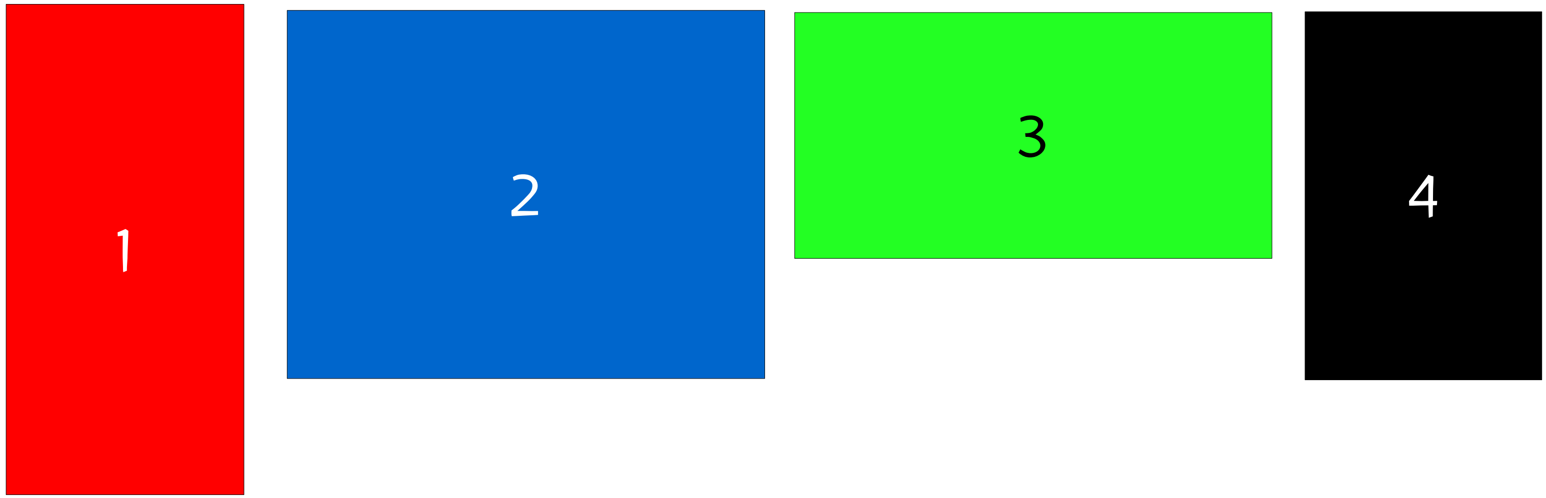


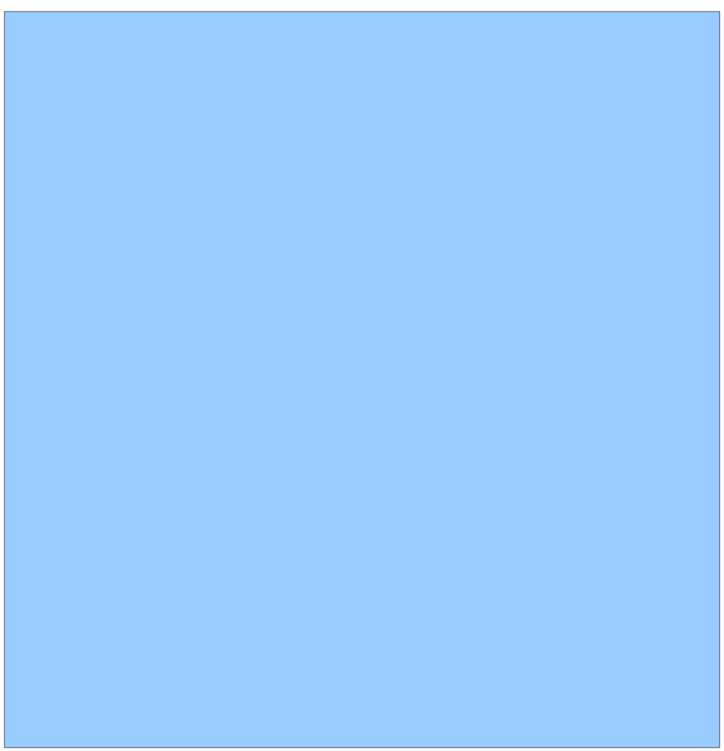
Decoding

- If BL is used, ERSs are ordered such that ERS[i] < ERS[j] if y[i] < y[j] or y[i] = y[j] and x[i] < x[j].



ERS[i] < ERS[j]

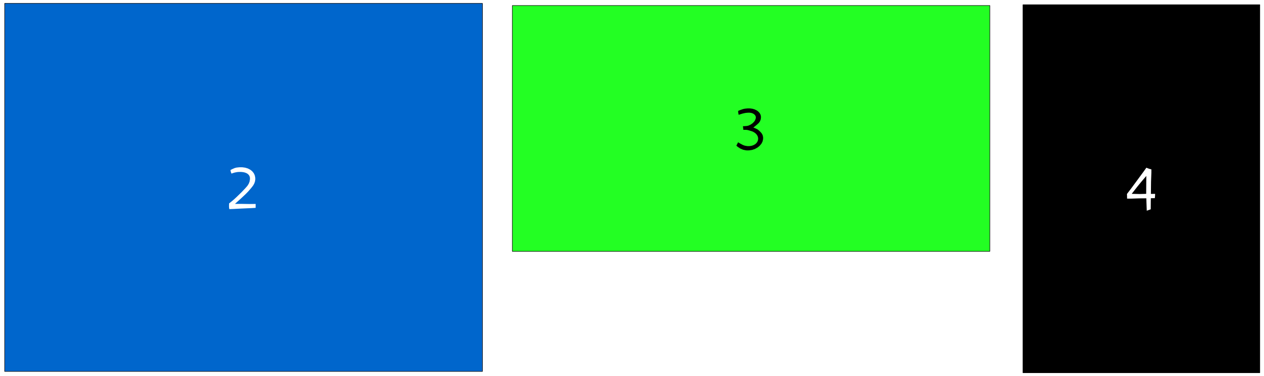


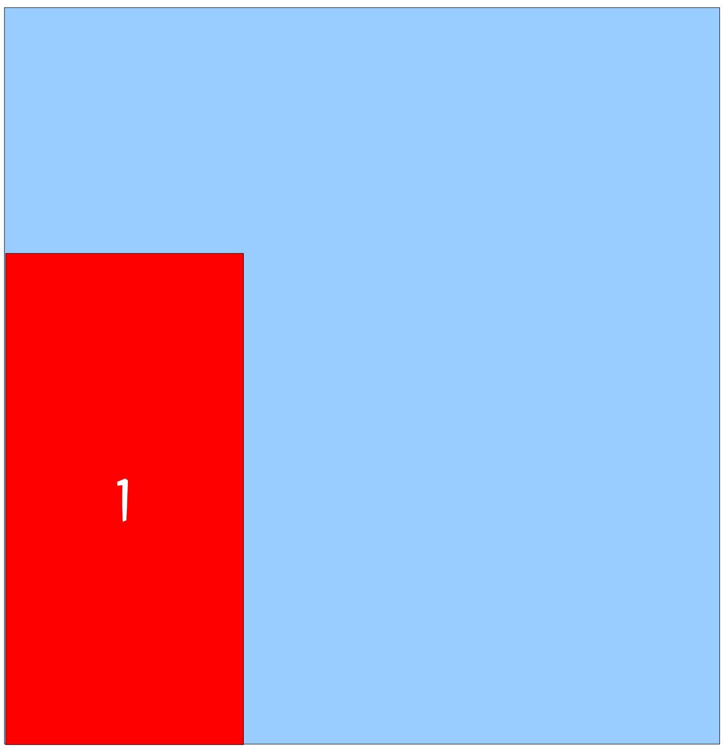


BL can run into problems even on small instances (Liu & Teng, 1999).

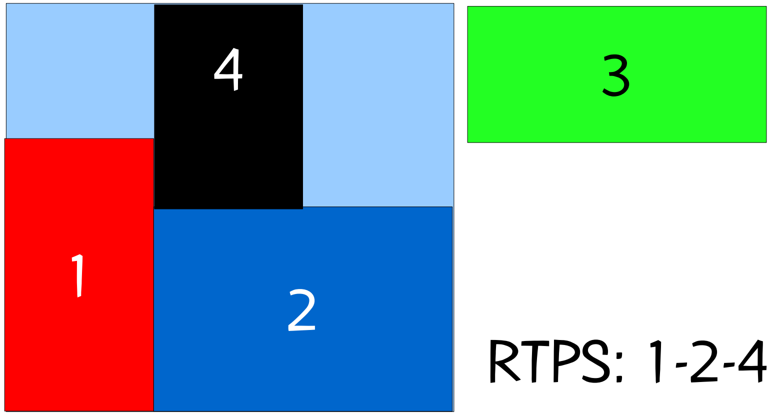
Consider this instance with 4 rectangles.

BL cannot find the optimal solution for any RTPS.

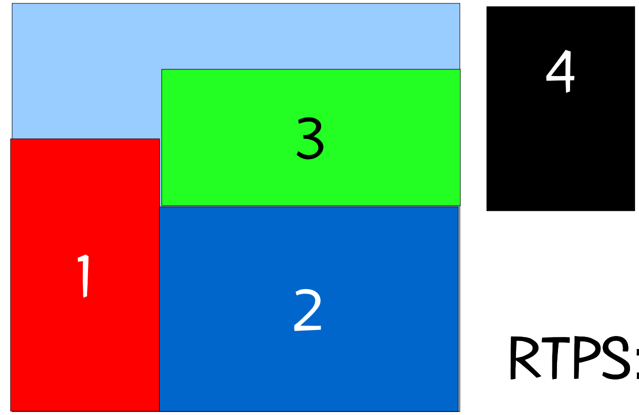




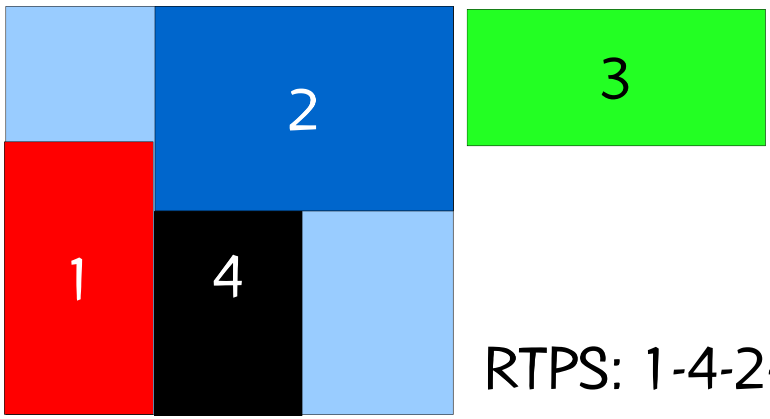
We show 6 rectangle type packing sequences (RTPS's) where we fix rectangle 1 in the first position.



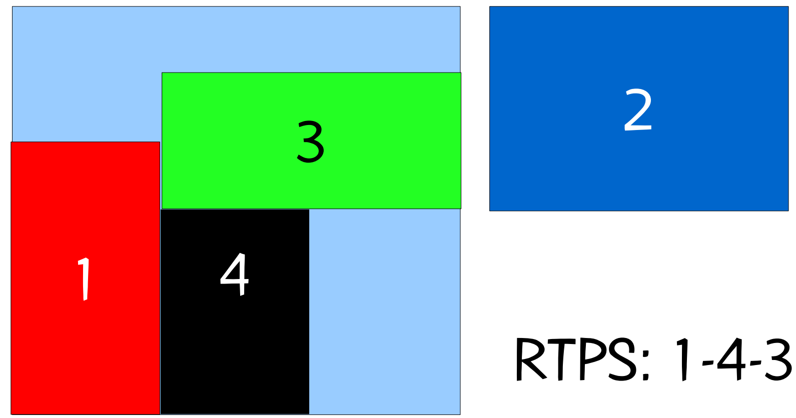
RTPS: 1-2-4-3



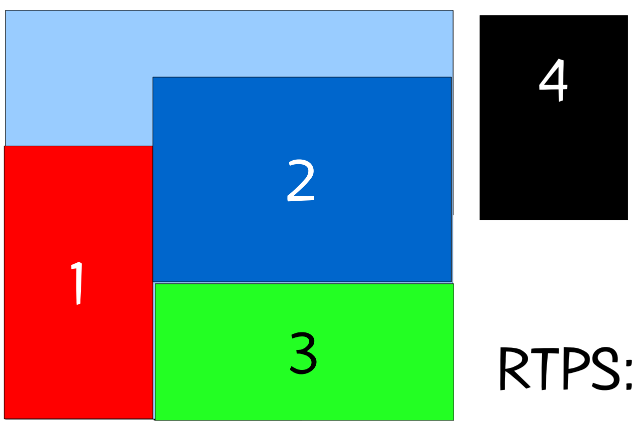
RTPS: 1-2-3-4



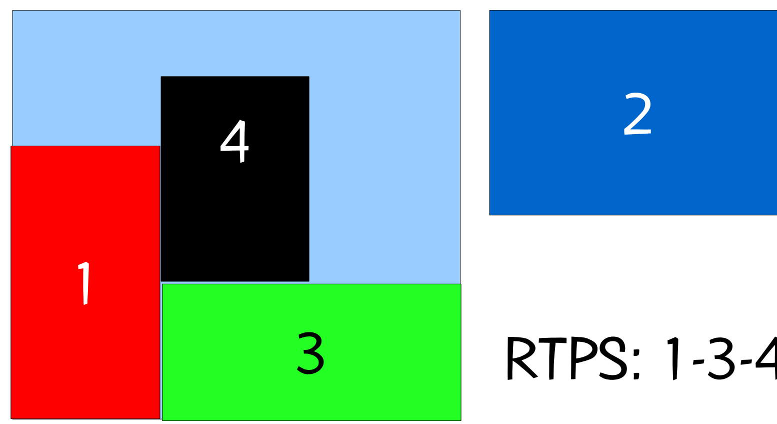
RTPS: 1-4-2-3



RTPS: 1-4-3-2

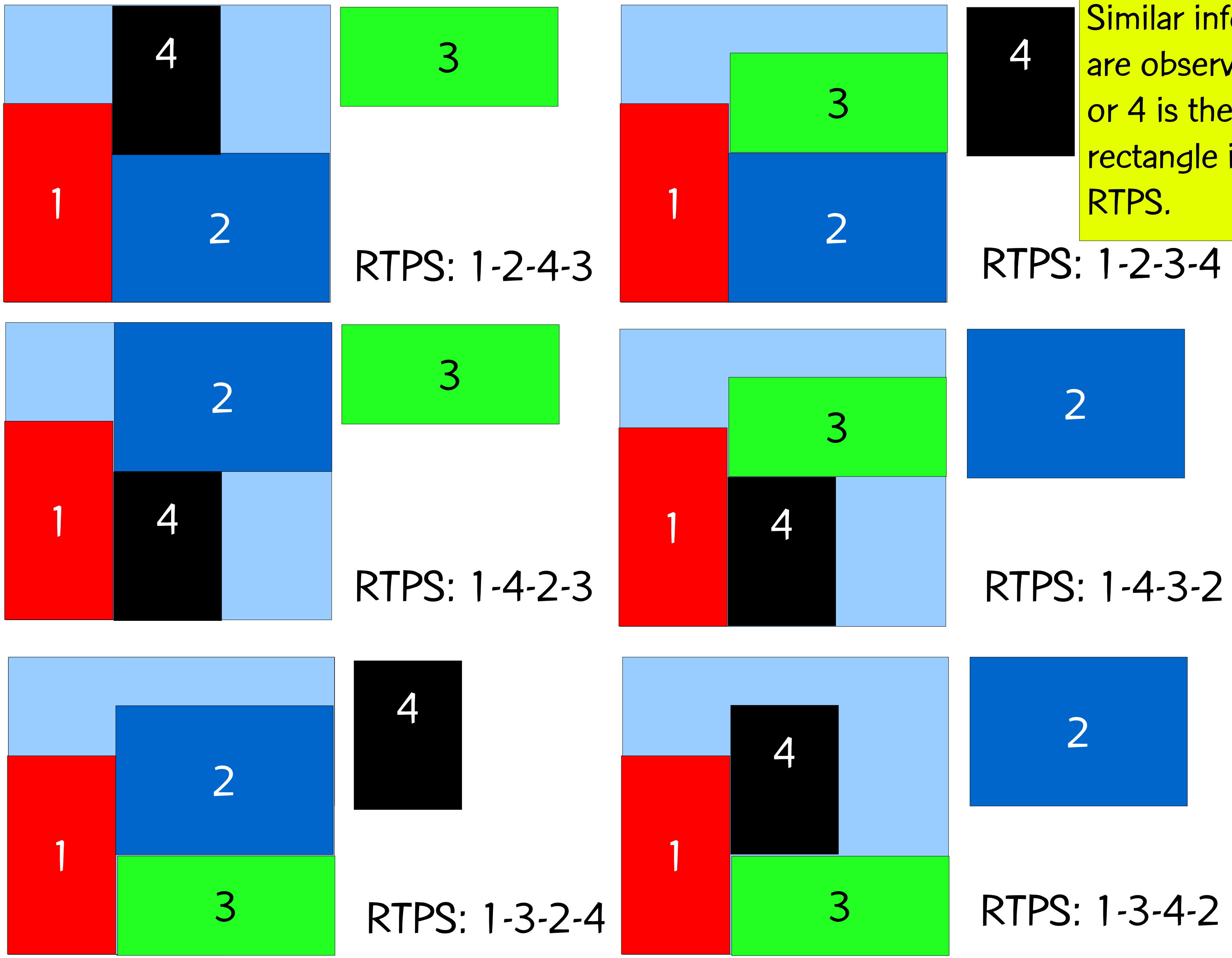


RTPS: 1-3-2-4



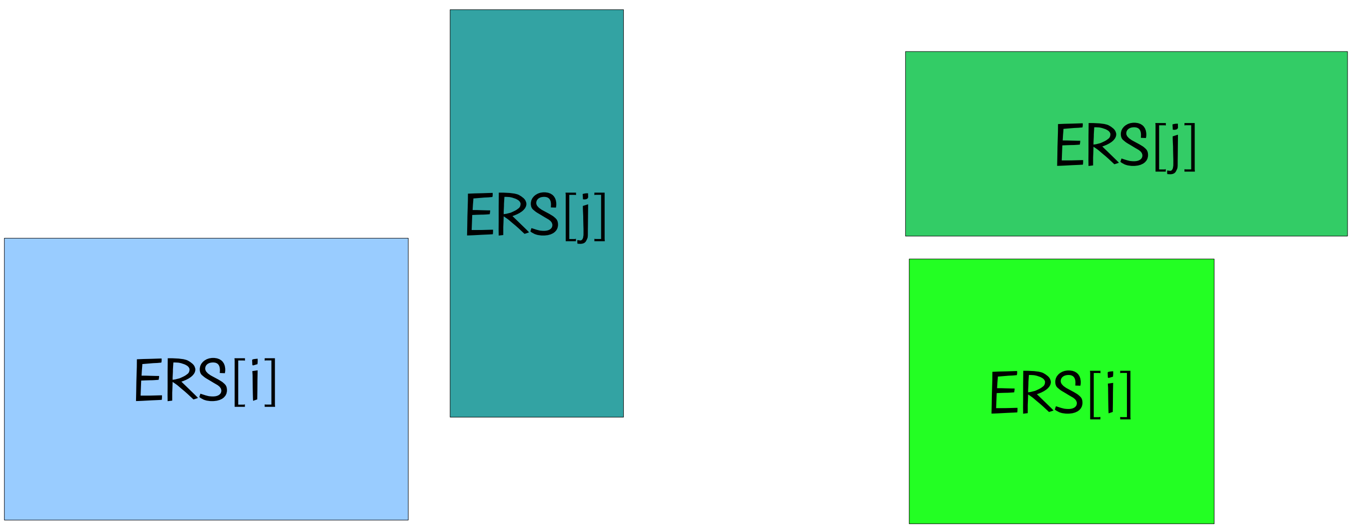
RTPS: 1-3-4-2

Similar infeasibilities are observed if 2, 3, or 4 is the first rectangle in the RTPS.

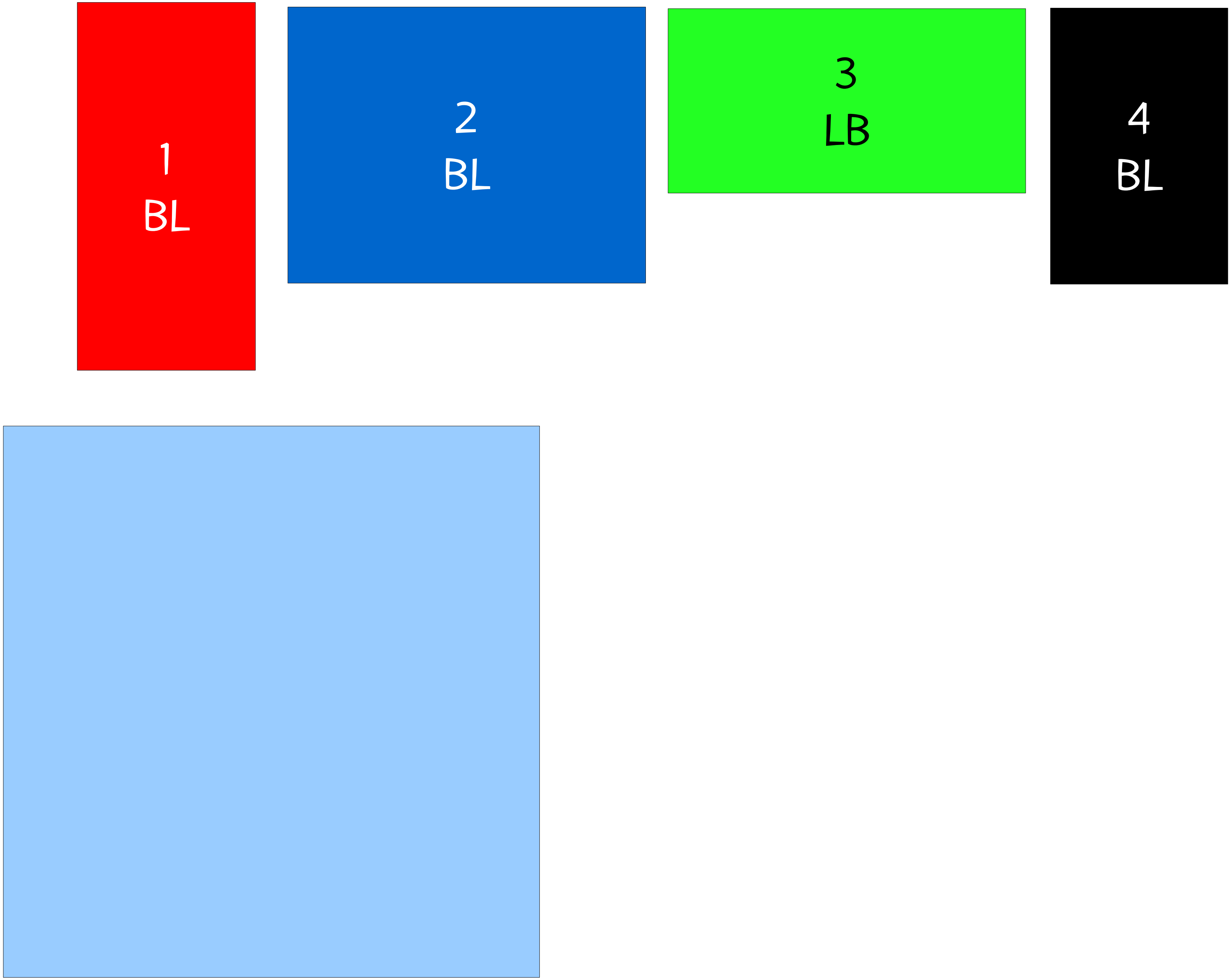


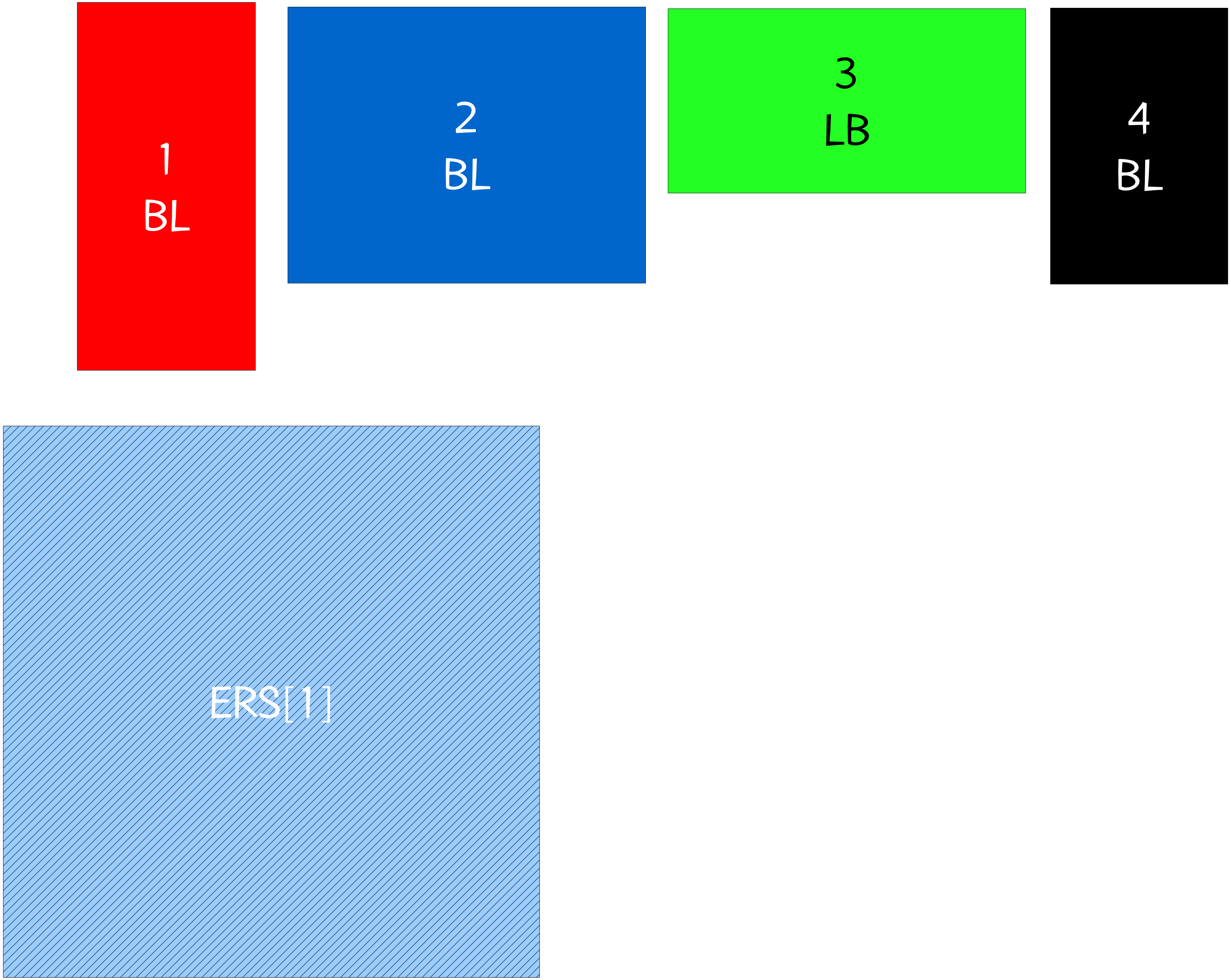
Decoding

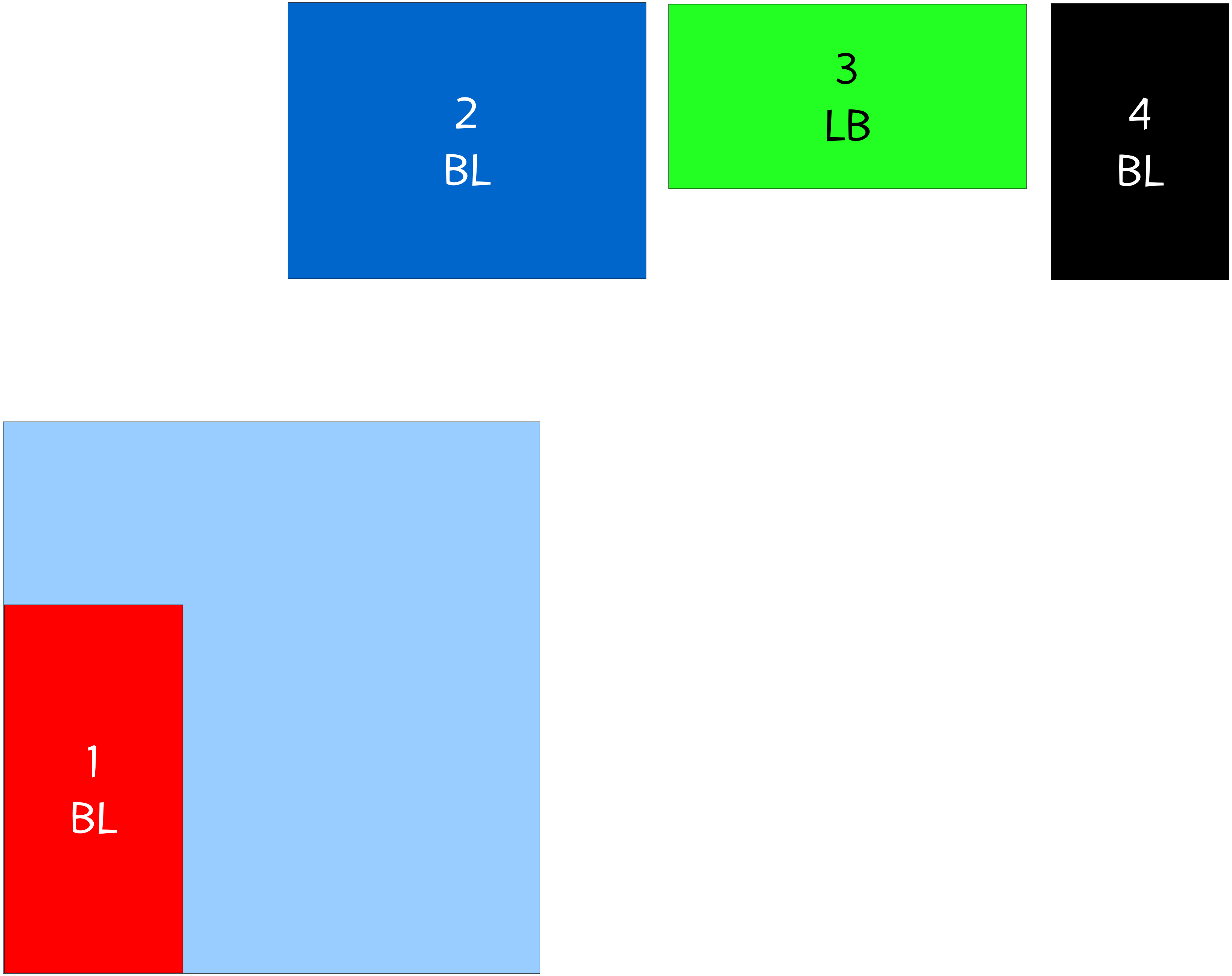
- If LB is used, ERSs are ordered such that ERS[i] < ERS[j] if x[i] < x[j] or x[i] = x[j] and y[i] < y[j].

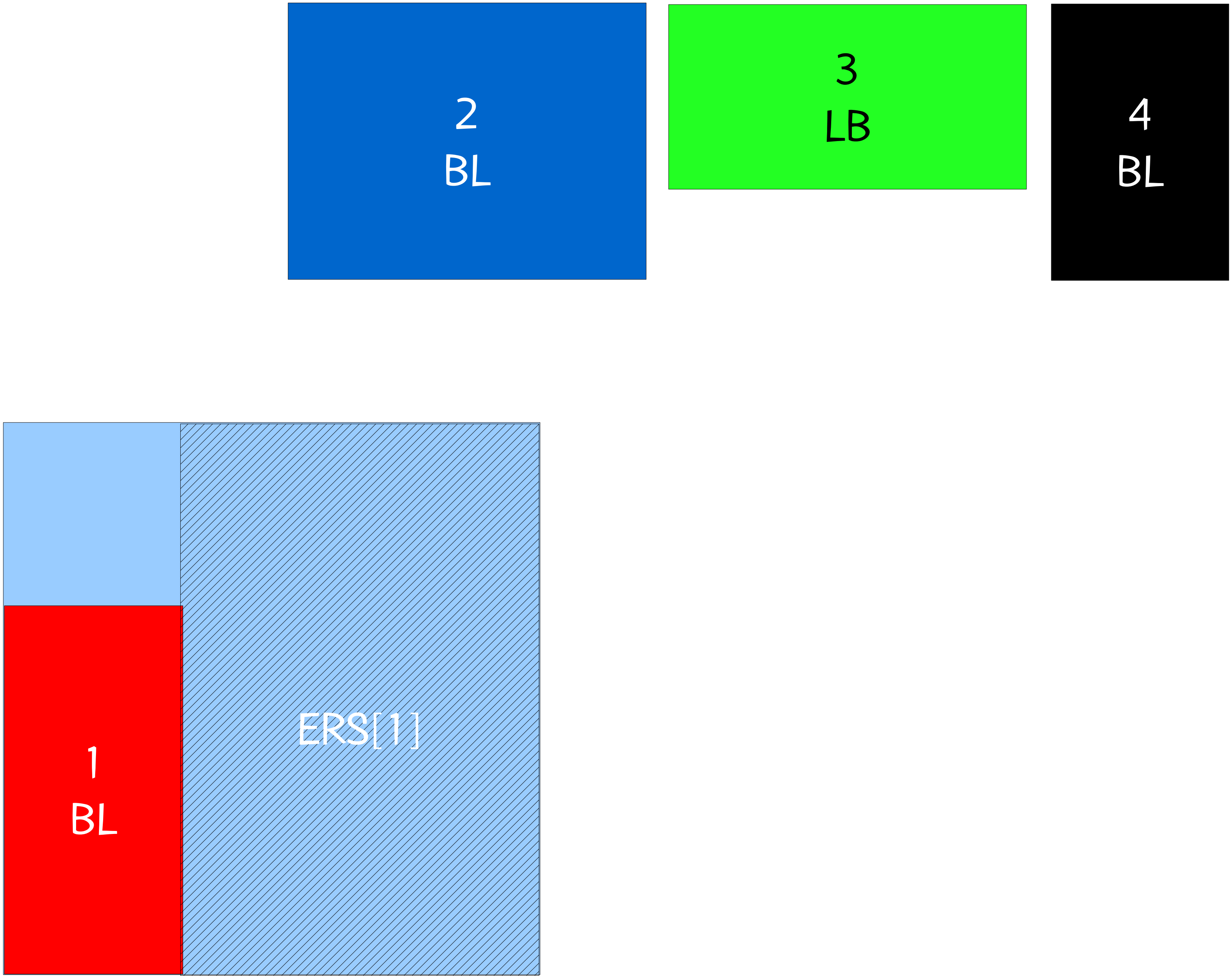


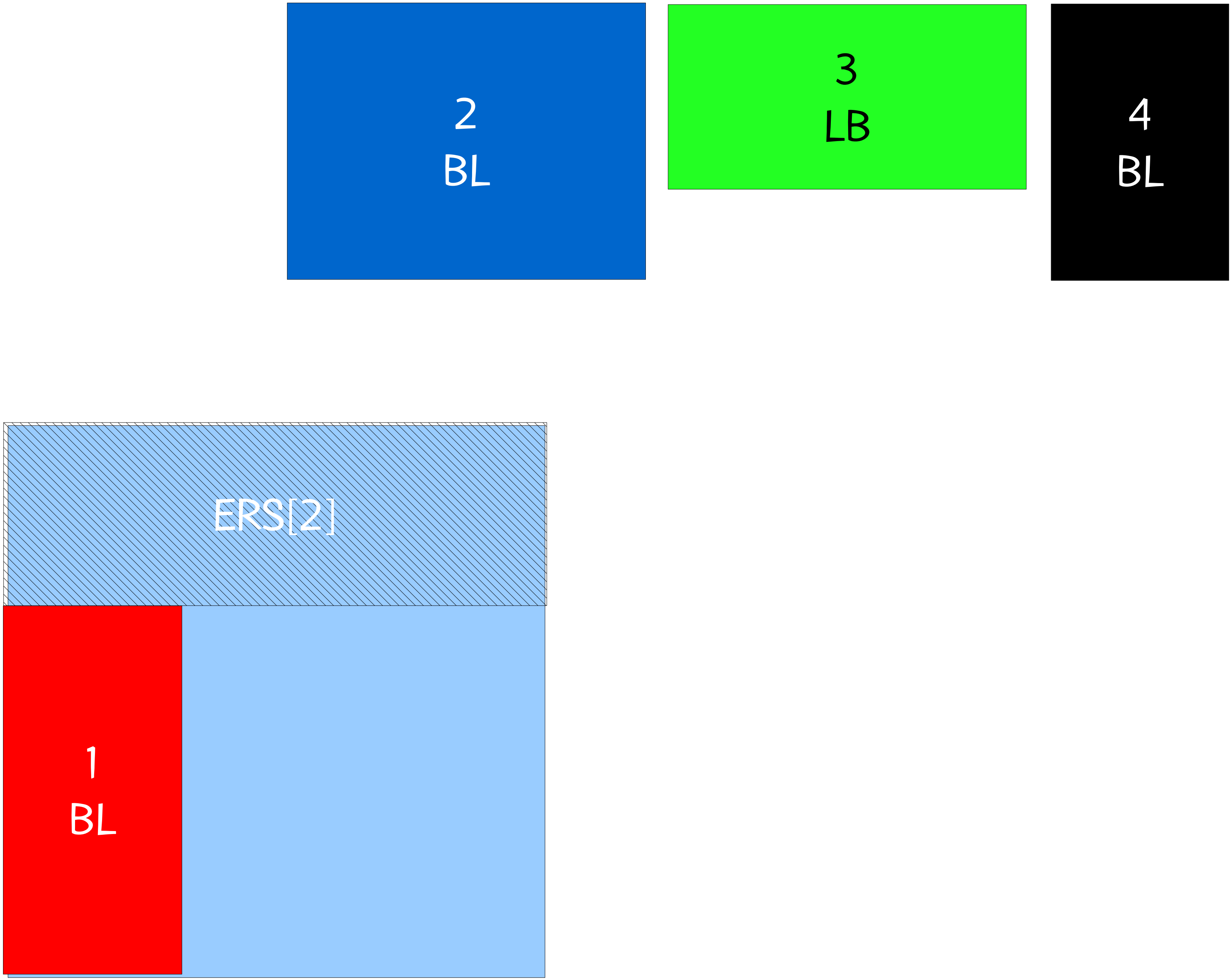
ERS[i] < ERS[j]

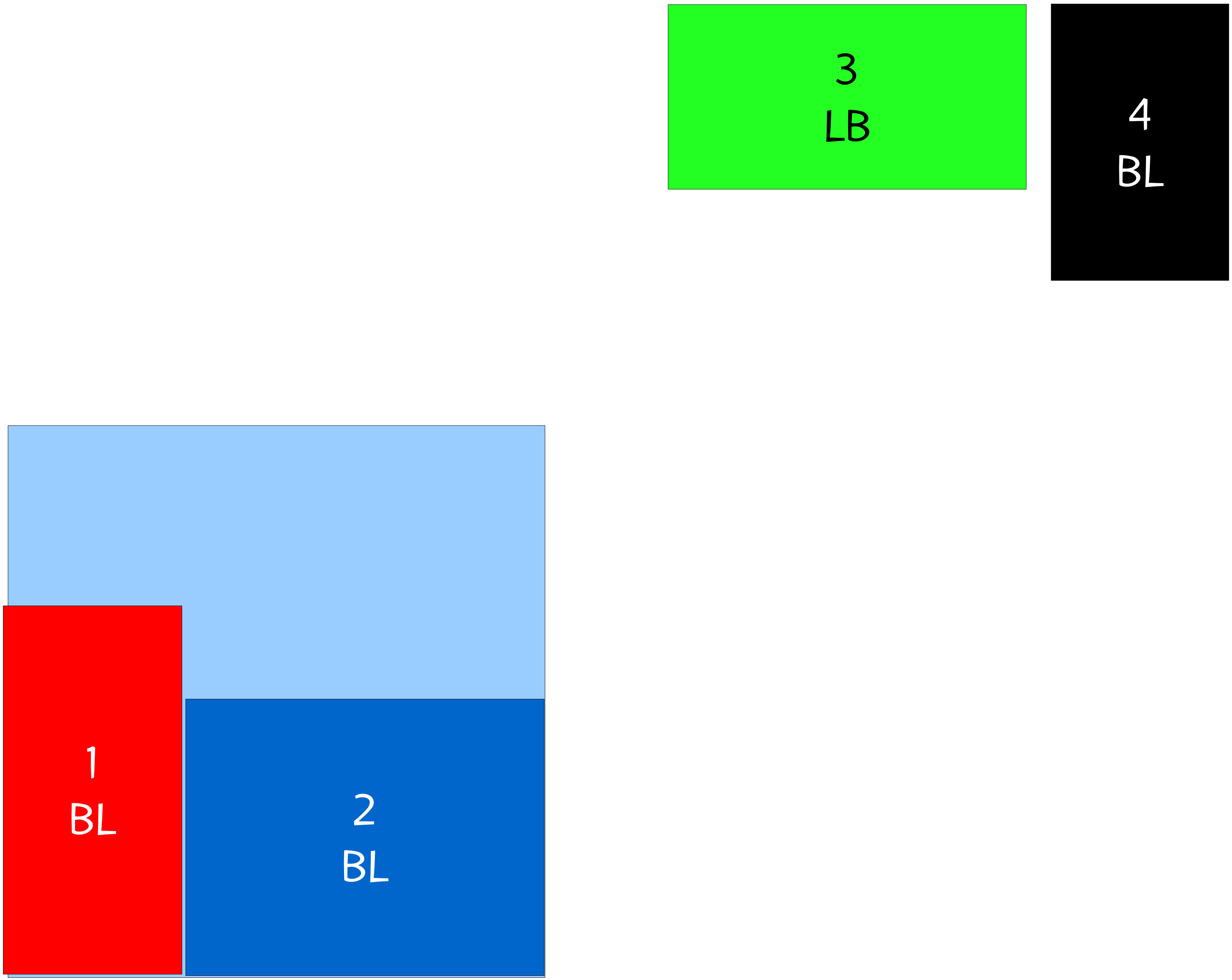

1
BL
2
BL
3
LB
4
BL

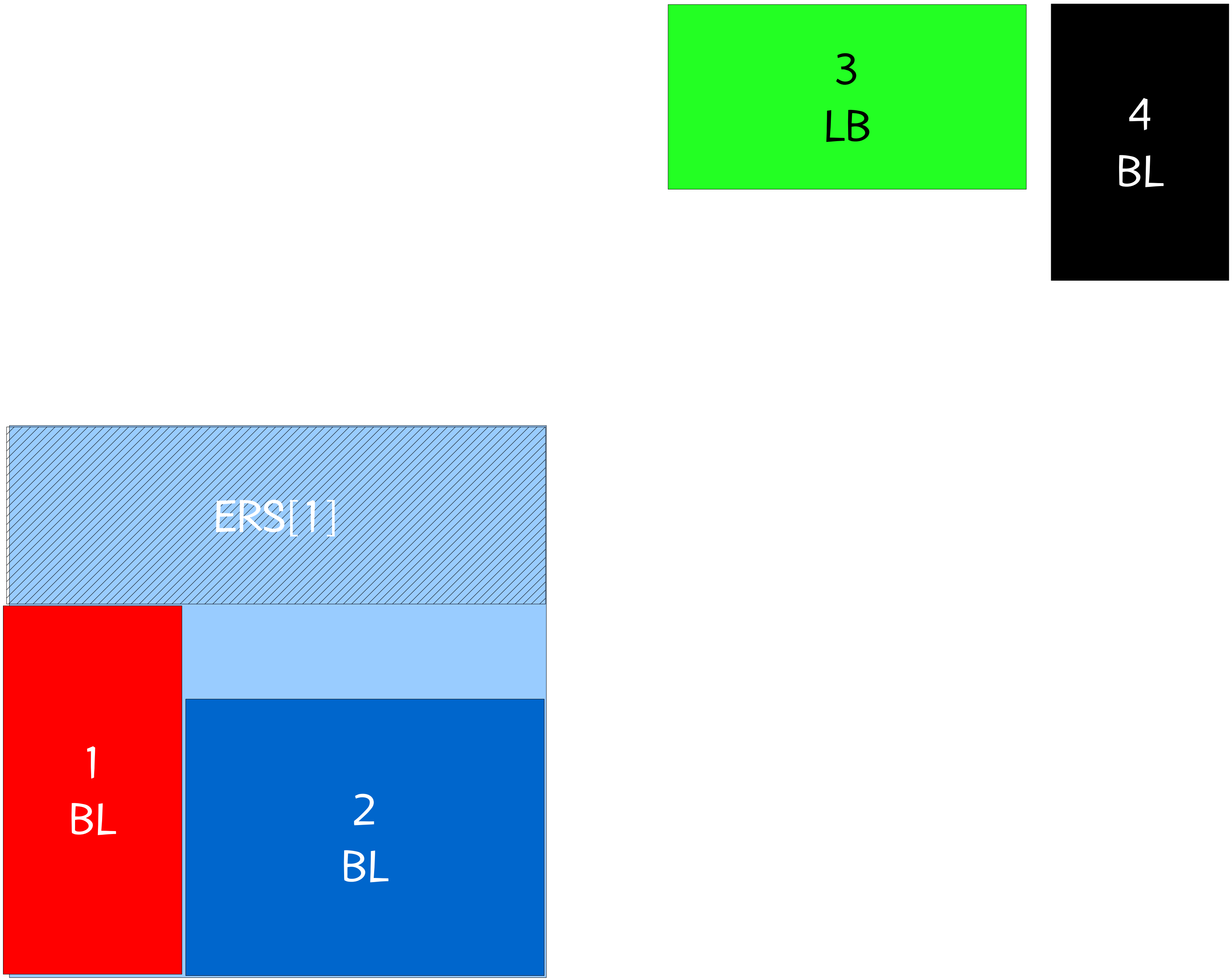

1
BL
2
BL
3
LB
4
BL
ERS[1]

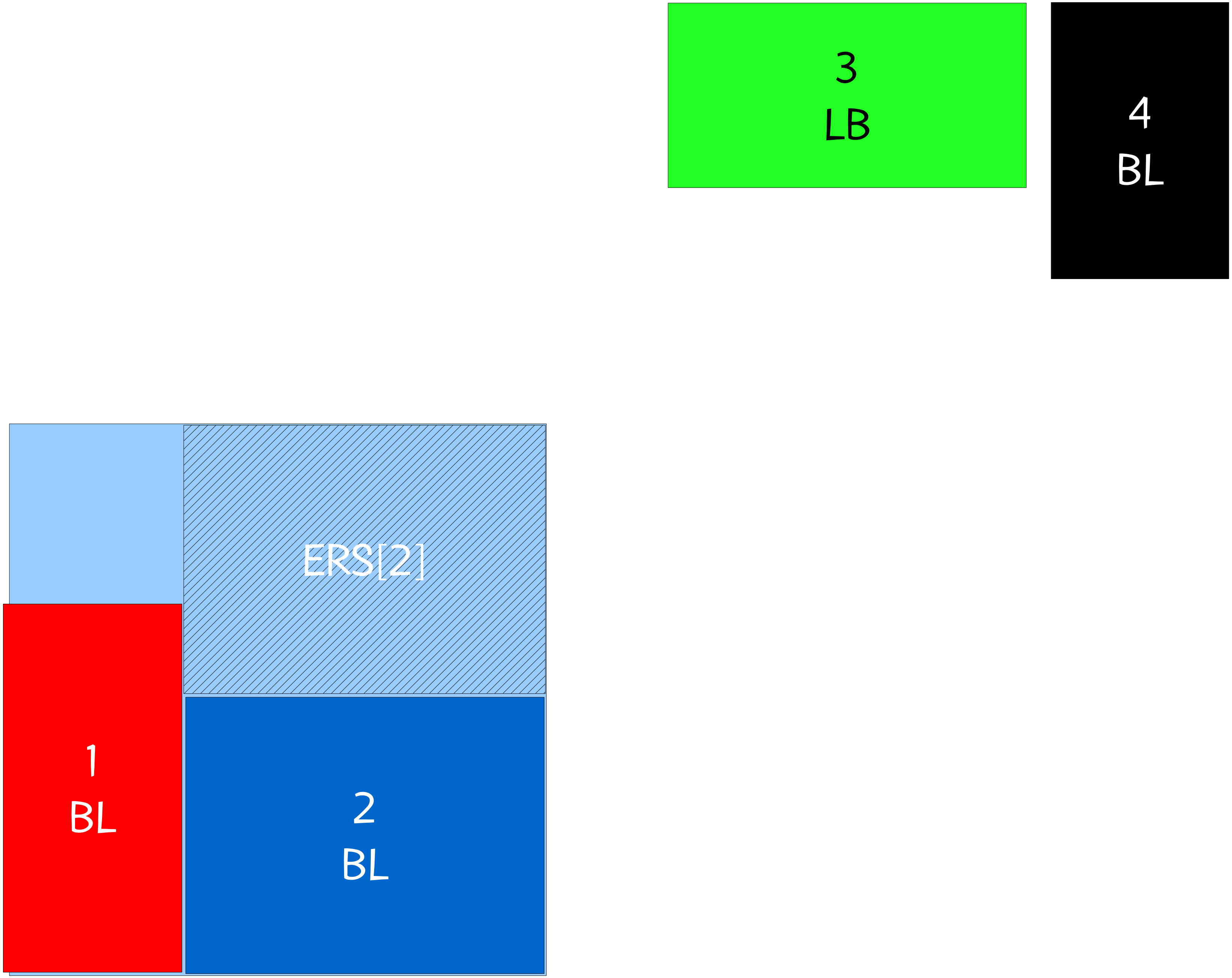

2
BL
3
LB
4
BL
1
BL

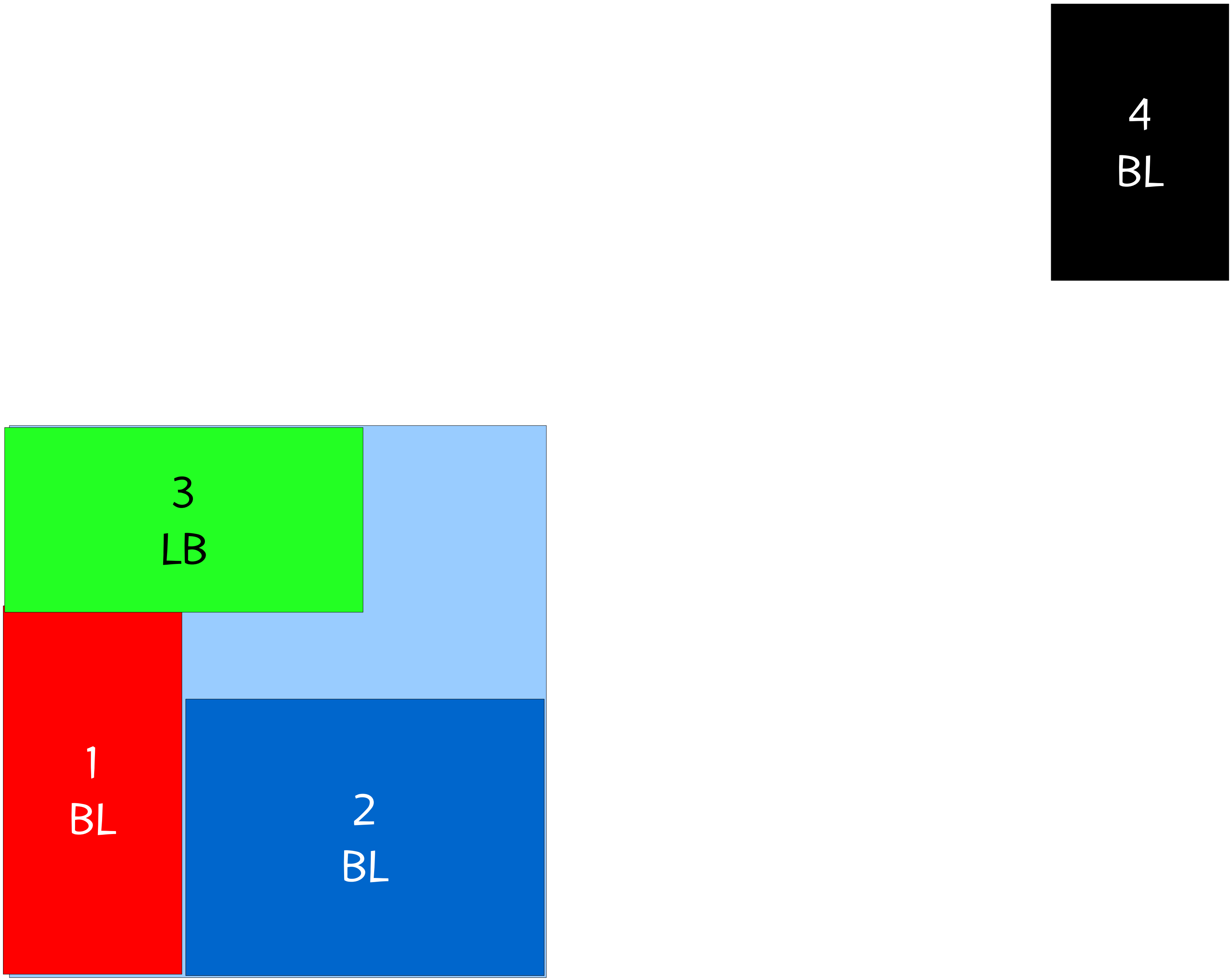

2
BL
3
LB
4
BL
ERS[1]
1
BL

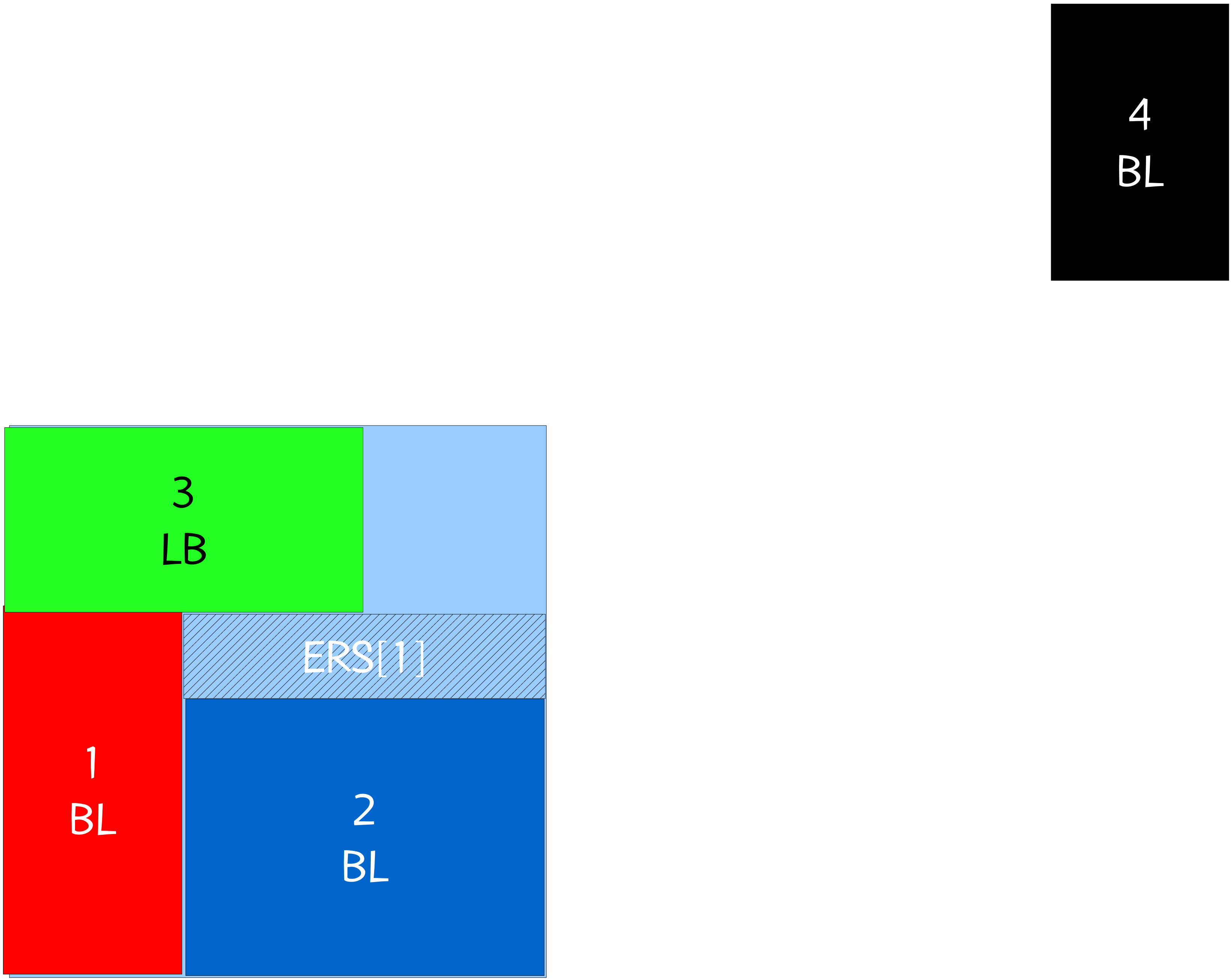

2
BL
3
LB
4
BL
ERS[2]
1
BL

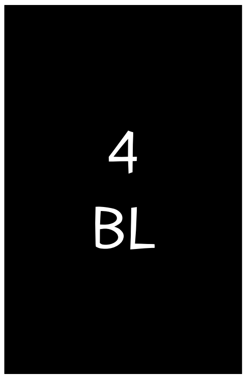

3
LB
4
BL
1
BL
2
BL


3
LB
4
BL
ERS[1]
1
BL
2
BL

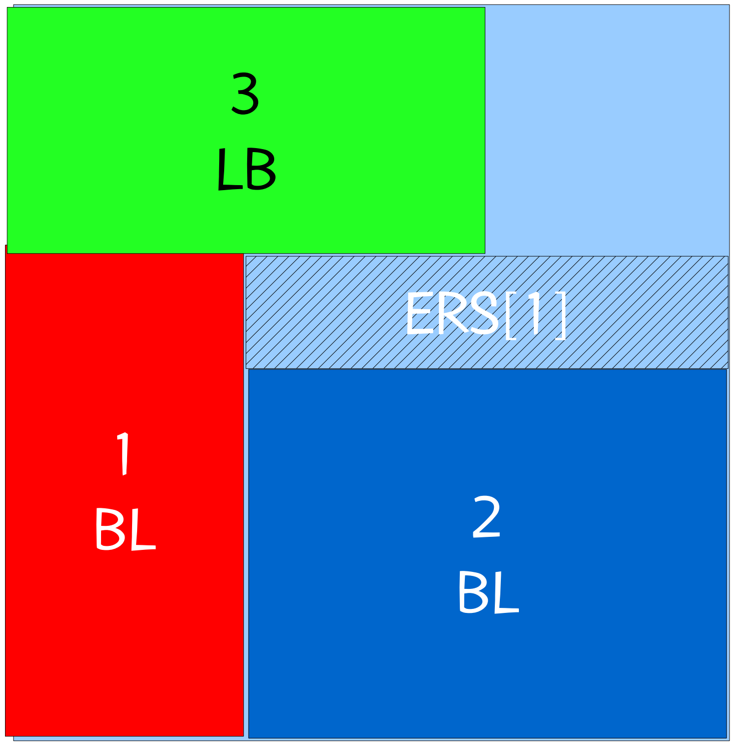

3
LB
4
BL
ERS[2]
1
BL
2
BL

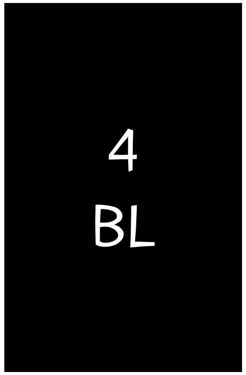

4
BL
3
LB
1
BL
2
BL


4
BL
3
LB
ERS[1]
1
BL
2
BL

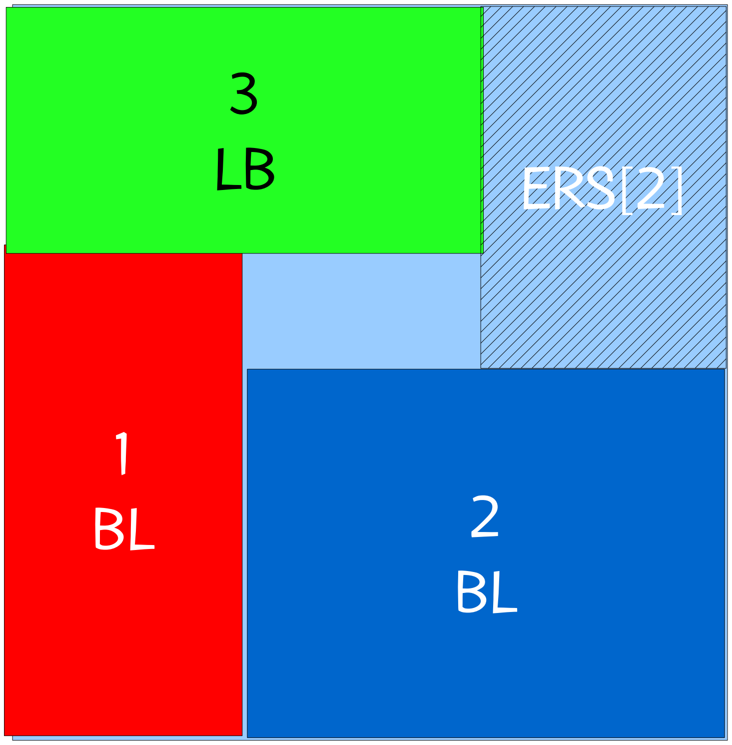


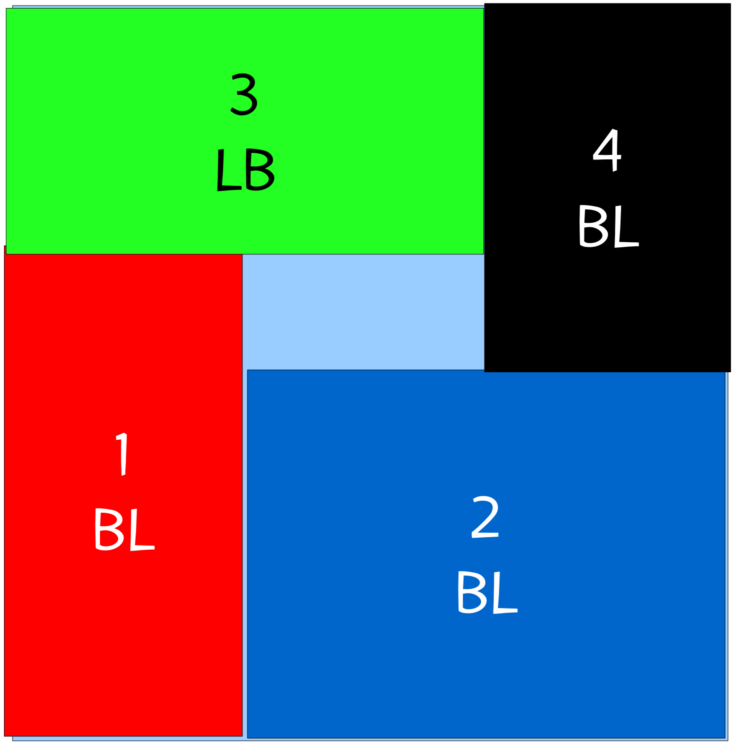
4 does not fit in ERS[1].





4 does fit
in ERS[2].





Optimal solution!

Experimental results

Design

- We compare solution values obtained by the parallel multi-population BRKGA with solutions obtained by the heuristics that produced the best computational results to date:

Design

- We compare solution values obtained by the parallel multi-population BRKGA with solutions obtained by the heuristics that produced the best computational results to date:
  - PH: population-based heuristic of Beasley (2004)

Design

- We compare solution values obtained by the parallel multi-population BRKGA with solutions obtained by the heuristics that produced the best computational results to date:
  - PH: population-based heuristic of Beasley (2004)
  - GA: genetic algorithm of Hadjiconsantinou & Iori (2007)

Design

- We compare solution values obtained by the parallel multi-population BRKGA with solutions obtained by the heuristics that produced the best computational results to date:
  - PH: population-based heuristic of Beasley (2004)
  - GA: genetic algorithm of Hadjiconsantinou & Iori (2007)
  - GRASP: greedy randomized adaptive search procedure of Alvarez-Valdes et al. (2005)

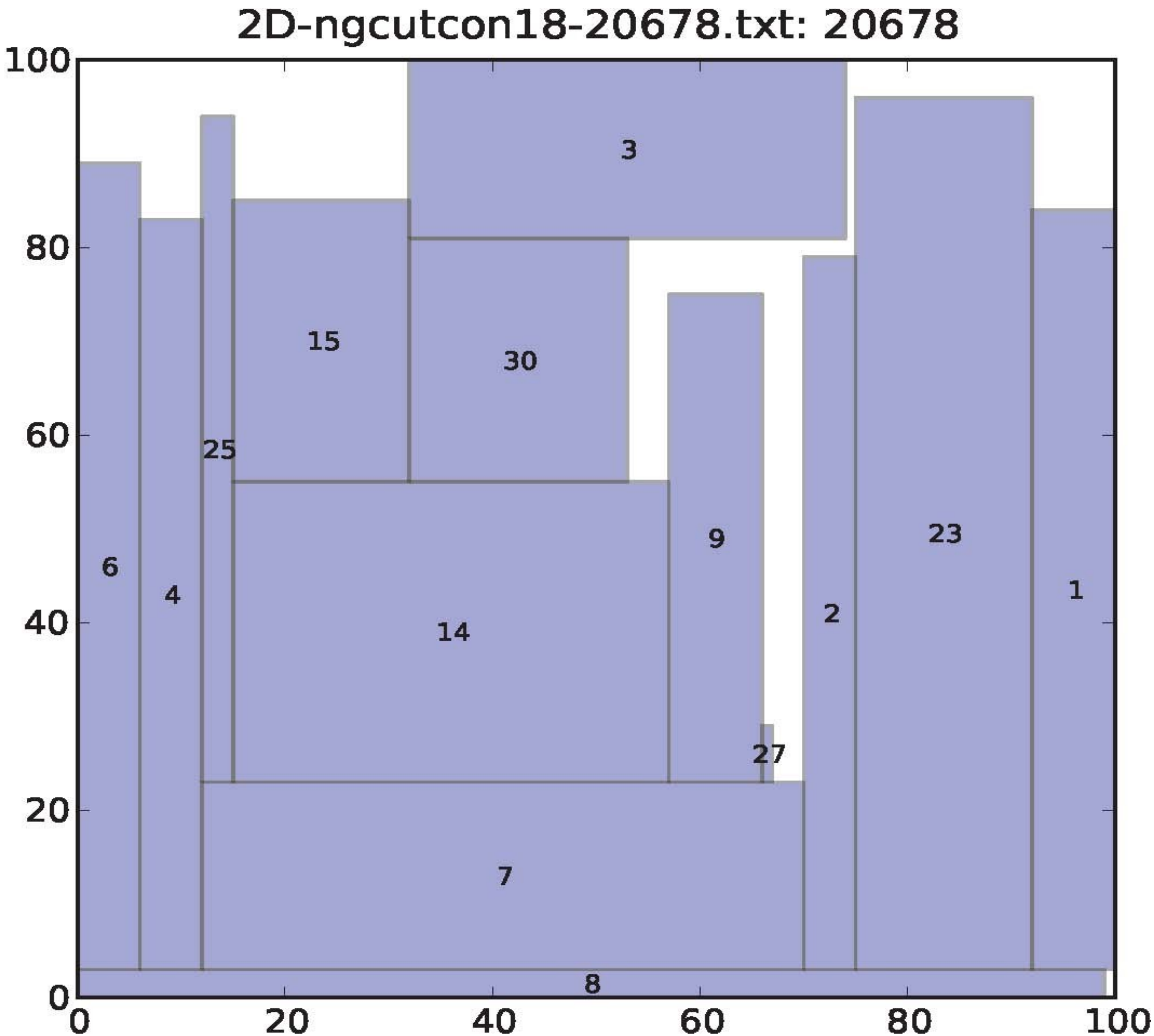
Design

- We compare solution values obtained by the parallel multi-population BRKGA with solutions obtained by the heuristics that produced the best computational results to date:
  - PH: population-based heuristic of Beasley (2004)
  - GA: genetic algorithm of Hadjiconsantinou & Iori (2007)
  - GRASP: greedy randomized adaptive search procedure of Alvarez-Valdes et al. (2005)
  - TABU: tabu search of Alvarez-Valdes et al. (2007)

Number of best solutions / total instances

Problem	PH	GA	GRASP	TABU	BRKGA BL-LB-L-4NR
From literature (optimal)	13/21	**21/21**	18/21	**21/21**	**21/21**
Large random*	0/21	0/21	5/21	8/21	**20/21**
Zero-waste			5/31	17/31	**30/31**
Doubly constrained	11/21		12/21	17/21	**19/21**

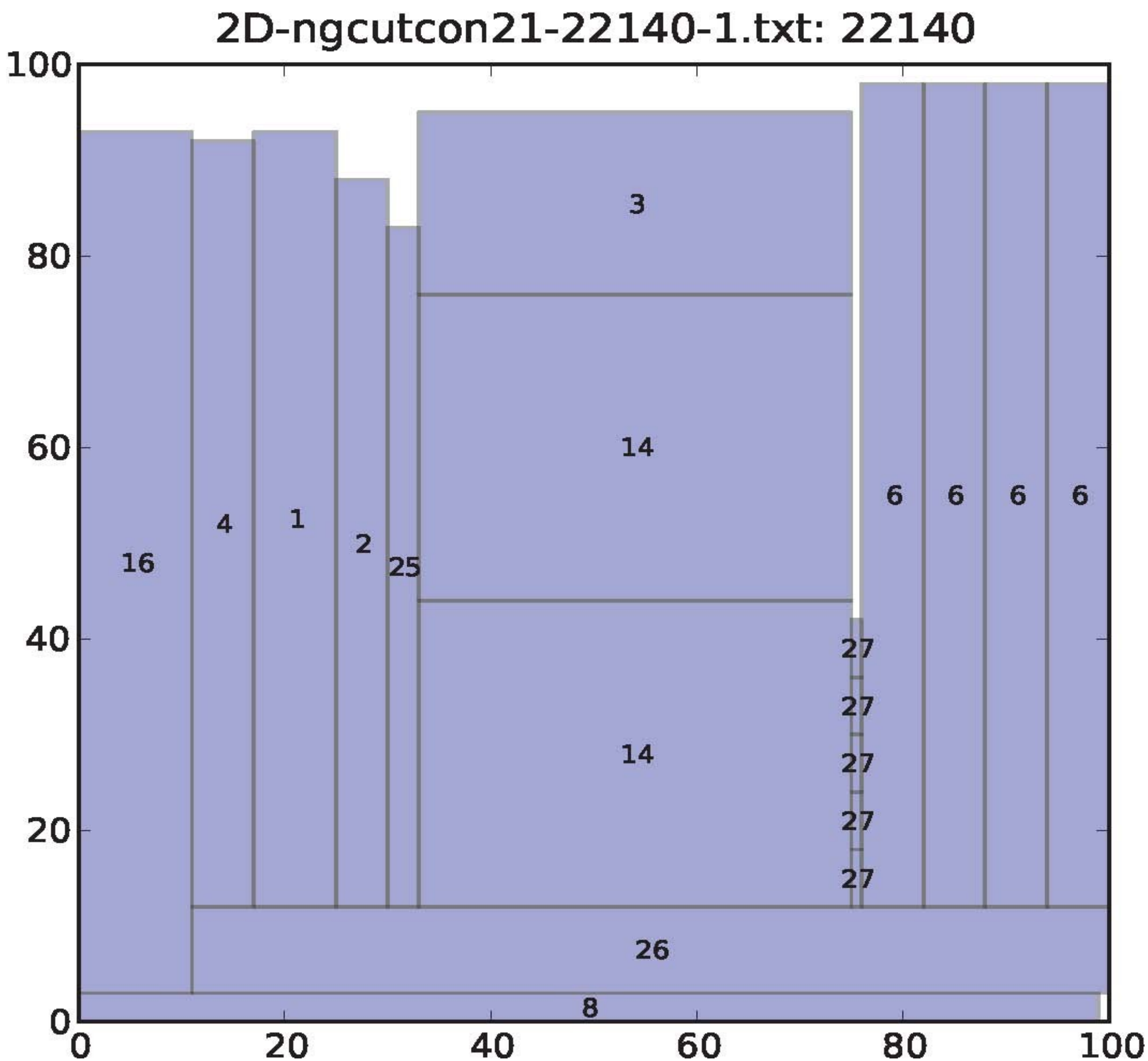
* For large random: number of best average solutions / total instance classes



New BKS for a 100 x100 doubly constrained instance of Fekete & Schepers (1997) of value **20678.** Previous best was **19657** by tabu search of Alvarez-Valdes et al., (2007).

30 types

30 rectangles



New BKS for a 100 x 100 doubly constrained instance Fekete & Schepers (1997) of value **22140**.

Previous BKS was **22011** by tabu search of Alvarez-Valdes et al. (2007).

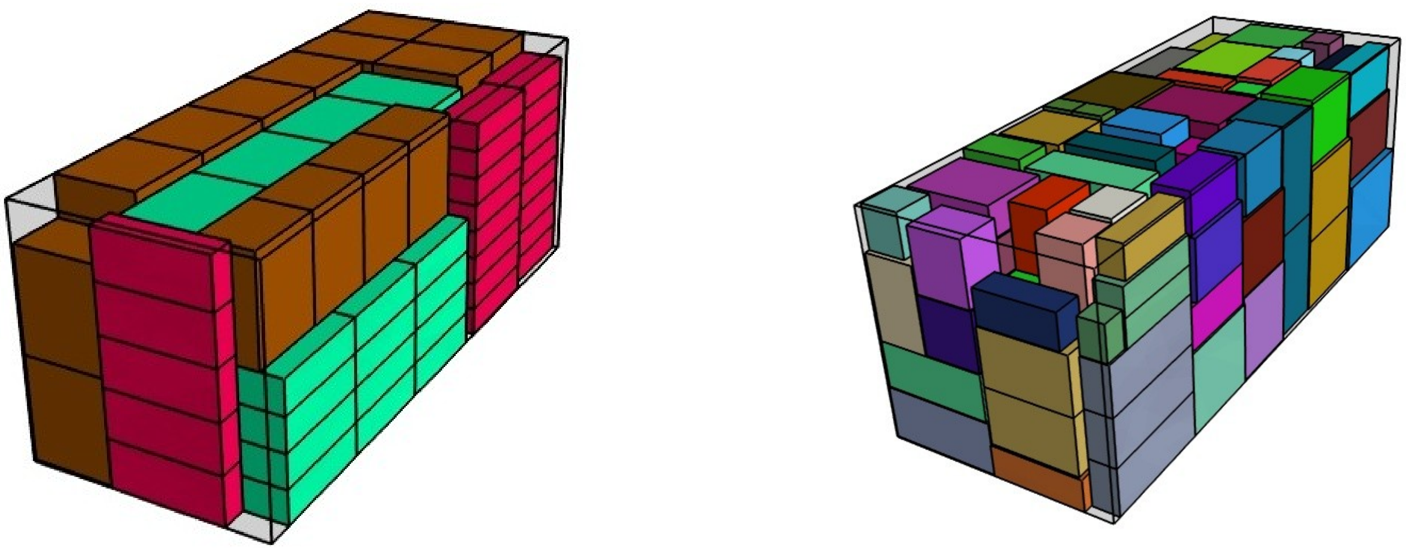
29 types
97 rectangles

3D packing

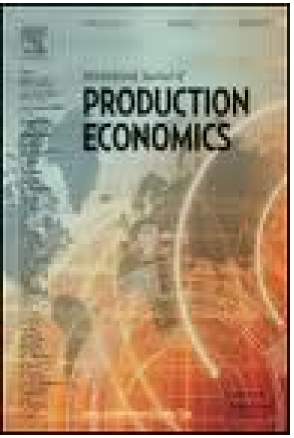
We have extended this to 3D packing:

J.F. Gonçalves and M.G.C.R., "A parallel multi-population biased random-key genetic algorithm for a container loading problem," Computers & Operations Research, vol. 29, pp. 179-190, 2012.

Tech report: http://www.research.att.com/~mgcr/doc/brkga-pack3d.pdf



3D bin packing

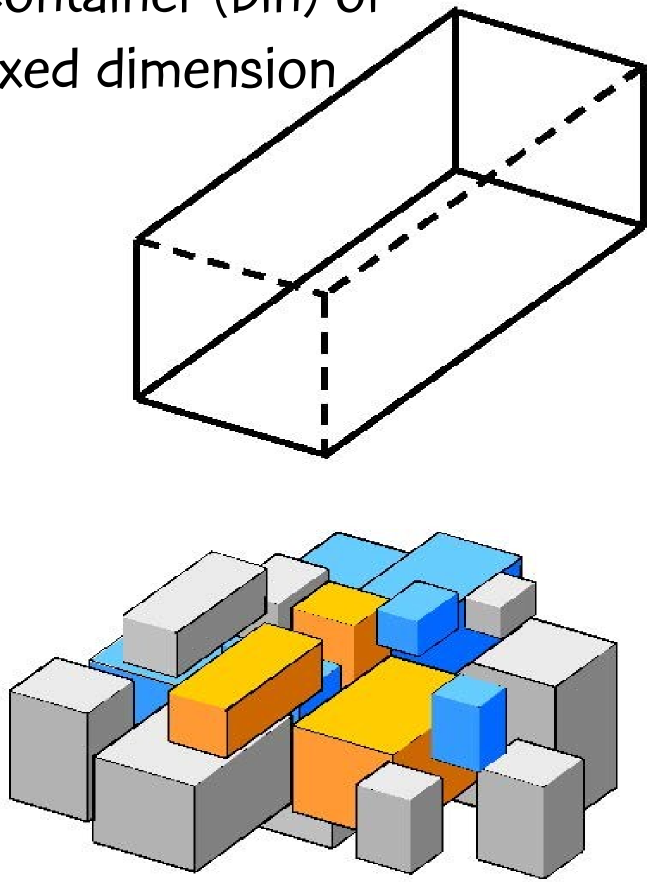


J.F. Gonçalves and M.G.C.R., "A biased random-key genetic algorithm for 2D and 3D bin packing problems," International J. of Production Economics, vol. 15, pp. 500–510, 2013.

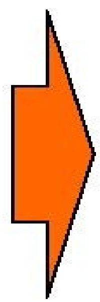
http://www.research.att.com/~mgcr/doc/brkga-binpacking.pdf

3D bin packing problem

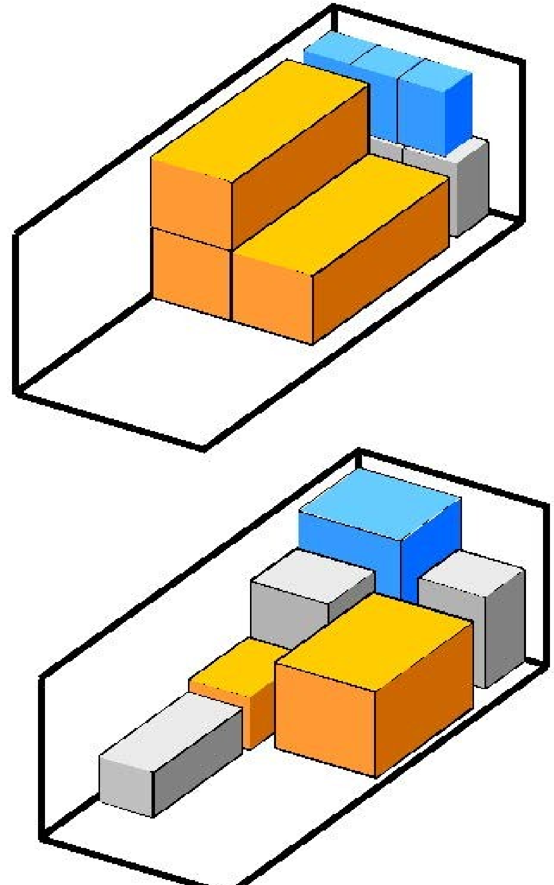
Container (bin) of fixed dimension



Boxes of different dimensions



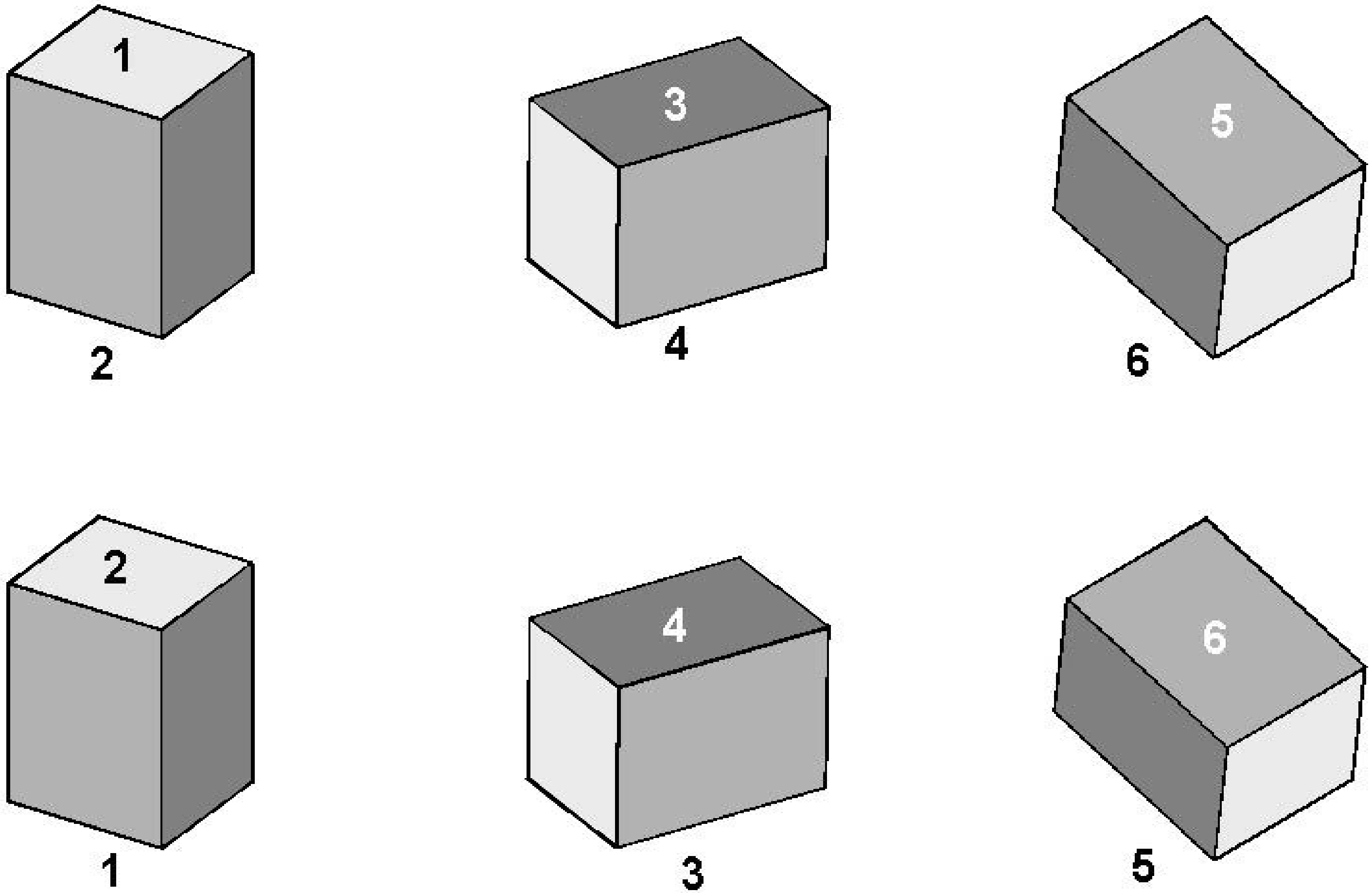
Minimize number of containers (bins) needed to pack all boxes



3D bin packing constraints

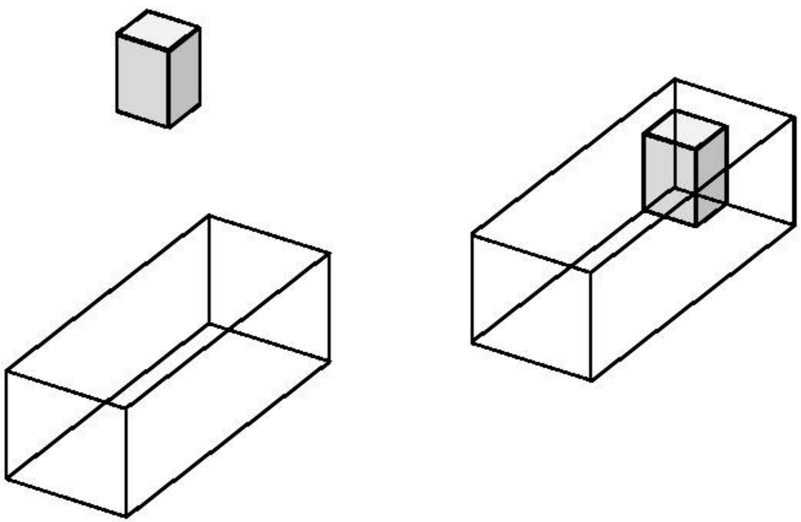
- Each box is placed completely within container
- Boxes do not overlap with each other
- Each box is placed parallel to the side walls of bin
- In some instances, only certain box orientations are allowed (there are at most six possible orientations)

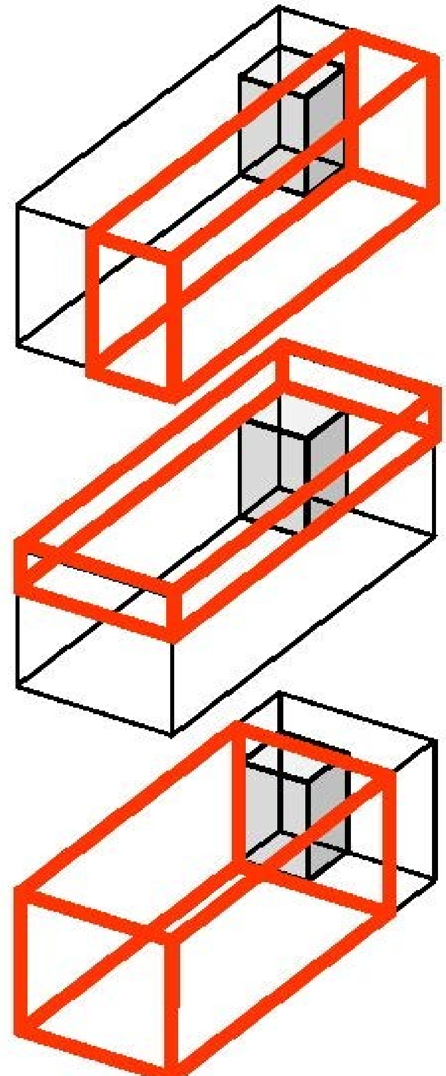
Six possible orientations for each box



Difference process - DP

(Lai & Chan, 1997)





When box is placed in container ...

use DP to keep track of maximal free spaces

Encoding

Solutions are encoded as vectors of 3n random keys, where n is the number of boxed to be packed.

$$X = (\underbrace{x_1, x_2, \ldots, x_n}_{\text{Box packing sequence}}, \underbrace{x_{n+1}, x_{n+2}, \ldots, x_{2n}}_{\text{Placement heuristic}}, \underbrace{x_{2n+1}, x_{2n+2}, \ldots, x_{3n}}_{\text{Box orientation}})$$

Decoding

1) Sort first n keys of X to produce sequence boxes will be packed;
2) Use second n keys of X to determine which placement heuristic to use (back-bottom-left or back-left-bottom):
   - if $x_{n+i} < ½$ then use back-bottom-left to pack i-th box
   - if $x_{n+i} \geq ½$ then use back-left-bottom to pack i-th box
3) Use third n keys of X to determine which of six orientations to use when packing box:
   - $x_{2n+i} \in [0,1/6)$: orientation 1;
   - $x_{2n+i} \in [1/6,2/6)$: orientation 2; ...
   - $x_{2n+i} \in [5/6,1]$: orientation 6.

Decoding

For each box

- scan containers in order they were opened
- use placement heuristic to place box in first container in which box fits with its specified orientation
- if box does not fit in any open container, open new container and place box using placement heuristic with its specified orientation

Fitness function

Instead of using as fitness measure the number of bins (NB)

- use adjusted fitness: aNB
- aNB = NB + (LeastLoad / BinVolume), where
  - LeastLoad is load on least loaded bin
  - BinVolume is volume of bin: H x W x L

Experiment

- Instances:
  - 320 instances of Martello *et al.* (2000)
  - generator is available at http://www.diku.dk/~pisinger/codes/html
  - 8 classes
  - 40 instances per class
  - 10 instances for each value of n ∈ {50, 100, 150, 200)

Experiment

- We compare BRKGA with:
  - TS3, the tabu search of Lodi et al. (2002)
  - GLS, the guided local search of Faroe et al. (2003)
  - TS2PACK, the tabu search of Crainic et al. (2009)
  - GRASP, the greedy randomized adaptive search procedure of Parreno et al. (2010)

Summary

Average number of bin in each class of 40 instances

Class	Bin size	BRKGA	GRASP	TS3	TS2PACK	GLS
1	100^3	**127.3**	**127.3**	127.9	128.2	128.3
2	100^3	**125.5**	125.8	126.8		
3	100^3	**126.5**	126.9	127.5		
4	100^3	294.0	294.0	294.0	**293.9**	294.2
5	100^3	**70.4**	70.5	71.4	71.0	70.8
6	10^3	**95.0**	95.4	96.1	95.8	96.0
7	40^3	**58.2**	59.4	60.0	59.0	59.0
8	100^3	**80.9**	82.0	82.6	81.9	81.9
Sum(rows 1, 4-8):		**725.8**	728.6	732.0	729.8	730.2
Sum(rows 1-8):		**977.8**	981.3	986.3		

Concluding remarks of technical part of talk

- Reviewed BRKGA framework
- Applied framework to
  - 2D/3D packing to maximize value packed
  - 2D/3D bin packing to minimize number of bins
- All decoders were simple heuristics
- BRKGA "learned" how to "operate" the heuristics
- In all cases, several new best known solutions were produced

My vision for the Department of ISE at UF

Department of Industrial & Systems Engineering at the U. of Florida

Internationally renowned center of excellence in operations research and industrial & systems engineering.

Focused on education and inter-disciplinary research on the big problems of the 21st century, including theoretical and applied issues in:

- Big data analytics
- Cloud computing
- Mobility
- Network science, including social networks
- Renewable energy
- Supply chain & manufacturing
- Health care
- Telecommunications
- Transportation

Raise department's U.S. News & World Report ranking

Raise ranking from 13th (tied with Columbia & NC State) to a top-10 (on a par with Cornell, Wisconsin, VA Tech & Purdue).

A higher ranking has the potential to result in a number of favorable outcomes:

- Attract more top-notch students
- Attract star faculty, both junior & senior
- Increase research production & funding
- Increased geographical diversity as a consequence of increase in number of
  - out-of-state undergraduate applicants
  - domestic graduate applicants

Raise department's U.S. News & World Report ranking

Raise ranking from 13th (tied with Columbia & NC State) to a top-10 (on a par with Cornell, Wisconsin, VA Tech & Purdue).

A higher ranking has the potential to result in a number of favorable outcomes:

- Attract more top-notch students
- Attract star faculty, both junior & senior
- Increase research production & funding
- Increased geographical diversity as a consequence of increase in number of
  - out-of-state undergraduate applicants
  - domestic graduate applicants

Raise department's U.S. News & World Report ranking

To achieve this ambitious goal, we envision a multi-faceted plan, including

- Undergraduate education
- Graduate education
- Research
- Administration
- Alumni relations
- Fund raising

Undergraduate education

Goal: Expose undergrads to an intellectually-rich environment so that upon graduation they best serve Florida, the Nation & the world.

The department will provide rigorous coursework and opportunities for research and practical experience.

Outcome: Graduates will be placed in industry & business where they can design, implement & manage complex systems.

Those wishing to pursue post-graduate education will have the needed skills to work toward graduate degrees in engineering, computer science, business, or applied mathematics.

Undergraduate students should be exposed ...

... not only to fundamental concepts in:

- Science
- Mathematics
- Computer science and information technology
- Economics
- Statistics
- Industrial and systems engineering
- Operations research

... but also to the new disciplines of:

- Data science
- Machine learning
- Network science

Undergraduate students should be exposed ...

... not only to fundamental concepts in:

- Science
- Mathematics
- Computer science and information technology
- Economics
- Statistics
- Industrial and systems engineering
- Operations research

... but also to the new disciplines of:

- Data science
- Machine learning
- Network science

They should learn how to apply these concepts to make data-driven decisions, carry out theoretical and empirical analyses, and manage complex systems in industry and government.

Graduate education

Goal: Prepare students for lifetime careers in academics, industry, or government.

Success of a graduate program depends heavily on

- Number
- Quality

of its student body.

Grad students have a significant impact on volume and quality of department's research production

- Directly, though student-faculty collaboration
- Indirectly, by assisting faculty and lessoning burden imposed on faculty

Outcome: Place graduates in industry, government, or university for careers in

- research
- teaching
- high-level technical position

Research

Goal: Balance theory and applications and encourage risk taking.

Labs enable research to focus on specific area, e.g.

- CAO – fomented research in biomedicine and energy
- Information lab – could facilitate thrust in big data analytics
- Simulation lab – stimulate simulation optimization

Grad students have a significant impact on volume and quality of department's research production

- Directly, though student-faculty research collaboration
- Indirectly, by assisting faculty with courses and lessening burden on faculty

Outlet: Though scholarly publications should continue to be the main outlet for research produced in department, the pursuit of patents should be encouraged.

Administration

Some opportunities:

- Manage a web-based system to aid in advising
- Aid faculty in writing proposals
- Help faculty write initial invention disclosures for possible patent filings

Alumni relations

Good relations with alumni are important not only because of potential funding opportunities, but also to maintain a strong sense of community.

Some ideas:

- Keep database of alumni up-to-date
- Graduating classes should be made aware of importance of keeping in touch
- Annual newsletter (as simple as an email) could be sent out to inform and engage alumni
- Department reception at INFORMS and IIE meetings

Fund raising

Fund raising is an important responsibility of a department chair. A constant effort must be sustained to seek new sources of funding.

Some potential sources:

- Alumni & non-alumni donations
- Grants from research funding agencies
- Grants from industrial partners
- Patent licensing
- Externally funded graduate students

What to do?

Hire 5 – 6 new faculty in four critical areas:

- data sciences
- machine learning
- stochastic processes
- simulation

What to do?

Recruit top-notch graduate students:

- From within the U.S.
- From abroad
- Increase availability of
  - Fellowships
  - Teaching assistantships
  - Research assistantships

What to do?

Increase alliances with industrial partners:

- Internships
- Research collaborations
- Expose students to real-world problems
- Harvest data for data science research

What to do?

Update computer infrastructure to support data science research:

- Within department
- In partnerships with other departments
- Commercial cloud computing

What to do?

Encourage graduate students to collaborate with faculty in the preparation and filing of patents:

- Potential source of funding
- Acquire critical skill

What to do?

Undergraduate students should have the opportunity to carry out independent research during their senior year:

- Senior thesis
- Introduce student to tasks involved in research: literature search, problem statement, empirical studies, and writing and defending the thesis
- Faculty member should supervise student

What to do?

Students should graduate with the programming skills needed for the technical jobs of the 21st century:

- Many traditional ISE jobs are not going to ISE graduates
- CS majors (software engineers) are taking many of these jobs

What to do?

Increase diversity through outreach and targeted fellowships:

- Recruit women and minorities to increase gender and ethnic diversity
- Recruit out-of-state & international undergraduate students and domestic graduate students to increase geographical diversity

Thanks!