High Multiplicity Strip Packing

Andrew Bloch-Hansen

Supervisor: Professor Solis-Oba
Western University, London, Ontario, Canada
September 9, 2019

Outline

Outline

Analysis of Algorithms

Algorithm Design

- The algorithm must always produce the correct answer
- The algorithm must not get stuck in an infinite loop

Algorithm Analysis

- How much time is needed to run the algorithm? (time complexity: "big-oh")
- How much storage is needed for the algorithm? (space complexity)

Algorithm 1 findLargestNumber(int[] A)

```
1: Input: The array A.
2: Output: The largest number in A.
3: int largest = A[0]
4: for int i = 1 to n do
5:     if A[i] > largest then
6:         largest = A[i]
7:     end if
8: end for
9: return largest
```

Optimization Problems

- Goal: find best solution among all possible solutions
  - Minimization problems
  - Maximization problems
- Many known algorithms for these problems take too long

Approximation Ratio

- Let $OPT(I)$ be an optimal solution for a problem on input I, and let $SOL(I)$ be a solution output by an approximation algorithm for the same input I
  - For minimization problems, *approximation ratio* $= \frac{SOL(I)}{OPT(I)}$
  - For maximization problems, *approximation ratio* $= \frac{OPT(I)}{SOL(I)}$

Packing Problems

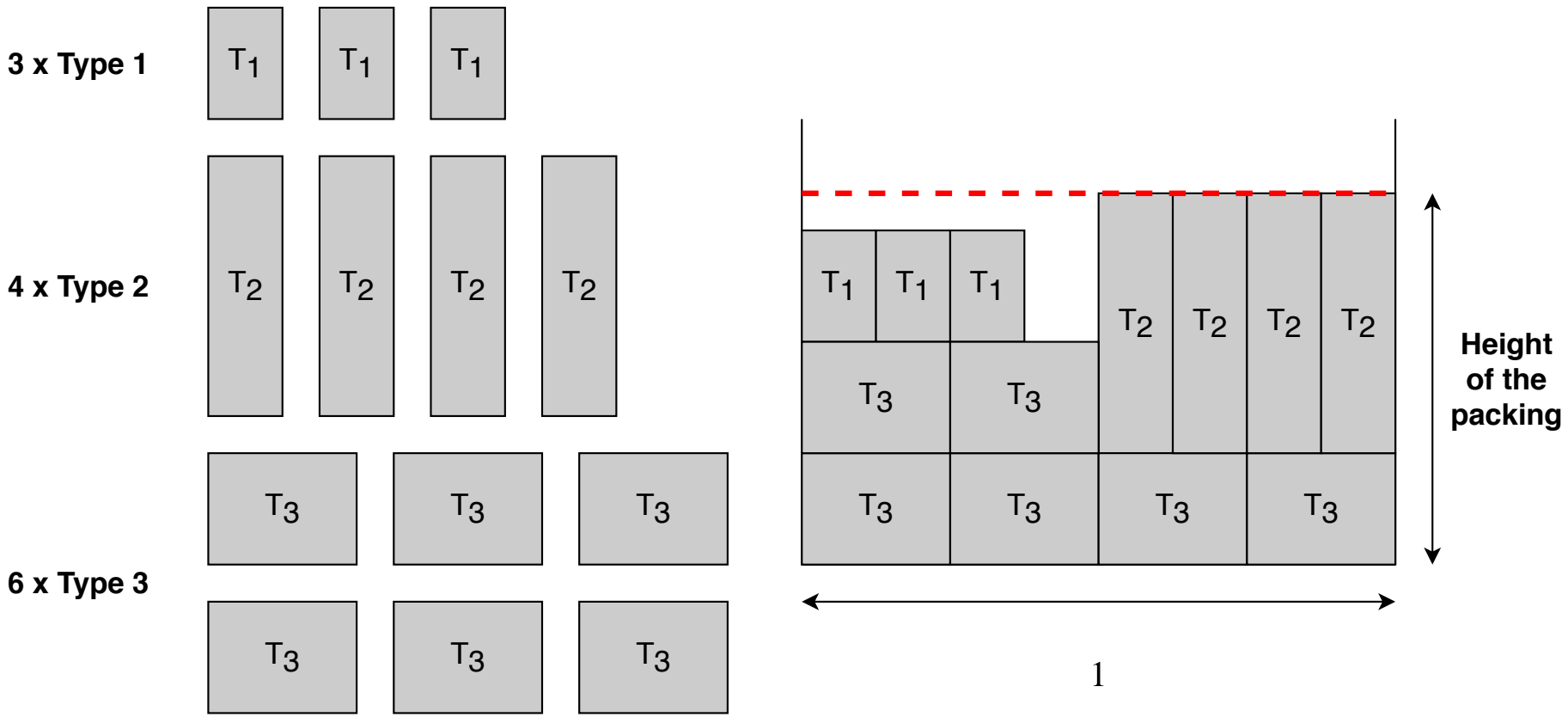
- Maximize the number of objects that can be packed in a container
- Minimize the number of containers needed to hold a set of objects
- Constraints:
  - Rotations
  - Overlaps
  - Shapes
  - Cost
  - Fixed-length objects
  - Fixed number of containers
  - Fixed dimensions of container
  - Group inputs into categories

Applications

Applications

Applications

High Multiplicity Strip Packing

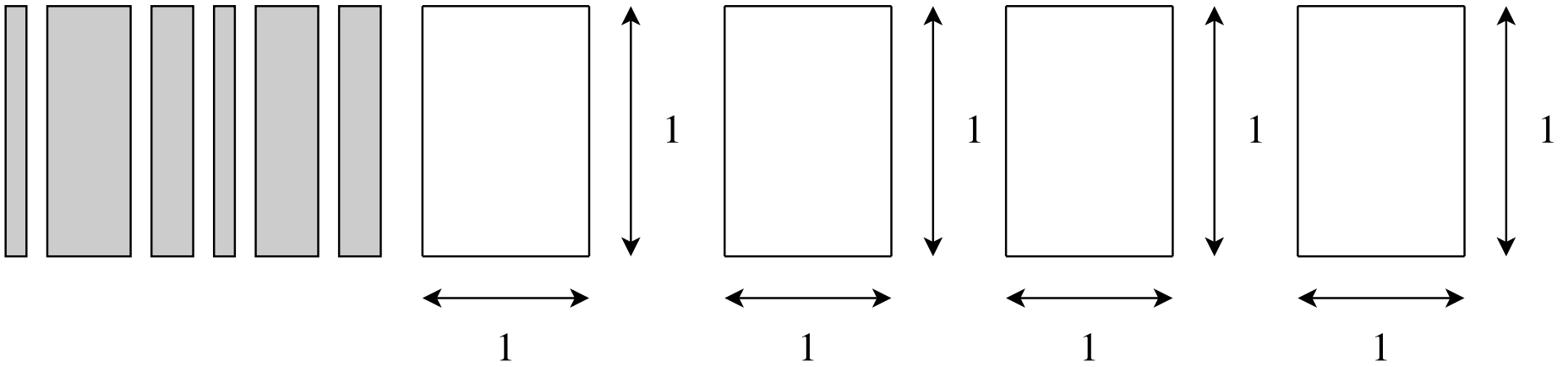


Problem Definition

- Given k distinct rectangle types T_i, where each type T_i has n_i rectangles of width $0 < w_i \leq 1$ and height $0 < h_i \leq 1$, the goal is to pack them into a strip of width 1 and minimum height, without rotations or overlaps

Outline

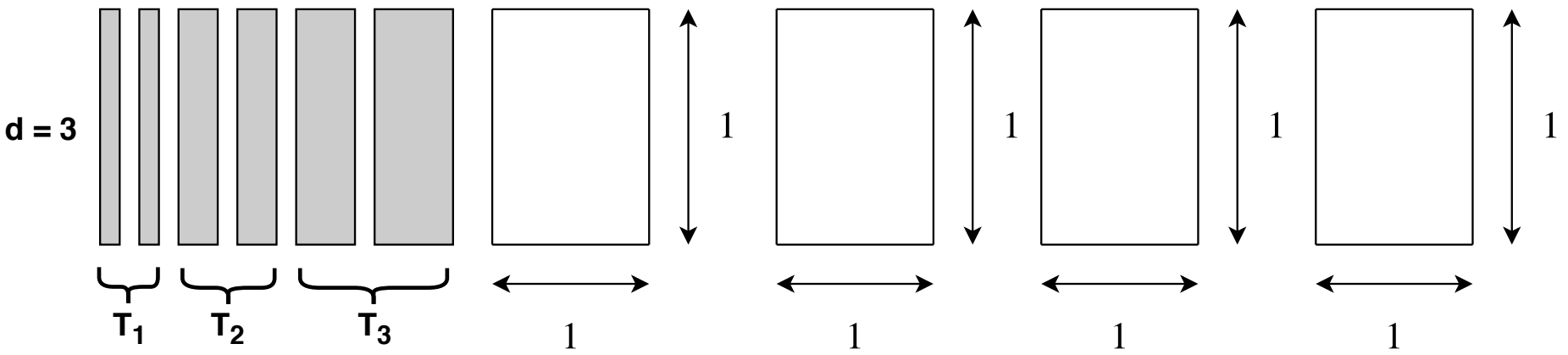
Bin Packing



Related Work

- Johnson et al.: $SOL(I) \leq \frac{3}{2}OPT(I)$
- Dósa: $SOL(I) \leq \frac{11}{9}OPT(I) + \frac{6}{9}$
- Yao: $SOL(I) \leq \frac{11}{9}OPT(I)$
- Garey and Johnson: $SOL(I) \leq \frac{71}{60}OPT(I)$
- Fernandez de la Vega and Lueker: $SOL(I) \leq (1+\epsilon)OPT(I)$
- Rothvoß: $OPT(I) + O(log\, OPT(I) * log\, log\, OPT(I))$

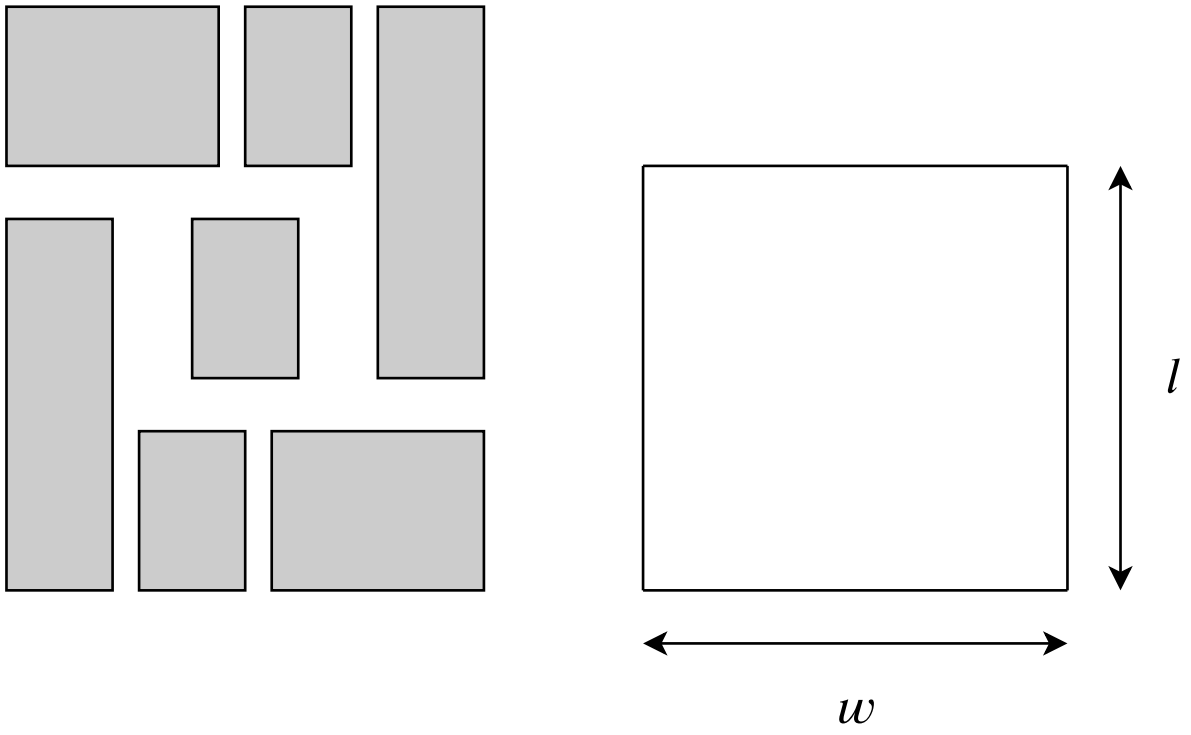
Cutting Stock



Related Work

- Filippi and Agnetis: $SOL(I) \leq OPT(I) + (d-2)$
- Filippi and Agnetis $(2 < d \leq 6)$: $SOL(I) \leq OPT(I) + 1$
- Filippi and Agnetis $(d > 6)$: $SOL(I) \leq OPT(I) + 1 + \lfloor \frac{d-1}{3} \rfloor$
- Jansen and Solis-Oba: $SOL(I) \leq OPT(I) + 1$
- Goemans and Rothvoß: $SOL(I) = OPT(I)$
- Karmarkar and Karp: $SOL(I) \leq OPT(I) + O(log^2\, d)$
- Rothvoß: $SOL(I) \leq OPT(I) + O(log\, d * log\, log\, d)$

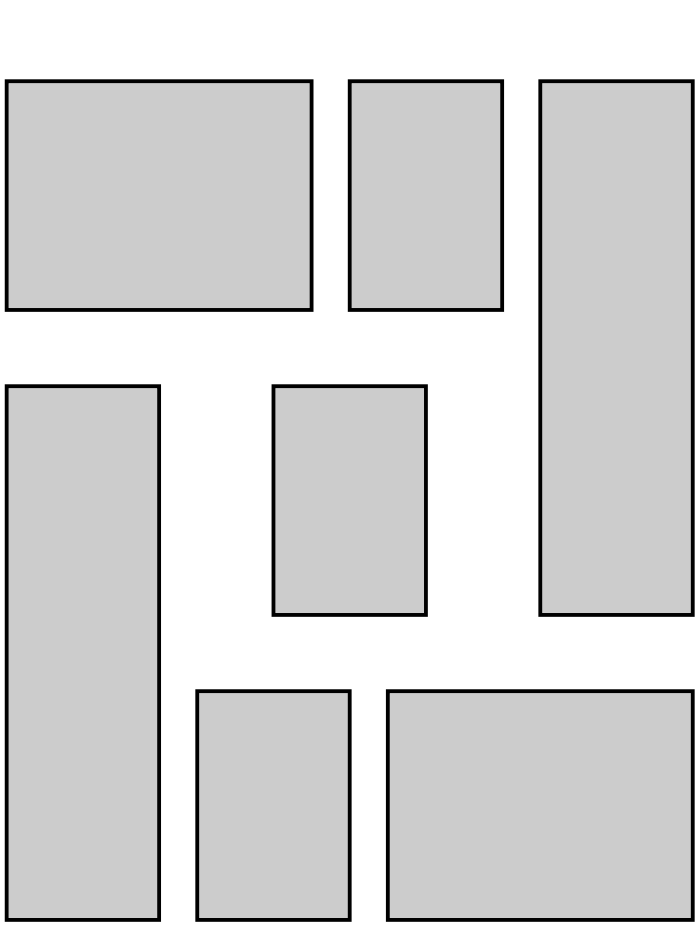
Rectangle Packing

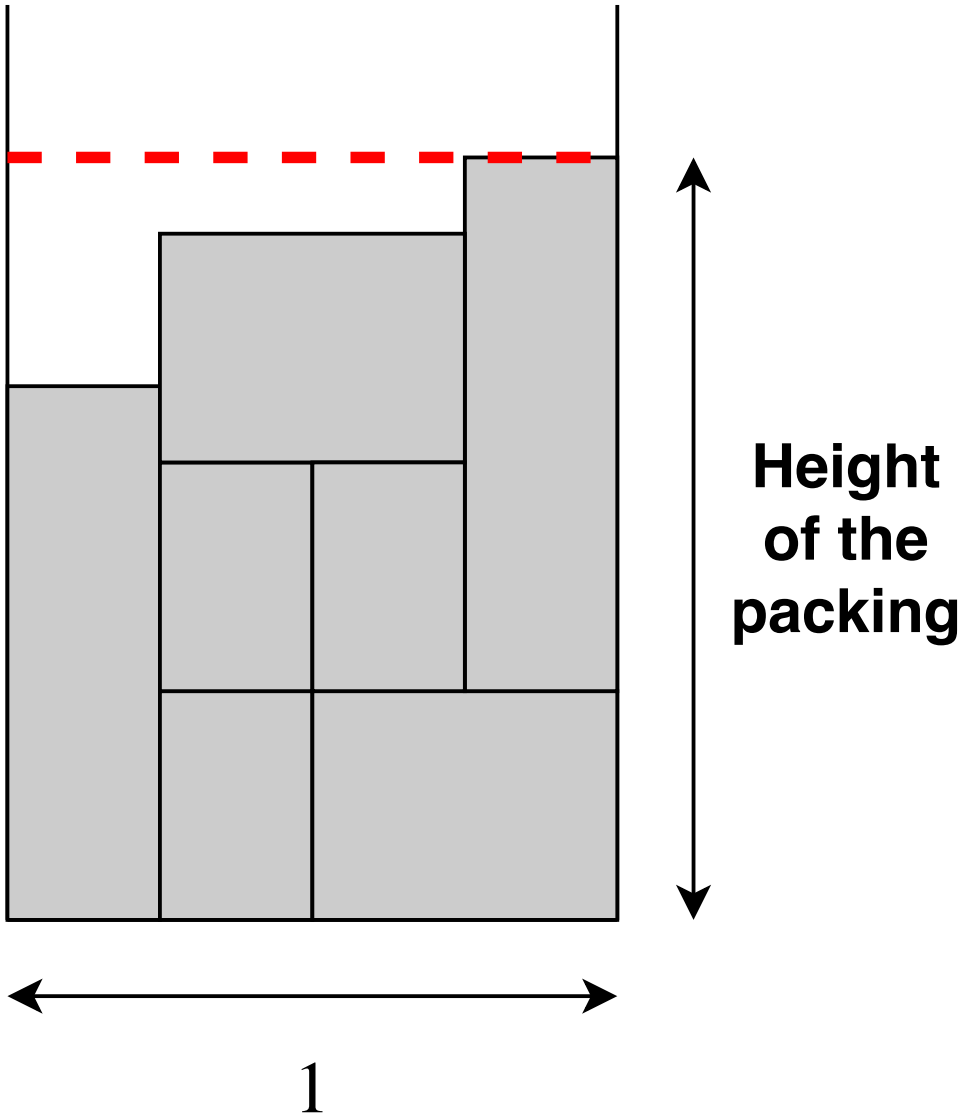


Related Work

- Baker et al. (unit-squares into rectangles): $SOL(I) \leq \frac{4}{3}OPT(I)$
- Caprara and Monaci (rectangles with profits): $SOL(I) \leq (3+\epsilon)OPT(I)$
- Jansen and Zhang: $SOL(I) \leq (2+\epsilon)OPT(I)$
- Harren (squares with profits): $SOL(I) \leq (\frac{5}{4}+\epsilon)OPT(I)$
- Jansen and Solis-Oba: $SOL(I) \leq (1+\epsilon)OPT(I)$
- Lan et al. (square without profits): $SOL(I) \leq (1+\epsilon)OPT(I)$

Strip Packing





Related Work

- Baker et al.: $SOL(I) \leq 3OPT(I)$
- Coffman et al.: $SOL(I) \leq 2.7OPT(I)$
- Sleator: $SOL(I) \leq 2.5OPT(I)$
- Schiermeyer: $SOL(I) \leq 2OPT(I)$
- Steinberg: $SOL(I) \leq 2OPT(I)$
- Harren and van Stee: $SOL(I) \leq 1.9396OPT(I)$
- Harren et al.: $SOL(I) \leq (\frac{5}{3} + \epsilon)OPT(I)$
- Golan: $SOL(I) \leq \frac{4}{3}OPT(I)$
- Baker et al: $SOL(I) \leq \frac{5}{4}OPT(I)$
- Kenyon and Rémila: $SOL(I) \leq (1 + \epsilon)OPT(I) + O(\frac{1}{\epsilon^2})$
- Jansen and Solis-Oba: $SOL(I) \leq (1 + \epsilon)OPT(I) + 1$

Outline

High Multiplicity Strip Packing

Motivation

- Can we design an algorithm for HMSPP that finds better solutions than algorithms for the strip packing problem?

High Multiplicity Strip Packing

Contributions

- We present an algorithm designed in collaboration with Yu for HMSPP when $k = 3$ for which $SOL(I) \leq OPT(I) + \frac{5}{3}$ and for any fixed value k $SOL(I) \leq OPT(I) + (k - \frac{4}{3})$.
- We present an algorithm for HMSPP when $k = 4$ for which $SOL(I) \leq OPT(I) + \frac{5}{2}$ and for any fixed value k $SOL(I) \leq OPT(I) + (k - \frac{3}{2})$

Linear Program

- A *linear program* is an encoding of a mathematical model, and can be expressed in the general form:

$$\begin{array}{ll} \text{maximize} & c^T x \\ \text{subject to} & Ax \leq b, \\ \text{and} & x \geq 0 \end{array}$$

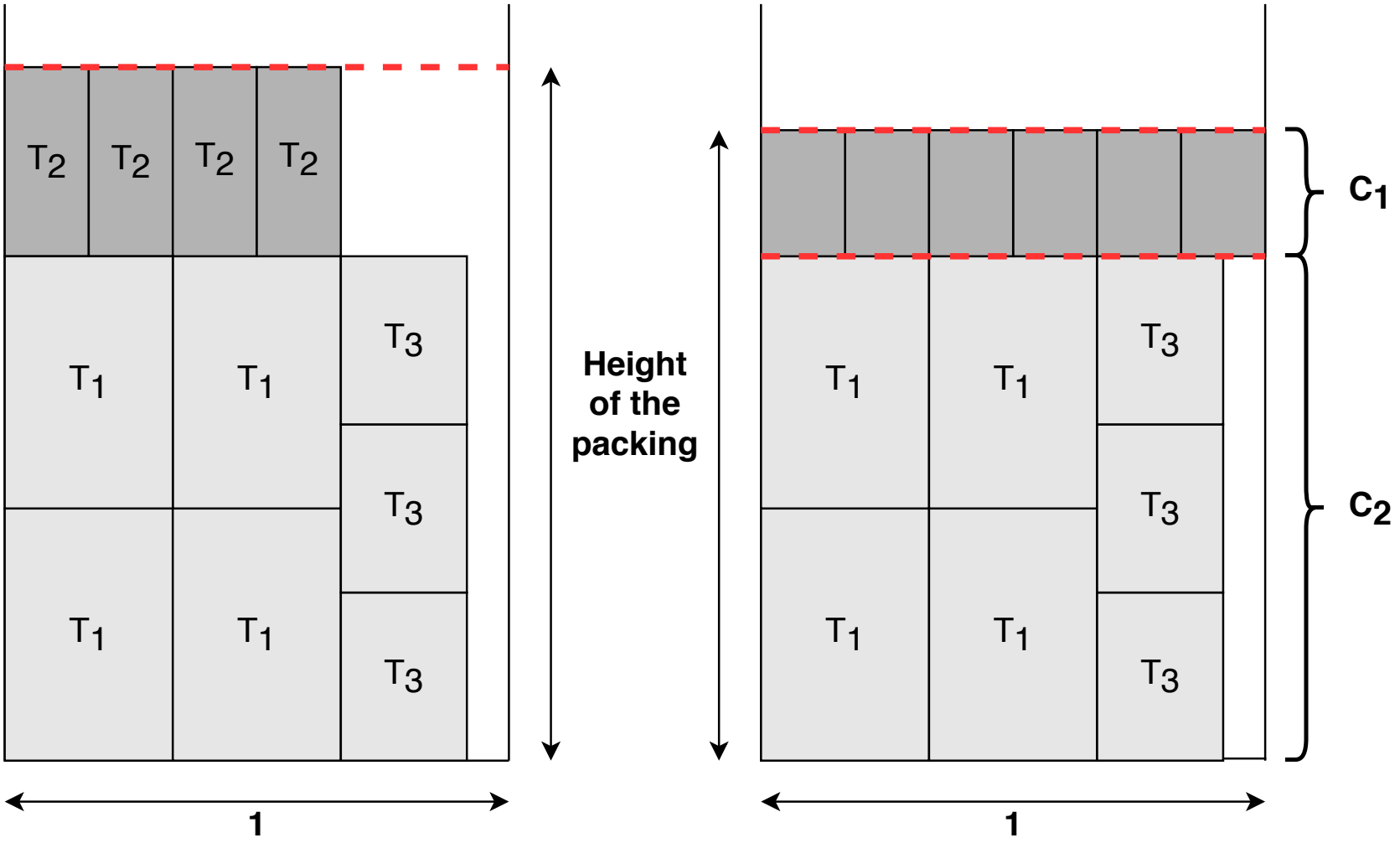
Integer Program

- An *integer program* is a linear program where all the variables are integers

Linear Program Rounding

1. Formulate a hard problem that we wish to solve as an integer program and then relax the integrality constraints to obtain a linear program
2. Solve the linear program
3. Transform the solution of the linear program into a feasible solution for the hard problem by rounding the values of non-integral variables to integer values

Fractional Strip Packing (FSPP)

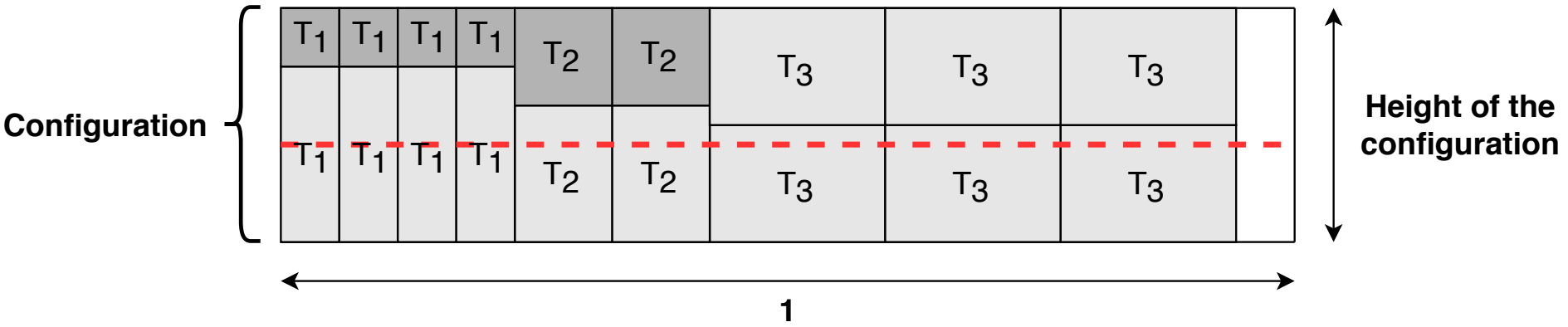


High Multiplicity Strip Packing Problem

Fractional Strip Packing Problem

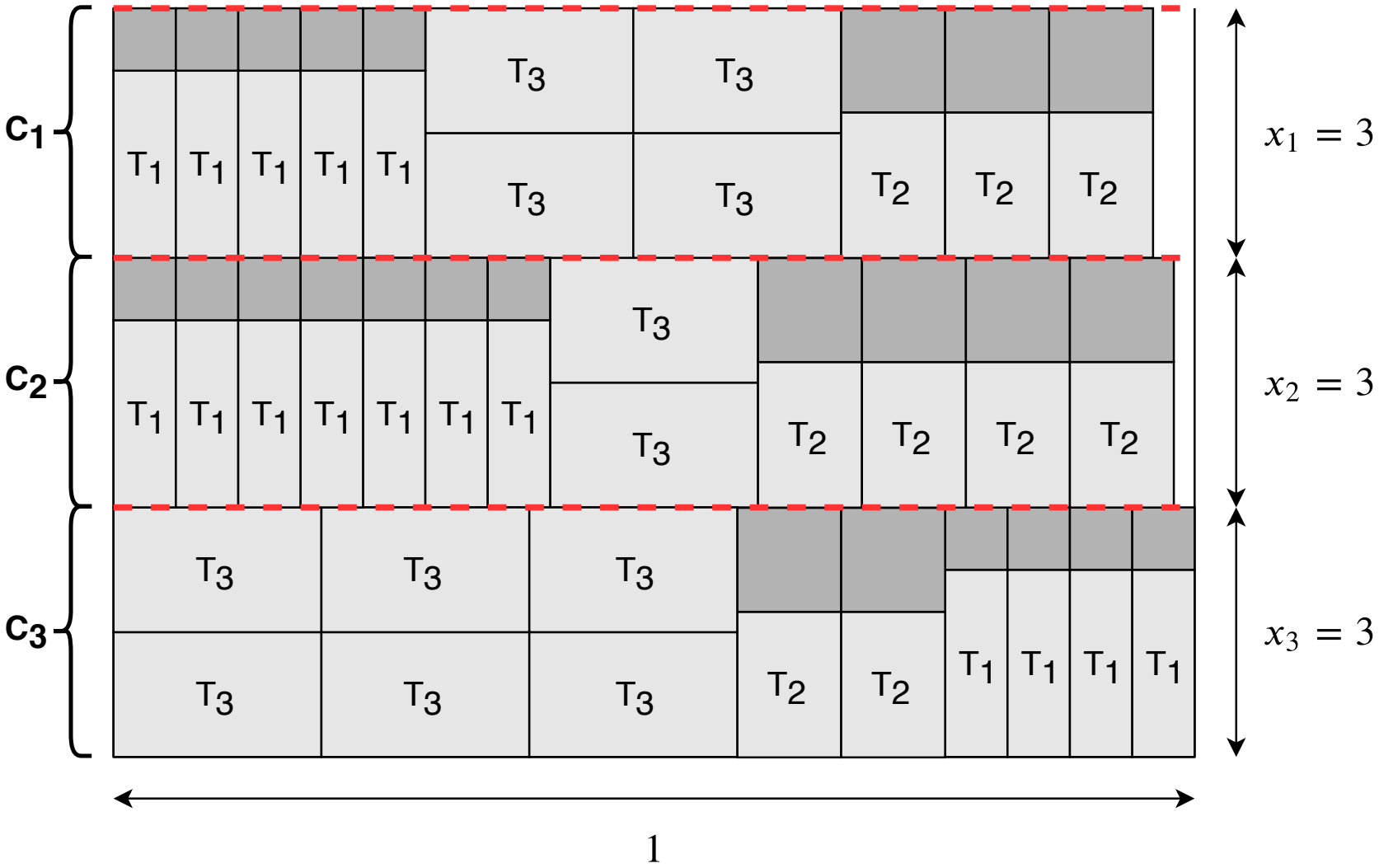
- Consider the version of the strip packing problem where rectangles can be sliced

Fractional Strip Packing



- A *configuration* is group of rectangles packed side-by-side of width at most 1

Fractional Strip Packing



- A fractional packing consists of a finite number of configurations

Fractional Strip Packing

Linear Program Formulation for Fractional Strip Packing

- Let x be a vector where x_j represents the height of configuration C_j
- Let n_i be the total number of rectangles of type T_i and let h_i be the height of each rectangle of type T_i.
- Let $n_{i,j}$ be the number of rectangles of type T_i in configuration C_j.

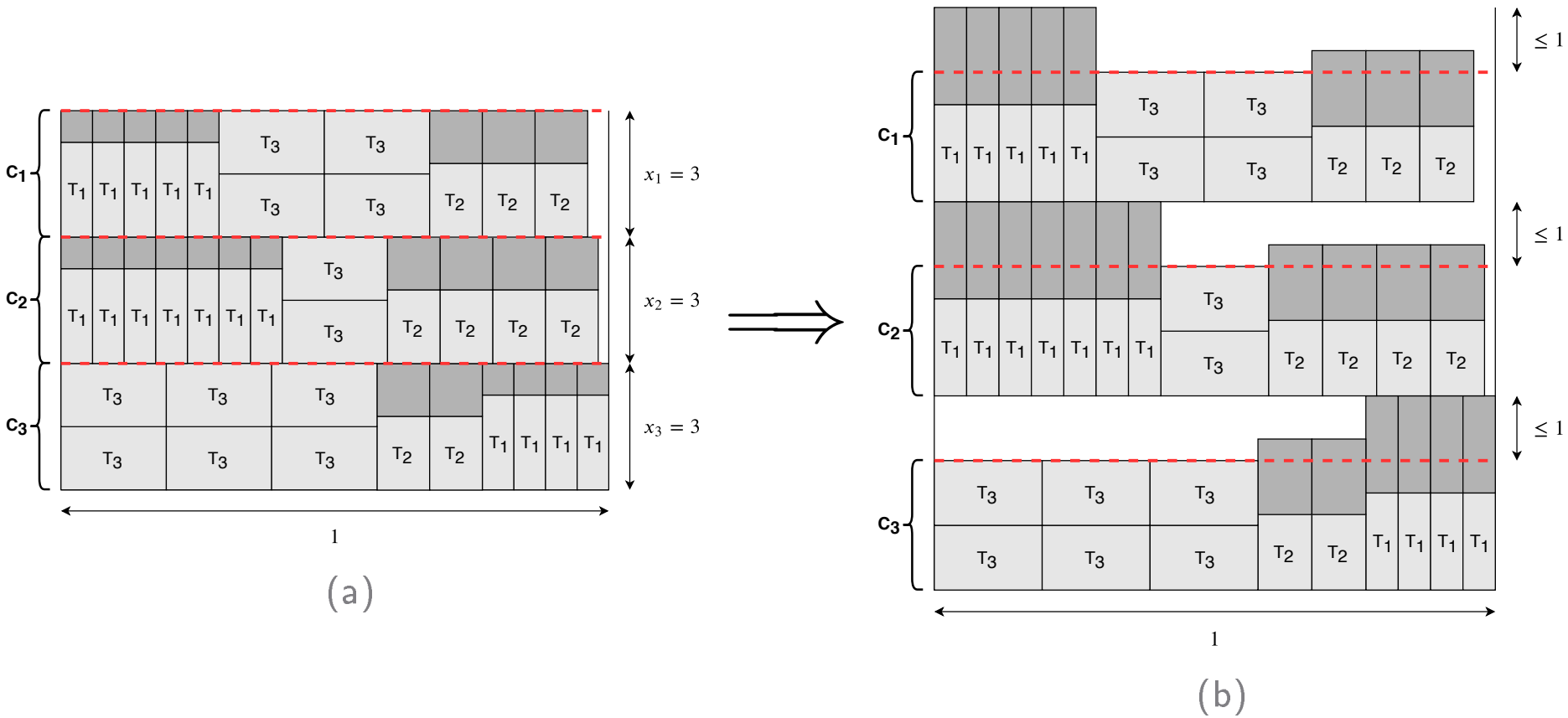
Linear Program for Fractional Strip Packing

$$\text{Minimize} \sum_{j \in J} x_j$$

$$\text{Subject to:} \sum_{j \in J} x_j n_{i,j} \geq n_i h_i, \text{ for each rectangle type } i$$

$$x_j \geq 0, \text{ for each } j \in J$$

A Simple Approximation Algorithm for HMSPP

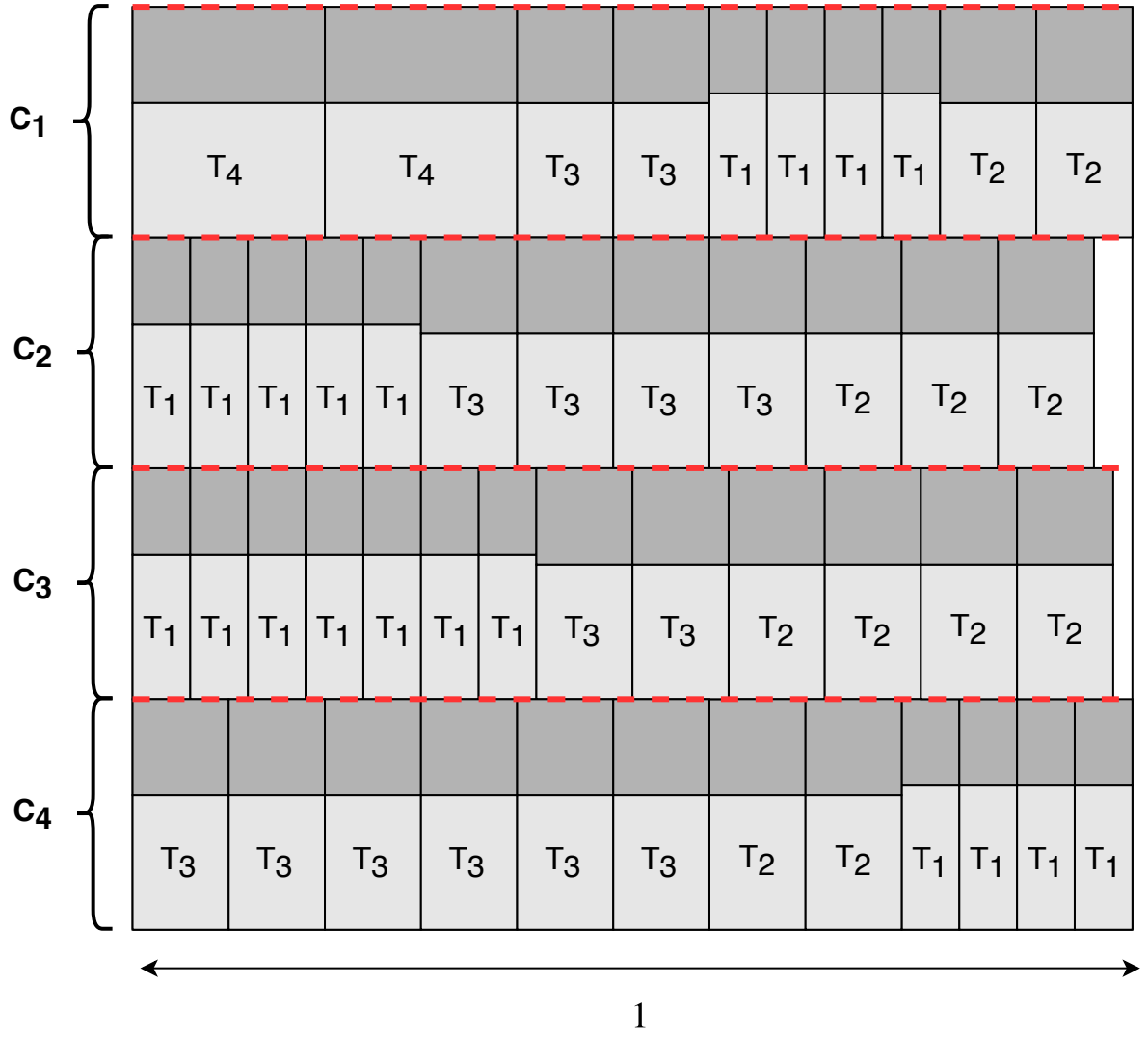


A Simple Approximation Algorithm for HMSPP

- Replace each fractional rectangle with a whole rectangle of its type

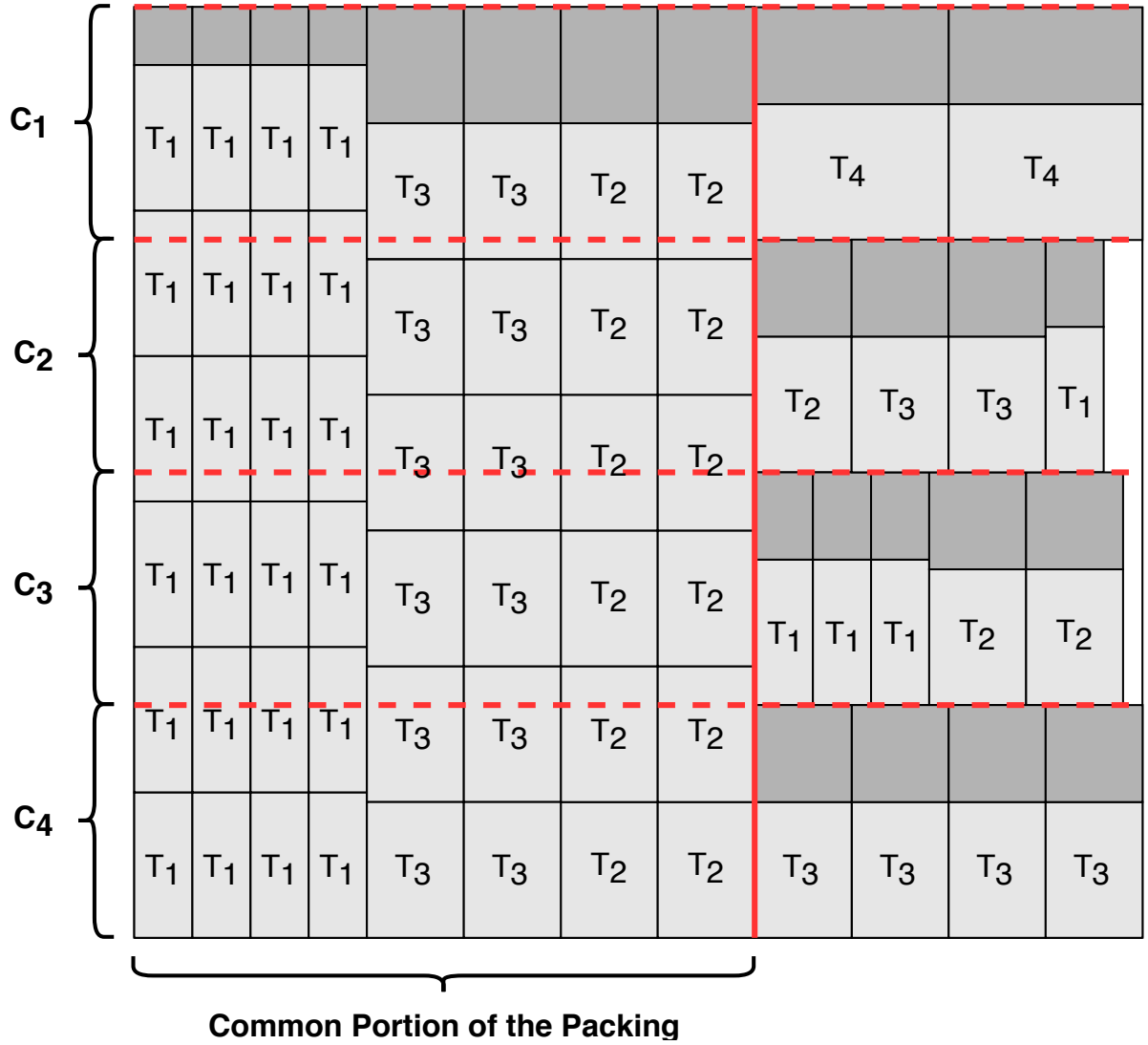
Outline

Four Configurations in the Solution of the Linear Program



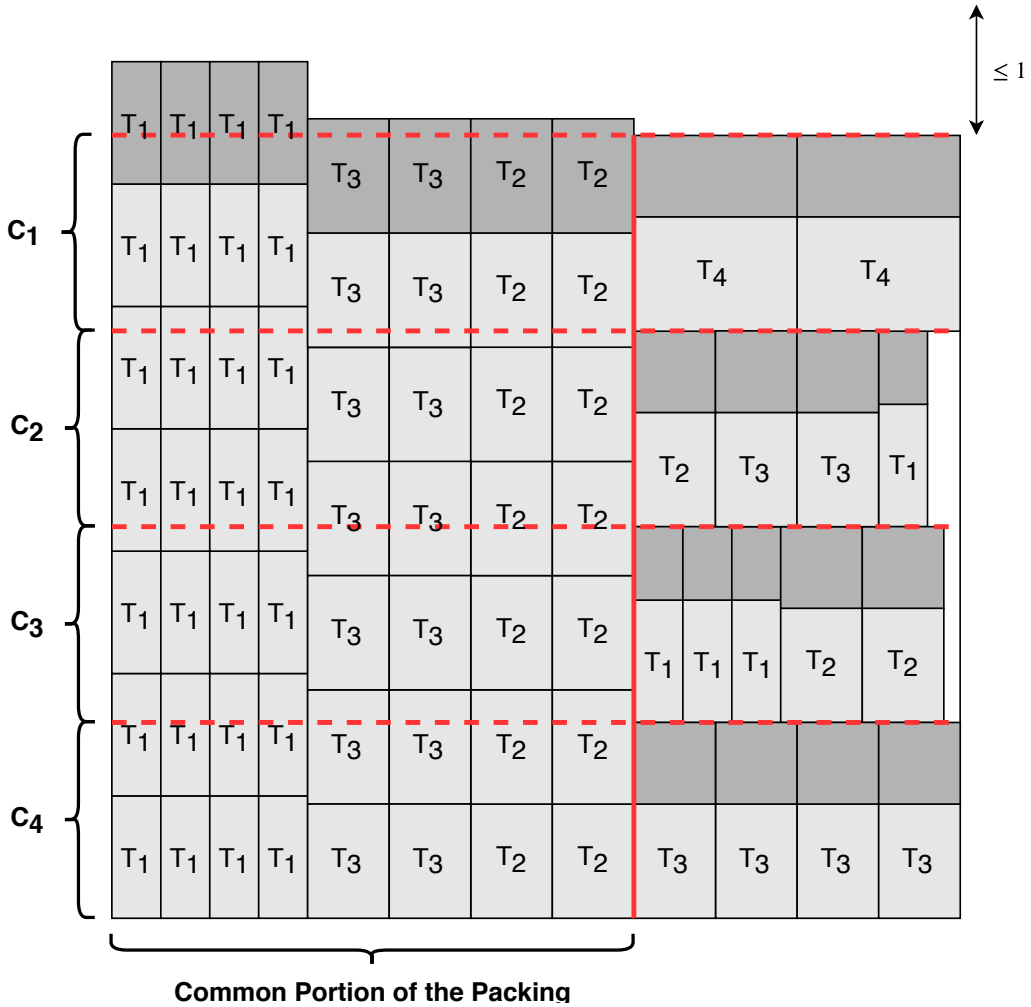
- The four configurations are packed one on top of the other

Common Portion of the Packing



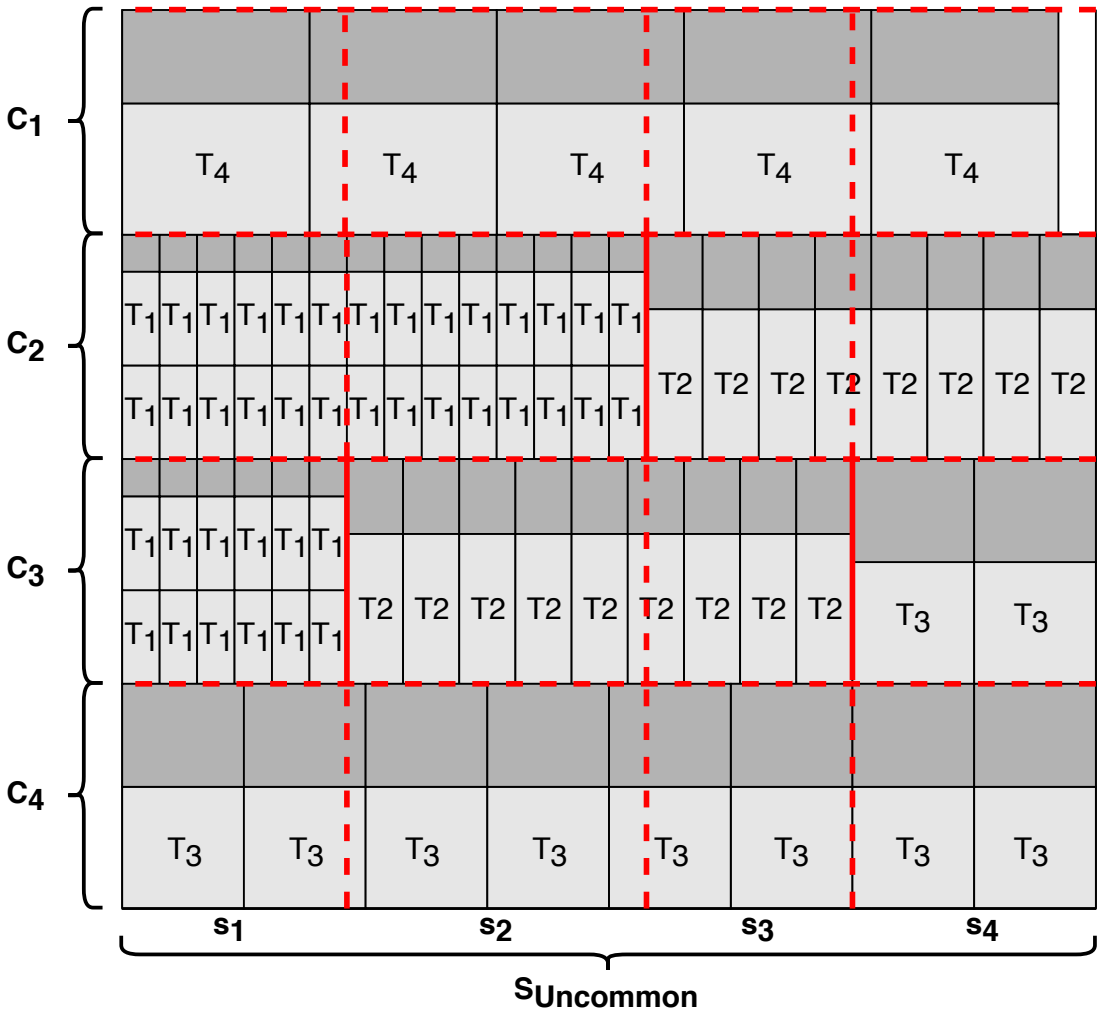
- Rectangles are arranged horizontally within the configurations

Rounding the Common Portion of the Packing



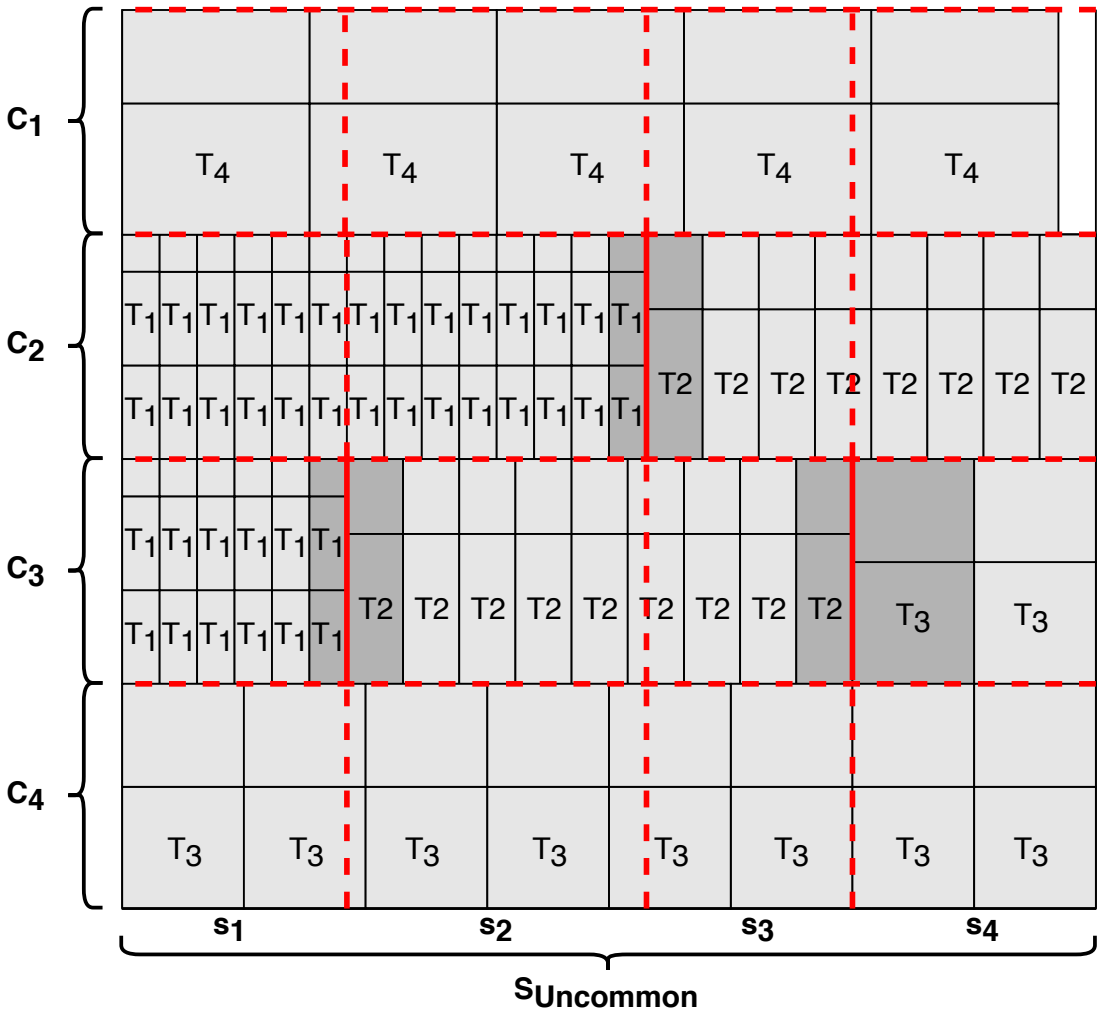
- Fractional rectangles in the common portion of the packing are *rounded up*

Sorting the Configurations in the Uncommon Portion



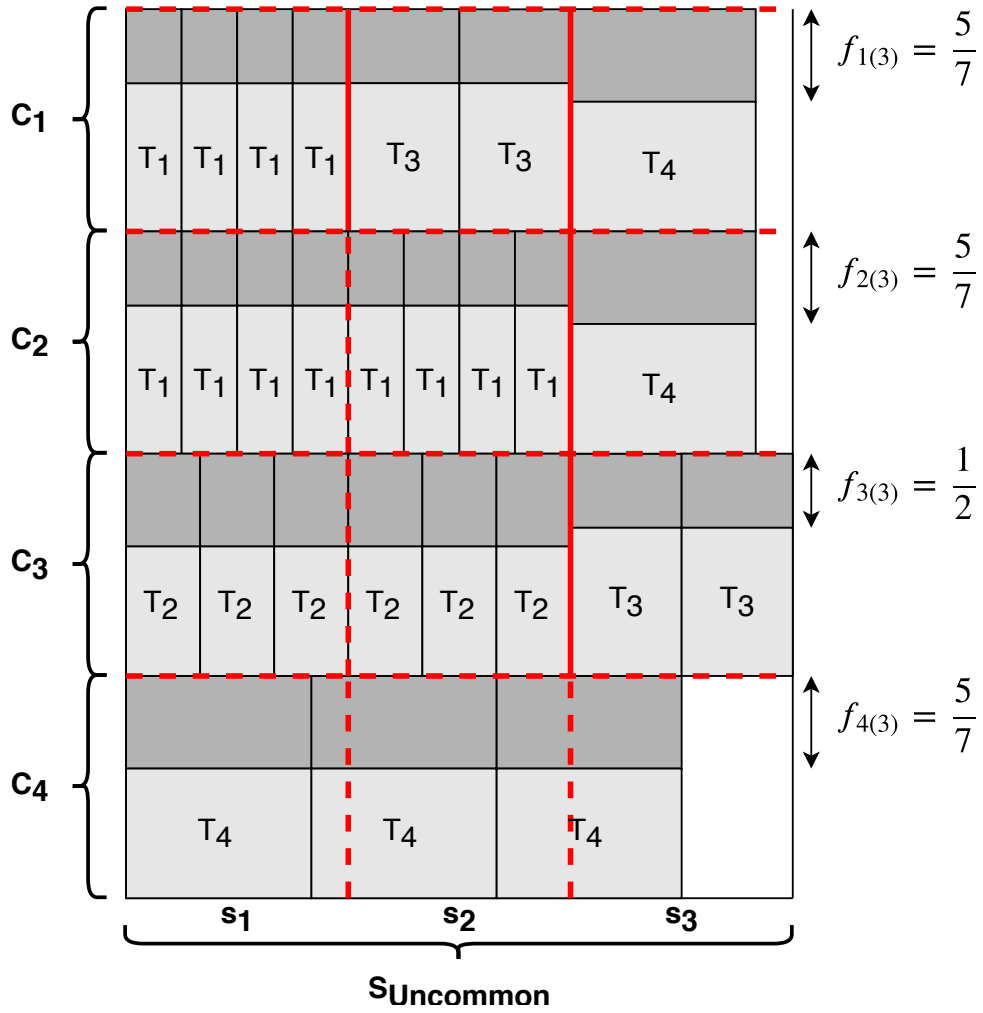
- Sort the rectangles within each configuration

Vertical Divisions in the Uncommon Portion



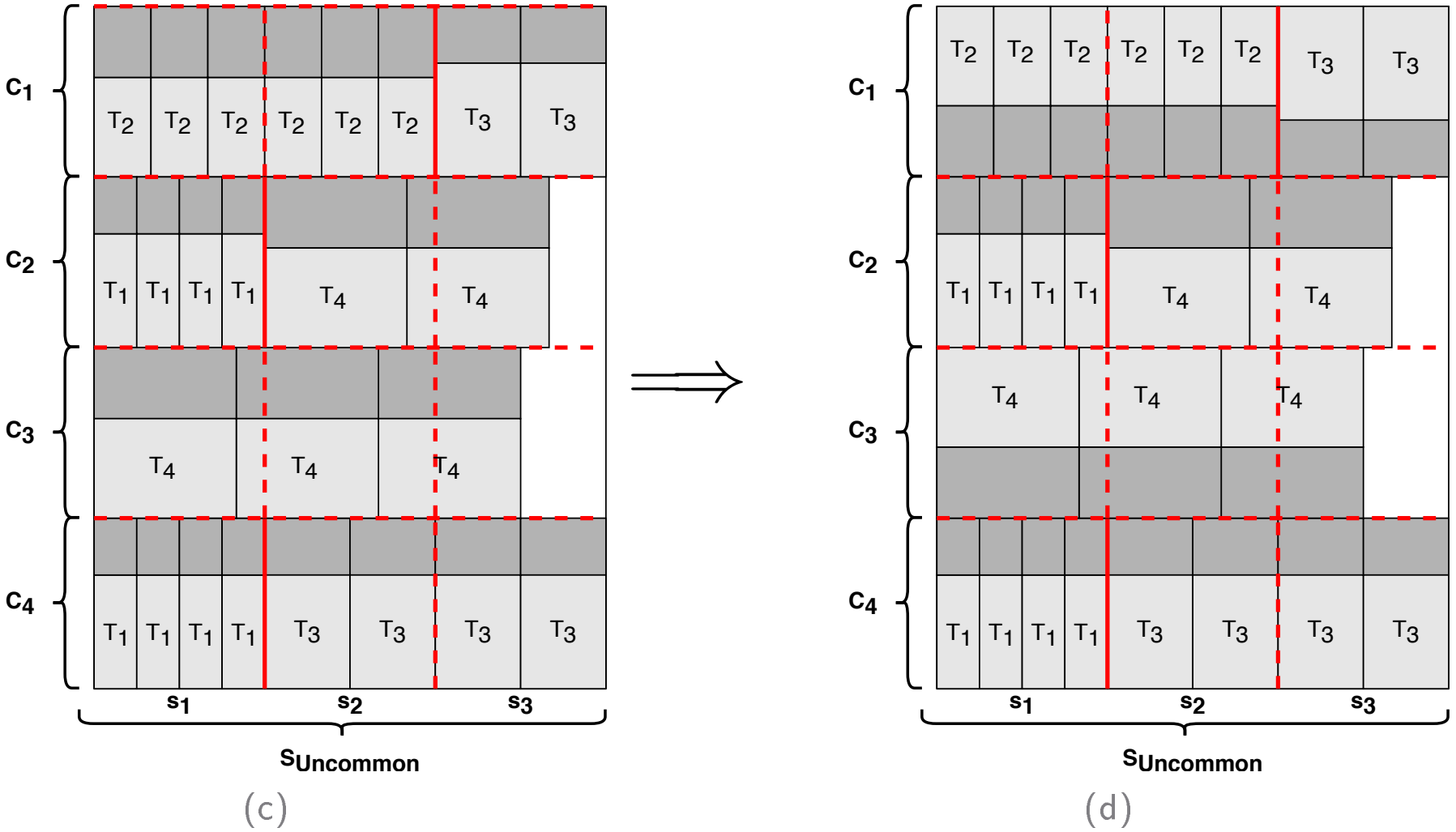
- A division is created between rectangles of different types within a configuration

Fractional Rectangles in a Vertical Section



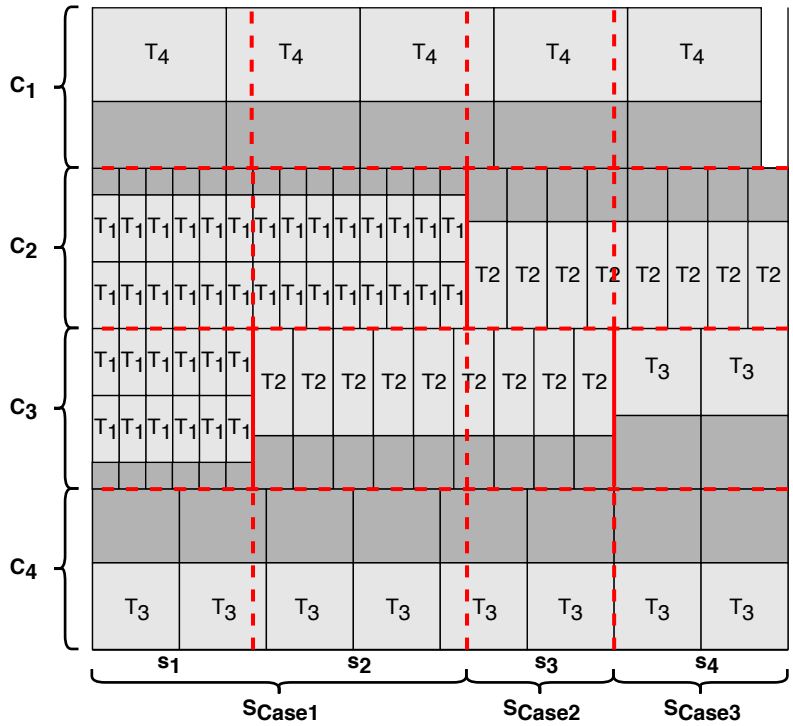
- Let $f_{1(i)}$, $f_{2(i)}$, $f_{3(i)}$, and $f_{4(i)}$ represent the fraction of the rectangles

The Solution to the Linear Program has Four Configurations



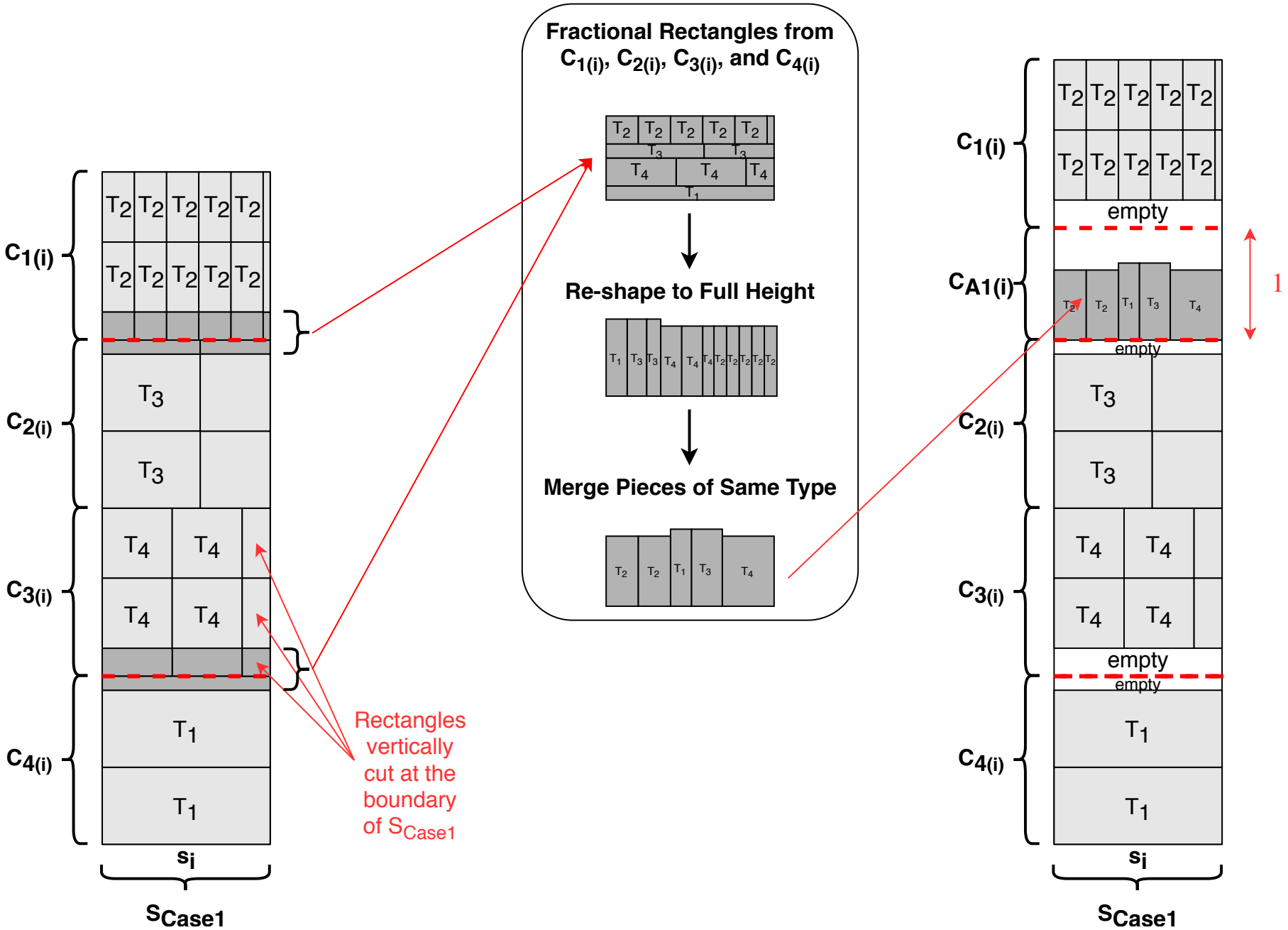
- C_1 and C_3 are flipped upside down

The Solution to the Linear Program has Four Configurations

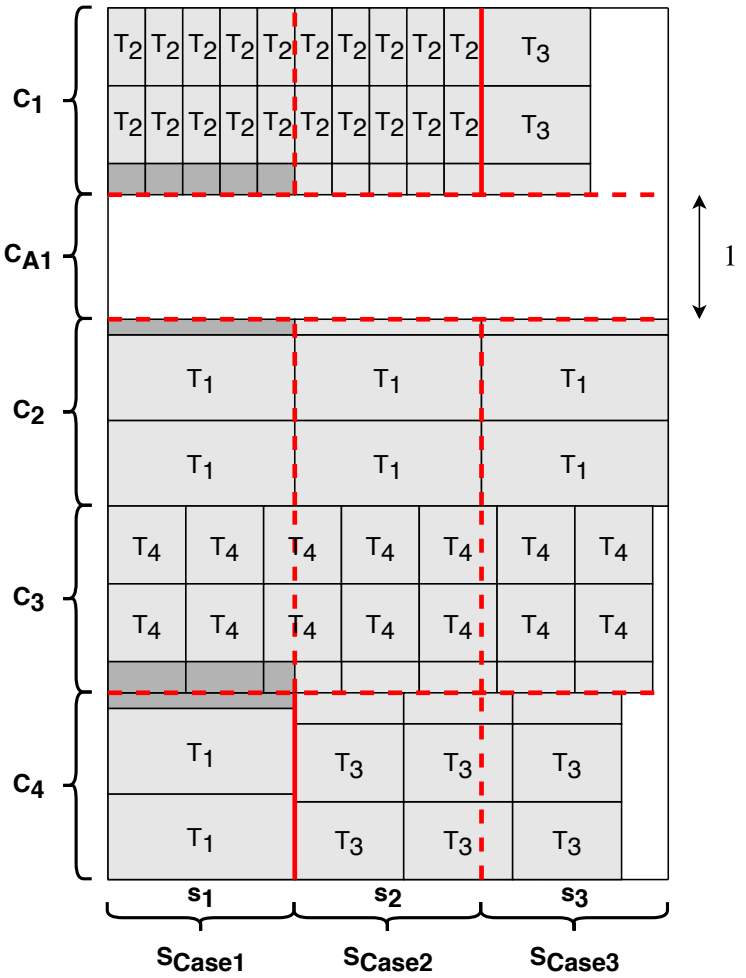


- Let i be the smallest index for which $f_{1(i)} + f_{2(i)} > \frac{1}{2}$ or $f_{3(i)} + f_{4(i)} > \frac{1}{2}$. Order the configurations so that $f_{3(j)} + f_{4(j)} > \frac{1}{2}$ for all $j \geq i$:
  - **Case 1:** $f_{1(i)} + f_{2(i)} \leq \frac{1}{2}$ and $f_{3(i)} + f_{4(i)} \leq \frac{1}{2}$,
  - **Case 2:** $f_{1(i)} + f_{2(i)} \leq 1$ and $f_{3(i)} + f_{4(i)} > \frac{1}{2}$, and
  - **Case 3:** $f_{1(i)} + f_{2(i)} > 1$ and $f_{3(i)} + f_{4(i)} > \frac{1}{2}$.

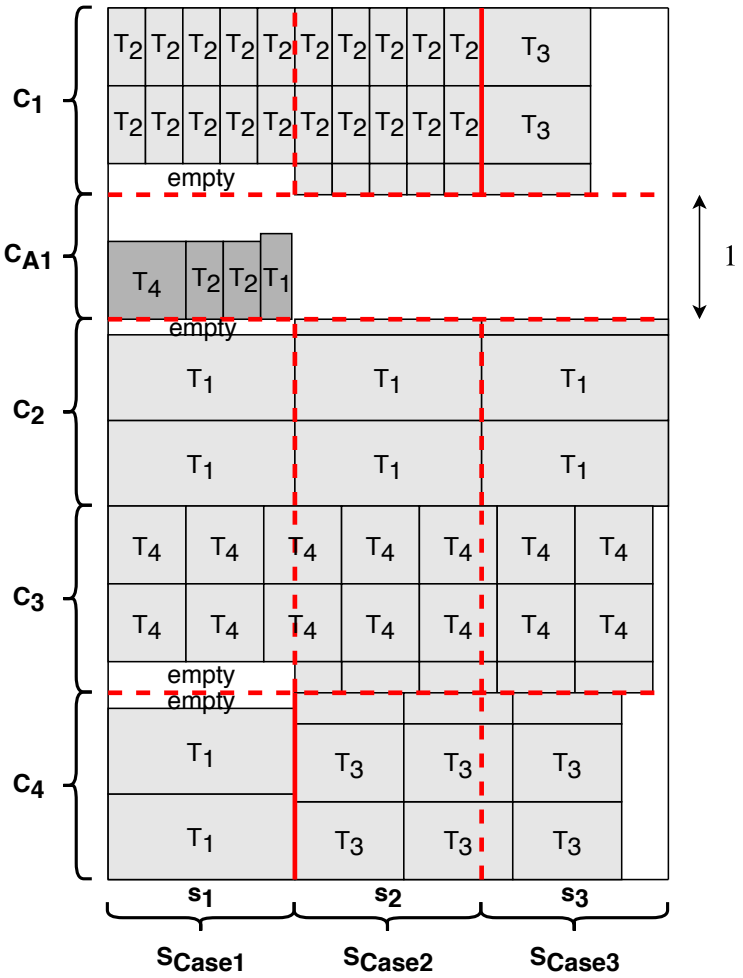
Case 1. $f_{1(i)} + f_{2(i)} \leq \frac{1}{2}$ and $f_{3(i)} + f_{4(i)} \leq \frac{1}{2}$



Case 1. $f_{1(i)} + f_{2(i)} \leq \frac{1}{2}$ and $f_{3(i)} + f_{4(i)} \leq \frac{1}{2}$

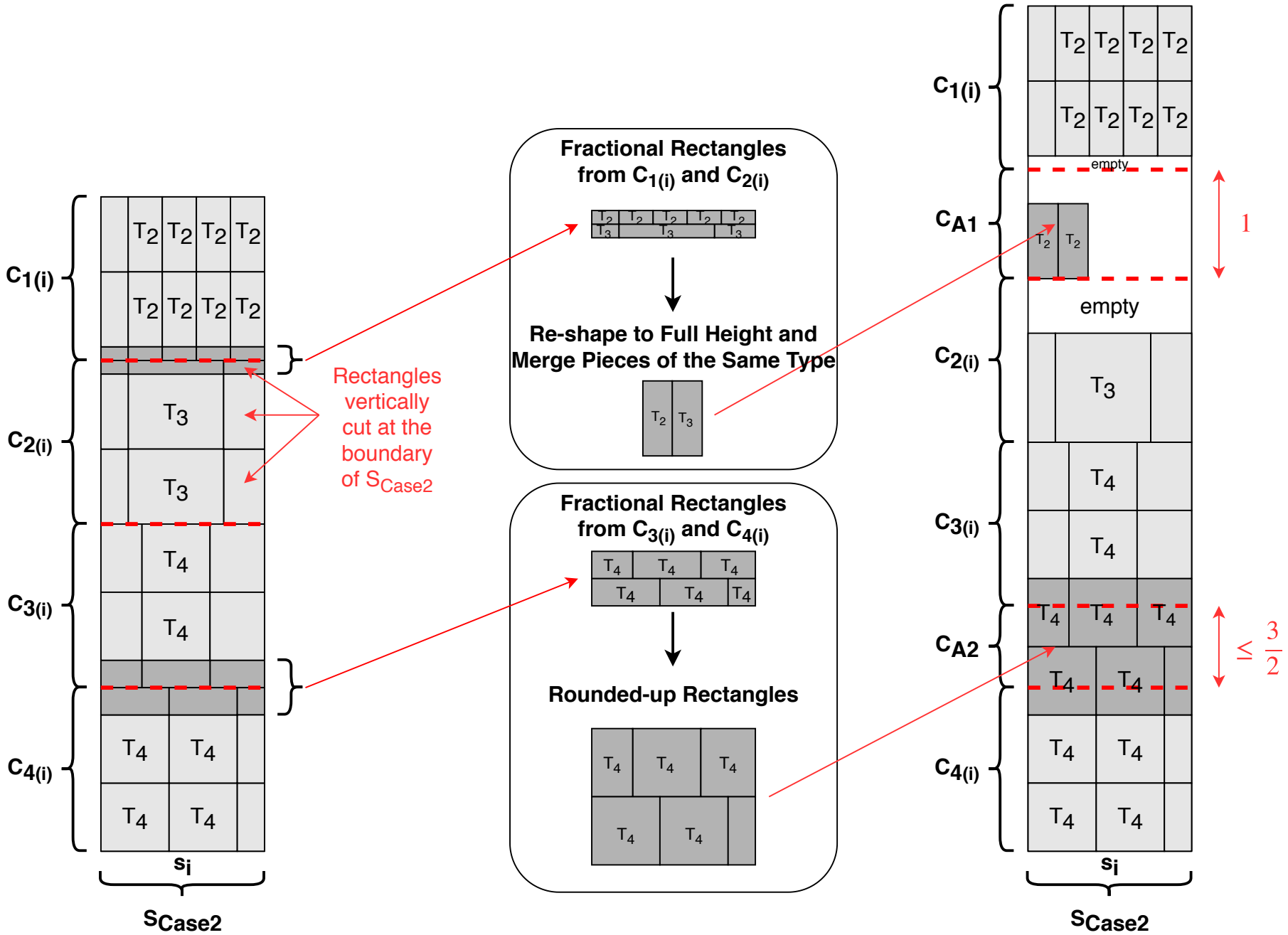


(e)

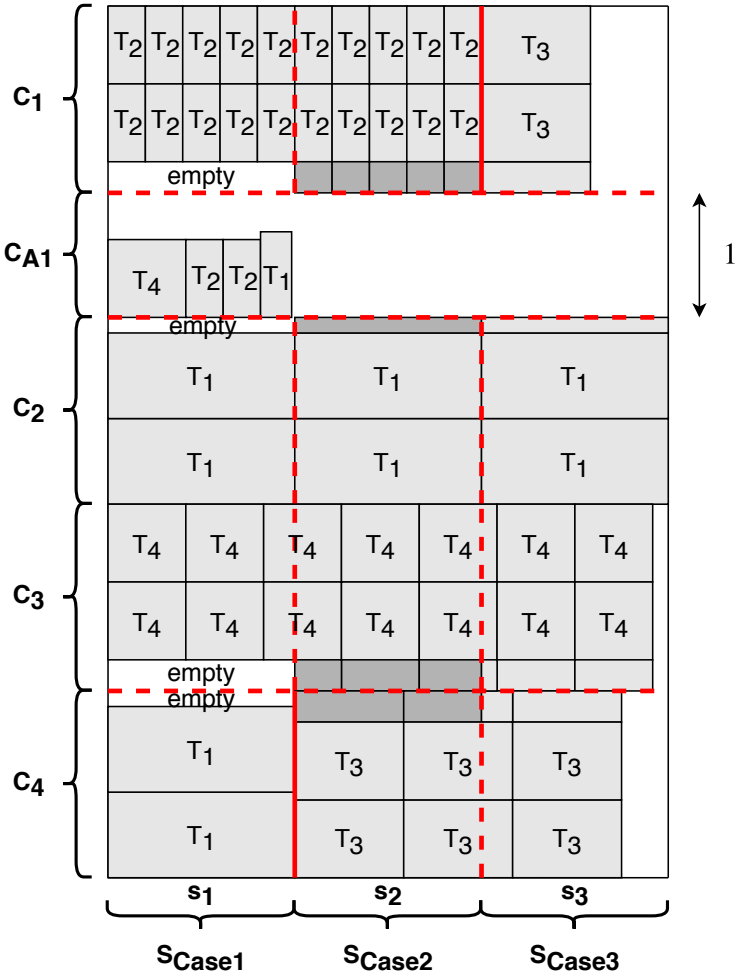


(f)

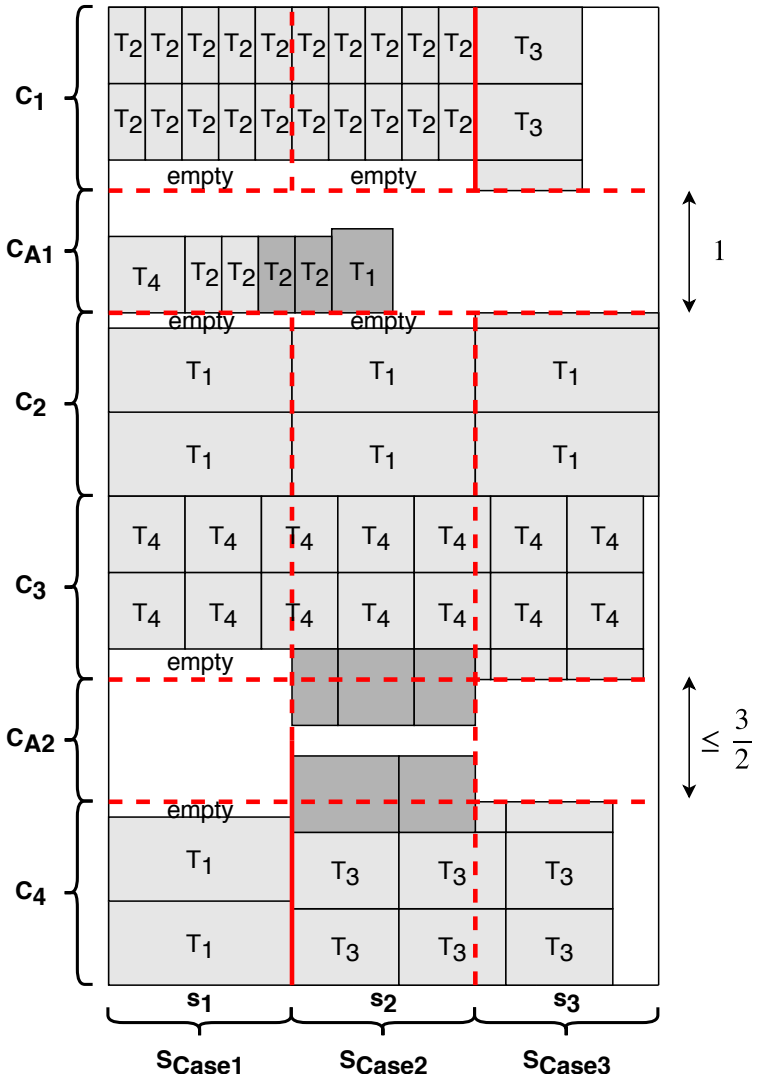
Case 2. $f_{1(i)} + f_{2(i)} \leq 1$ and $f_{3(i)} + f_{4(i)} > \frac{1}{2}$



Case 2. $f_{1(i)} + f_{2(i)} \leq 1$ and $f_{3(i)} + f_{4(i)} > \frac{1}{2}$

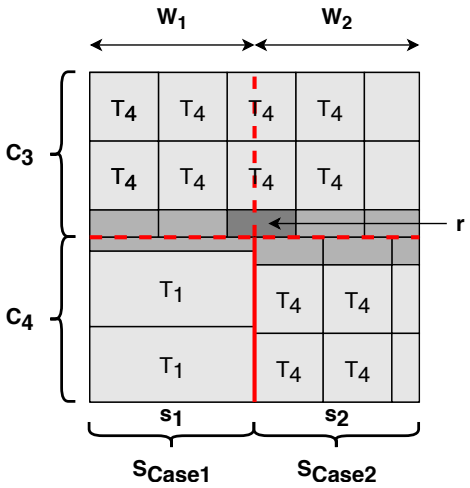


(g)

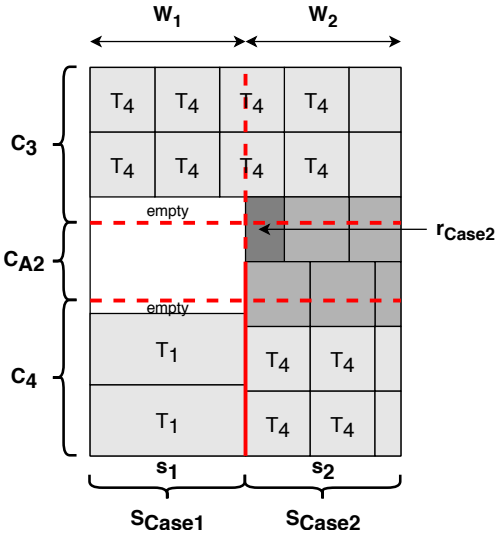


(h)

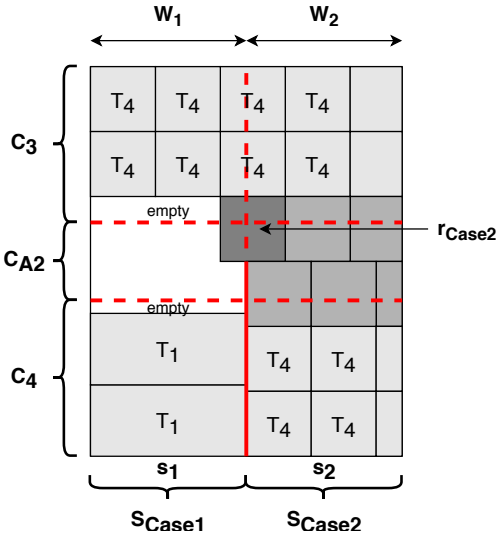
Fractional Rectangles Split between Case 1 and Case 2



(i)

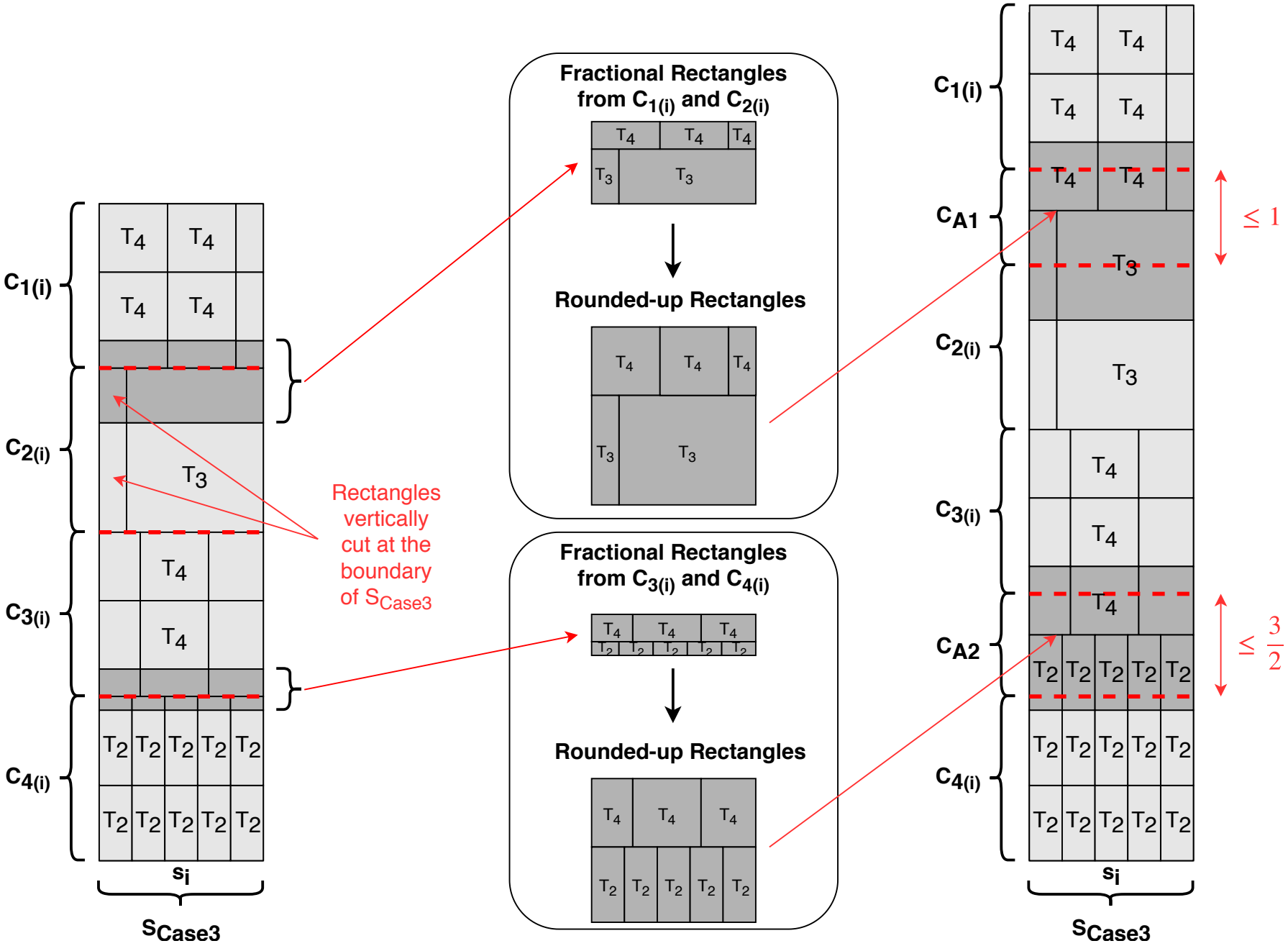


(j)

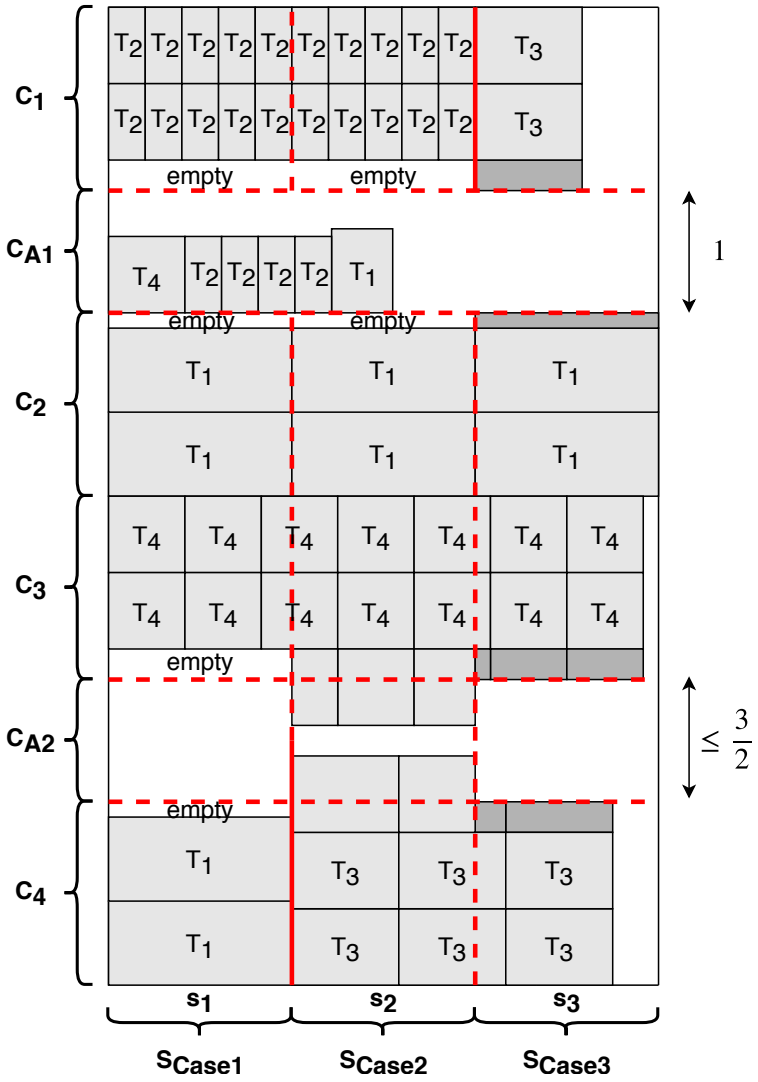


(k)

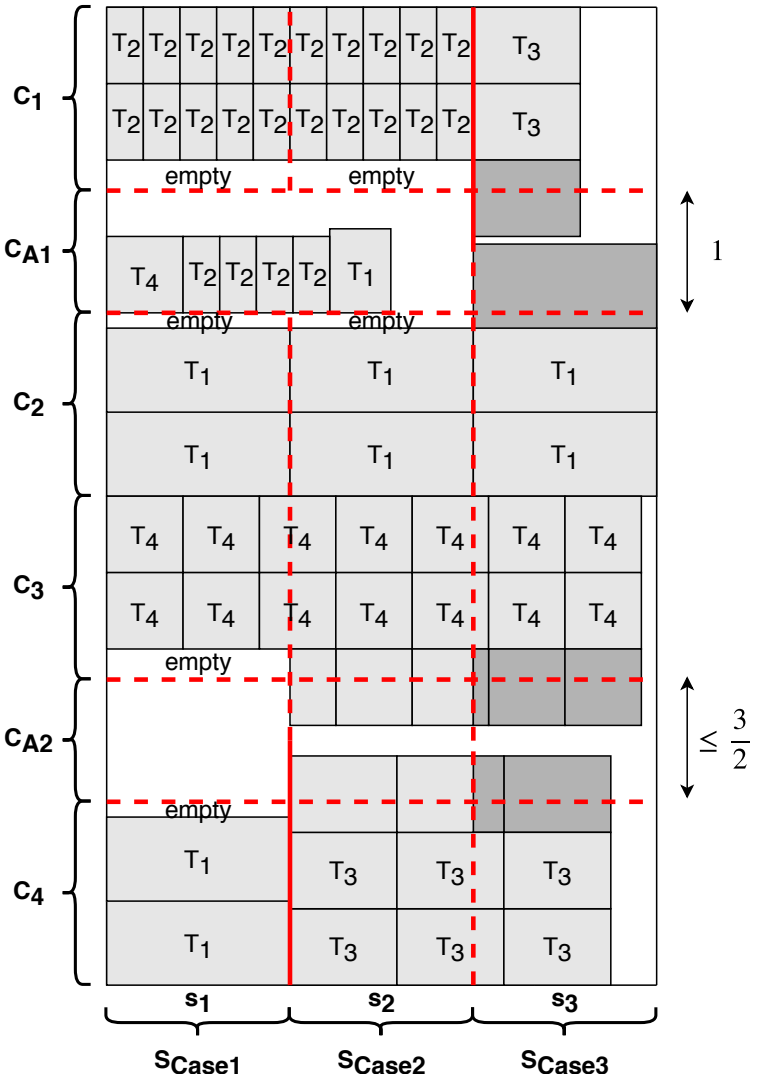
Case 3. $f_{1(i)} + f_{2(i)} > 1$ and $f_{3(i)} + f_{4(i)} > \frac{1}{2}$



Case 3. $f_{1(i)} + f_{2(i)} > 1$ and $f_{3(i)} + f_{4(i)} > \frac{1}{2}$

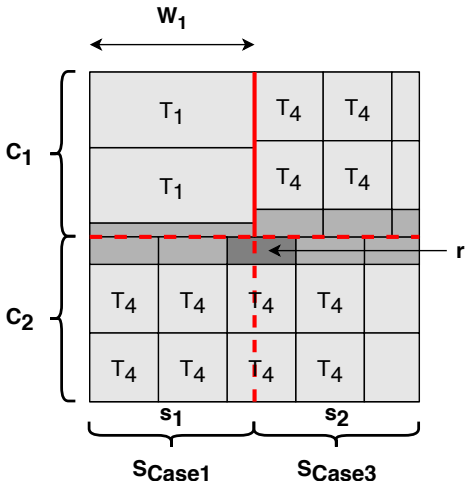


(l)

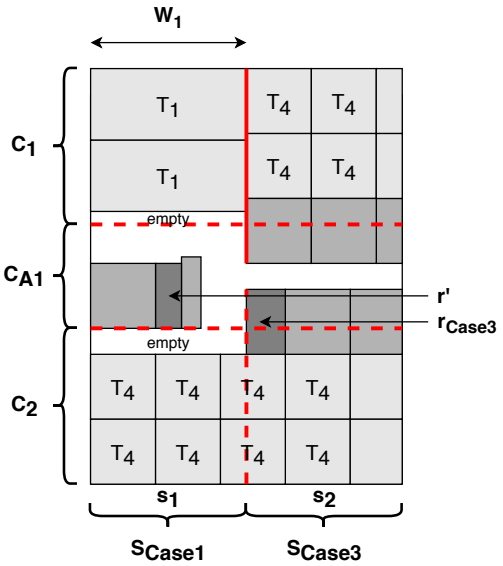


(m)

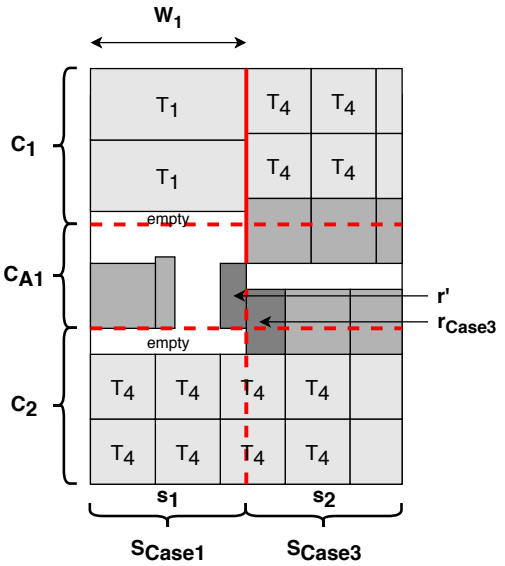
Fractional Rectangles Split between Case 1 and Case 3



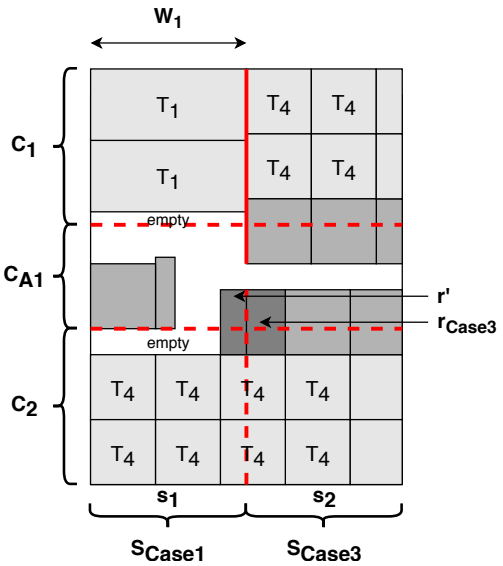
(n)



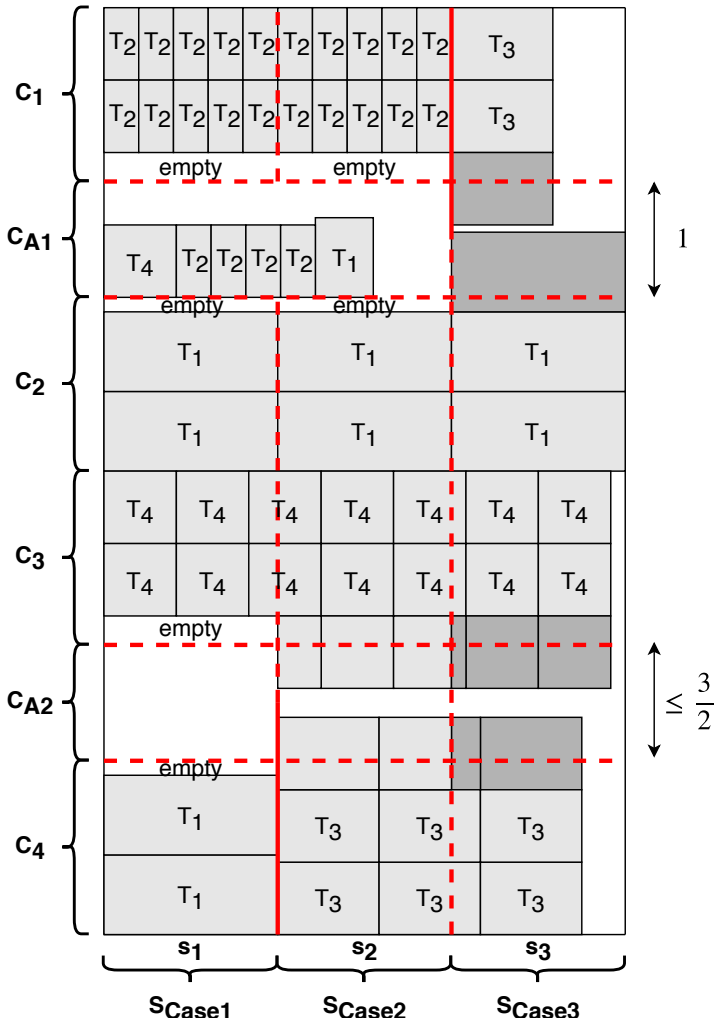
(o)



(p)



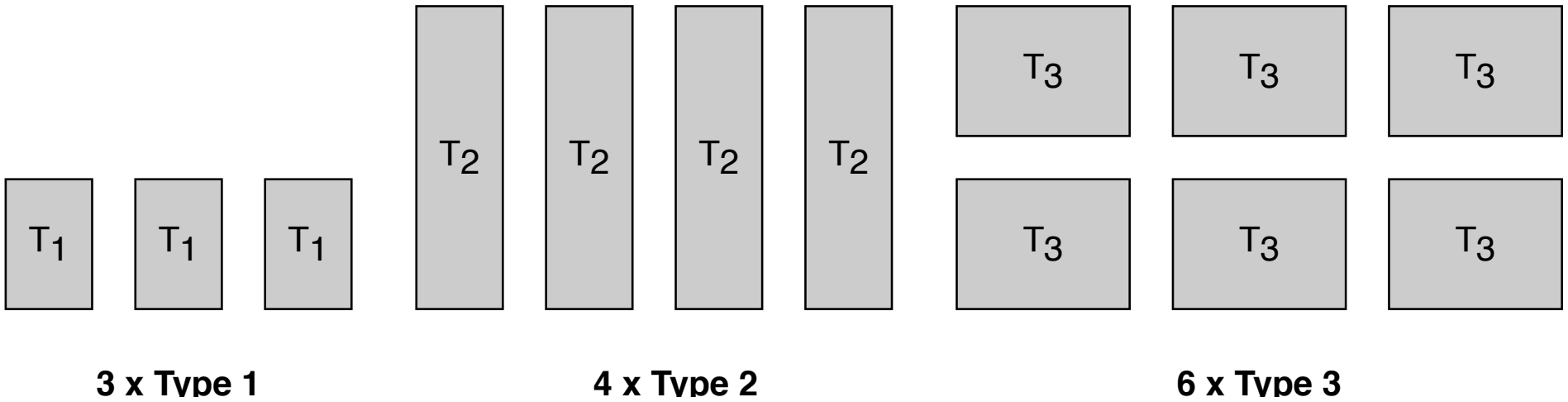
Approximation Ratio



- Our algorithm produces an integer packing of height at most $OPT + \frac{5}{2}$

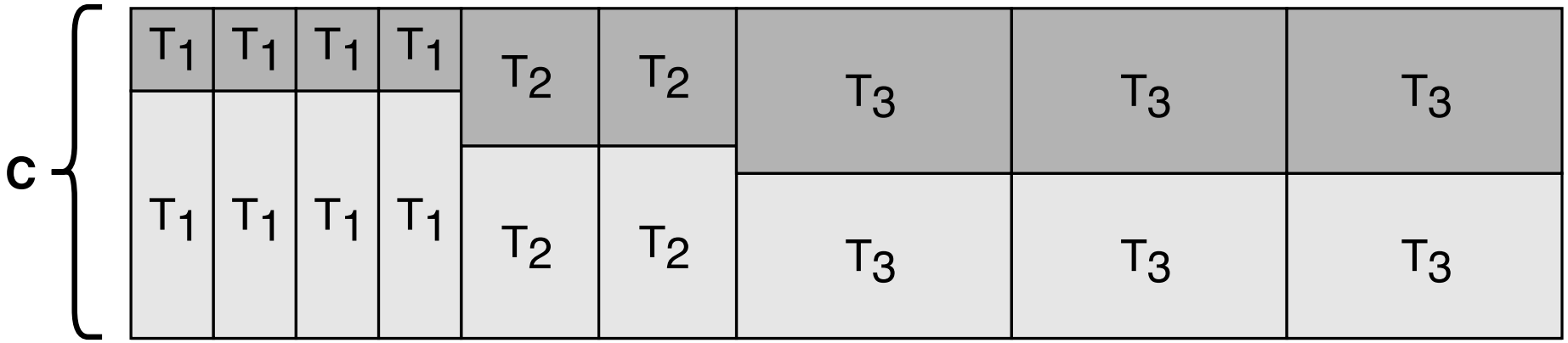
Outline

Input to HMSPP



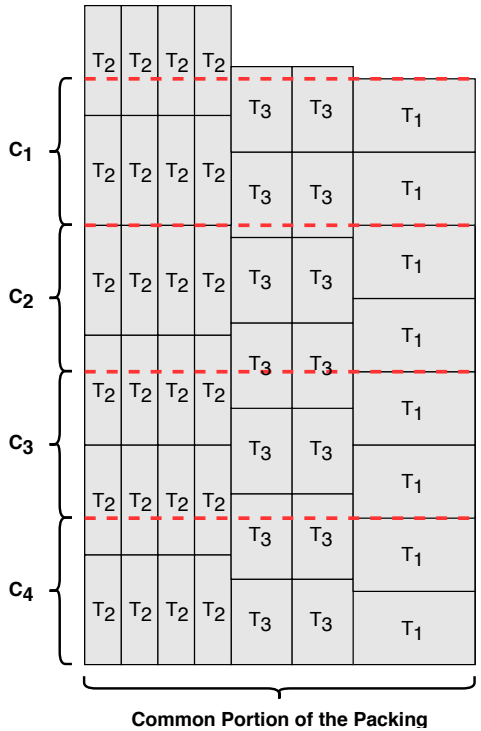
- A list of $3k$ numbers (width, height, multiplicty of each rectangle type)
- $\{(4, 6, 3), (4, 14, 4), (8, 6, 6)\}$

Fractional Packing



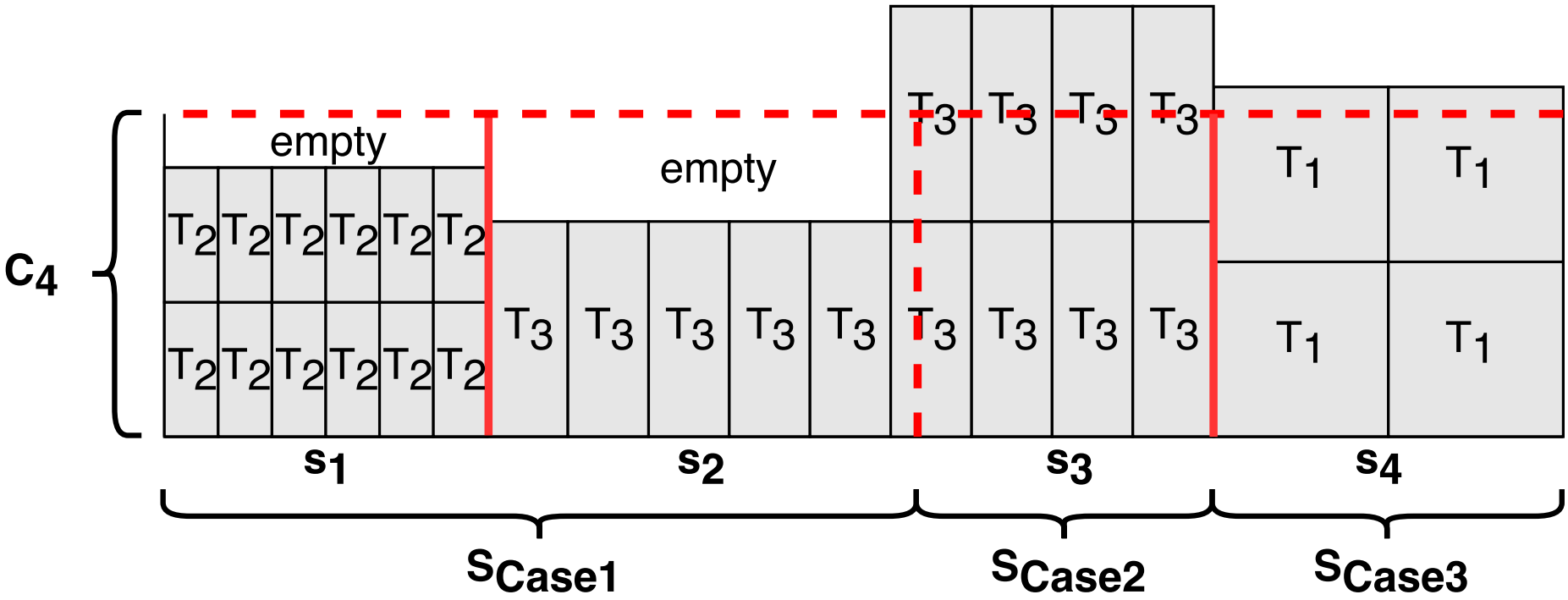
- For each configuration, a list of $O(k)$ numbers (type, number of rectangles packed side-by-side, number of rectangles packed one on top of the other)
- $\{(1, 4, 1.33), (2, 2, 1.71), (3, 3, 2)\}$

Outputting the Common Portion of the Packing



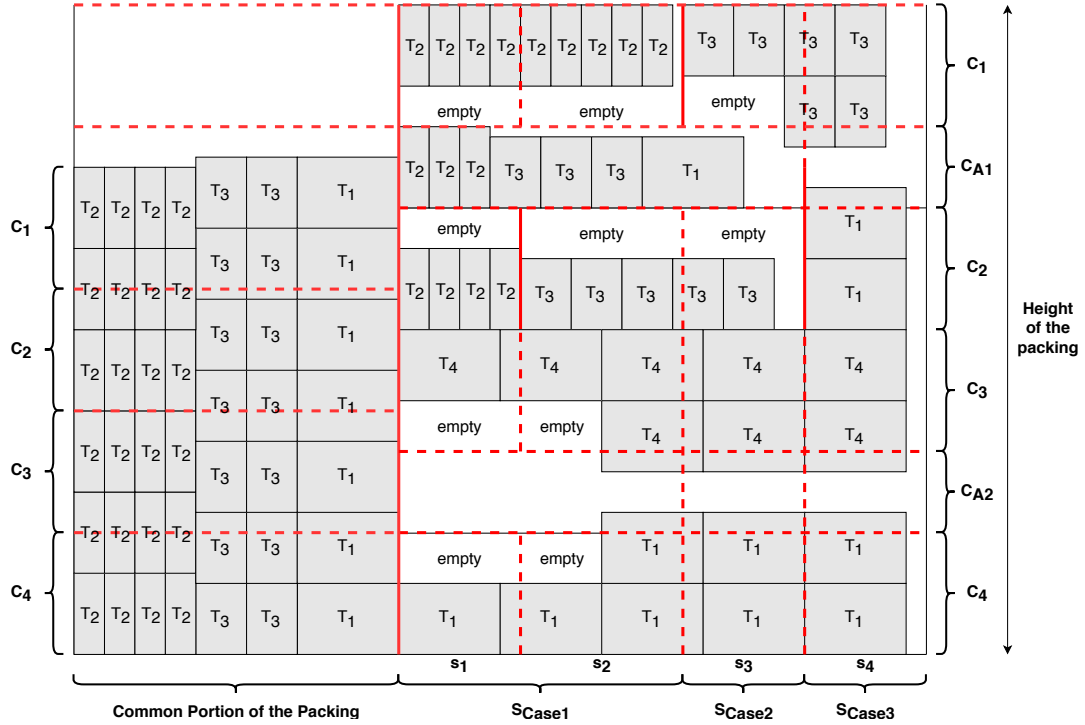
- A list of $O(k)$ numbers (type, number of rectangles packed side-by-side, number of rectangles packed one on top of the other)
- $\{(2, 4, 6), (3, 2, 7), (1, 1, 8)\}$

Outputting a Configuration



- A list of $O(k)$ numbers (type, number of rectangles packed side-by-side, number of rectangles packed one on top of the other)
- $\{(2, 6, 2), (3, 5, 1), (3, 4, 2), (1, 2, 2)\}$

Output of Our Algorithm: A List of $O(k^2)$ Numbers



- Common portion: $\{(2, 4, 6), (3, 2, 7), (1, 1, 8)\}$
- C_1 in uncommon portion: $\{(2, 9, 1), (3, 2, 1), (3, 2, 2)\}$
- C_{A1} in uncommon portion: $\{(2, 3, 1), (3, 3, 1), (1, 1, 1)\}$
- C_2 in uncommon portion: $\{(2, 4, 1), (3, 5, 1), (1, 1, 2)\}$
- C_3 in uncommon portion: $\{(4, 2, 1), (4, 3, 2)\}$
- C_4 in uncommon portion: $\{(1, 2, 1), (1, 3, 2)\}$

Running Time

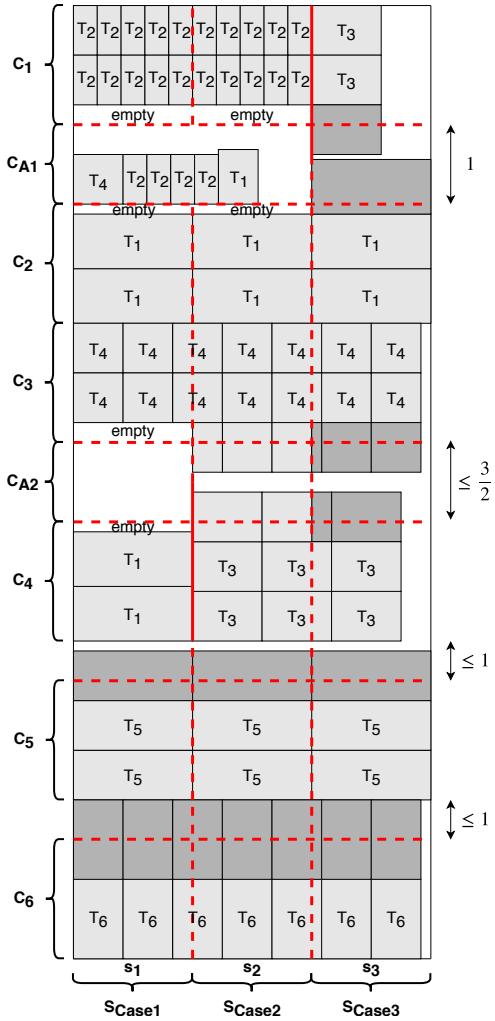
- Our algorithm produces an integer packing of height at most $OPT + \frac{5}{2}$ using at most $O(k^3)$ operations plus the time needed to compute the solution of the linear program:
  - Partitioning the list of numbers into common and uncommon portions: $O(k^2)$
  - Processing the common portion of the packing: $O(k)$
  - Sorting the rectangles in each configuration: $O(k^2)$
  - Processing a section $s_i \in S_{Case1}$: $O(k)$
  - Processing a section $s_i \in S_{Case2}$: $O(k)$
  - Processing a section $s_i \in S_{Case3}$: $O(k)$
  - There are $O(k^2)$ sections

Outline

Summary

- We presented an algorithm for HMSPP when $k = 3$ for which $SOL(I) \leq OPT(I) + \frac{5}{3}$. This algorithm can be generalized for any fixed value k to get $SOL(I) \leq OPT(I) + k - \frac{4}{3}$
- We presented an algorithm for HMSPP when $k = 4$ for which $SOL(I) \leq OPT(I) + \frac{5}{2}$. This algorithm can be generalized for any fixed value k to get $SOL(I) \leq OPT(I) + k - \frac{3}{2}$
- Our algorithms run in time $O(k^3)$ plus the time needed to compute the solution to the linear program, which was proven by Price et al. to be polynomial.

Generalizing the Algorithms for any Fixed k



- Configurations 5, 6, and so on, have their fractional rectangles rounded up

Future Work

- Could our algorithms be improved to decrease the height of the packings that they produce?
- Design algorithms for HMSP for the cases when $k = 5$, 6, etc.
- Design a generalized algorithm for any fixed value of k
- Study HMSPP with 90 degree rotations allowed, with three-dimensional rectangles, or with n-dimensional rectangles

Thank You!

Questions?