Allocating employees to cost centres

A case study

- We are given a number of employees (n)
  - where each employee has a name and a salary
- We are given a number of cost centres (m)
  - where each cost centre has a budget
  - all cost centres have the same budget
- Allocate the employees to cost centres, where
  - the sum of salaries in a cost centre is within budget

```
import java.util.*;

public class Person implements Comparable<Person> {

    String name;
    int salary;

    public Person(String name,int salary){
        this.name = name;
        this.salary = salary;
    }

    public int compareTo(Person p){
        return p.salary - salary;
    }

    public String toString(){
        return "("+ name +","+ salary +")";
    }
}
```

```
import java.util.*;

public class Person implements Comp                    {

    String name;
    int salary;

    public Person(String name,int
        this.name = name;
        this.salary = salary;
    }

    public int compareTo(Person p)
        return p.salary - salary;
    }

    public String toString(){
        return "("+ name +","+ sal
    }
}
```

C:\cpM\choco4\teamsWithBuc

File Edit Search View Encoding

2.txt | 7.txt | 0.txt

```
 1 Robyn 97
 2 Veronica 99
 3 Seth 72
 4 Cameron 43
 5 Constance 94
 6 Jenny 25
 7 Katharine 94
 8 Sheryl 83
 9 Tim 75
10 Edwin 26
11 Hilda 48
12 Ronnie 85
13 Marsha 39
14 Alvin 45
15 Michael 77
16 Nicole 83
17 Caroline 30
18 Stanley 40
19 Julie 98
20 Ernest 72
21 Kristine 77
22 Kristina 43
23 William 45
24 Maureen 39
25 Rachel 40
26 Gene 39
27 Oscar 28
28 Ethel 24
29 Stacey 54
30 Tom 32
31 Annette 46
32 Derek 94
```

Normal text file

```
public class Allocate {

    Person person[];       // the employees to be allocated to cost centres
    int salary[];          // salary[i] of ith person
    Model model;           // the model
    Solver solver;         // the solver object
    IntVar inCentre[][];   // inCentre[i][j] = 1 iff jth person works in ith cost centre
    IntVar centreSalary[]; // centreSalaries[i] is sum of the salaries in the ith cost centre
    int budget;            // the budget for each cost centre
    int n;                 // number of employees
    int m;                 // number of cost centres
    String id;             // an identification for the problem
```

```
public Allocate(String fname,int numberOfPeople,int numberOfCostCentres,int budget) throws Exception {
    n            = numberOfPeople;
    m            = numberOfCostCentres;
    this.budget  = budget;
    id           = fname;
    person       = new Person[n];
    salary       = new int[n];
    model        = new Model(id);
    solver       = model.getSolver();
    inCentre     = model.intVarMatrix("inCentre",m,n,0,1);
    centreSalary = model.intVarArray("centre salaries",m,0,budget);
```

```
Scanner sc = new Scanner(new File(fname));
for (int i=0;i<n;i++) person[i] = new Person(sc.next(),sc.nextInt());
sc.close();
for (int i=0;i<n;i++) salary[i] = person[i].salary;
```

```
for (int i=0;i<m;i++)
    model.scalar(inCentre[i],salary,"=",centreSalary[i]).post();
//
// constrain centreSalary[i] to be the scalar product of inCentre[i] and salary[i]
//


for (int i=0;i<n;i++)
    model.sum(ArrayUtils.getColumn(inCentre,i),"=",1).post();
//
// constrain a person such that he/she can only be in one cost centre at a time!
// i.e. the sum of a column of the array inCentre must be equal to 1,
//      such that a person is in exactly one cost centre
//
```

```
public static void main(String[] args) throws FileNotFoundException, IOException, Exception {
    if (args.length == 0){
        System.out.println("java Allocate fname budget #employees #centres");
        return;
    }
    String fname          = args[0];
    int budget            = Integer.parseInt(args[1]);
    int numberOfEmployees = Integer.parseInt(args[2]);
    int numberOfCentres   = Integer.parseInt(args[3]);

    Allocate alloc = new Allocate(fname,numberOfEmployees,numberOfCentres,budget);
    boolean solved = alloc.solve();
    System.out.println(solved);
    if (solved) System.out.println(alloc);
    System.out.println("nodes: " + alloc.solver.getMeasures().getNodeCount() +
                       "   cpu: " + alloc.solver.getMeasures().getTimeCount());
}
```

Command Prompt

n = 10

```
C:\cpM\choco4\teamsWithBudgets>java Allocate
java Allocate fname budget #employees #centres

C:\cpM\choco4\teamsWithBudgets>java Allocate 0.txt 150 10 10
true
0.txt  #employees: 10 #centres: 10 budget: 150
centre[0] 0 0 1 0 0 1 0 0 0 0 97
centre[1] 0 0 0 0 0 0 0 0 0 1 26
centre[2] 0 0 0 1 0 0 0 1 0 0 126
centre[3] 0 0 0 0 0 0 0 0 0 0 0
centre[4] 1 0 0 0 0 0 0 0 0 0 97
centre[5] 0 0 0 0 0 0 0 0 1 0 75
centre[6] 0 0 0 0 0 0 0 0 0 0 0
centre[7] 0 0 0 0 0 0 1 0 0 0 94
centre[8] 0 1 0 0 0 0 0 0 0 0 99
centre[9] 0 0 0 0 1 0 0 0 0 0 94

centre-0: (Seth,72) (Jenny,25)  ... cost: 97
centre-1: (Edwin,26)  ... cost: 26
centre-2: (Cameron,43) (Sheryl,83)  ... cost: 126
centre-3:  ... cost: 0
centre-4: (Robyn,97)  ... cost: 97
centre-5: (Tim,75)  ... cost: 75
centre-6:  ... cost: 0
centre-7: (Katharine,94)  ... cost: 94
centre-8: (Veronica,99)  ... cost: 99
centre-9: (Constance,94)  ... cost: 94

nodes: 79   cpu: 0.038137063

C:\cpM\choco4\teamsWithBudgets>
```

Command Prompt

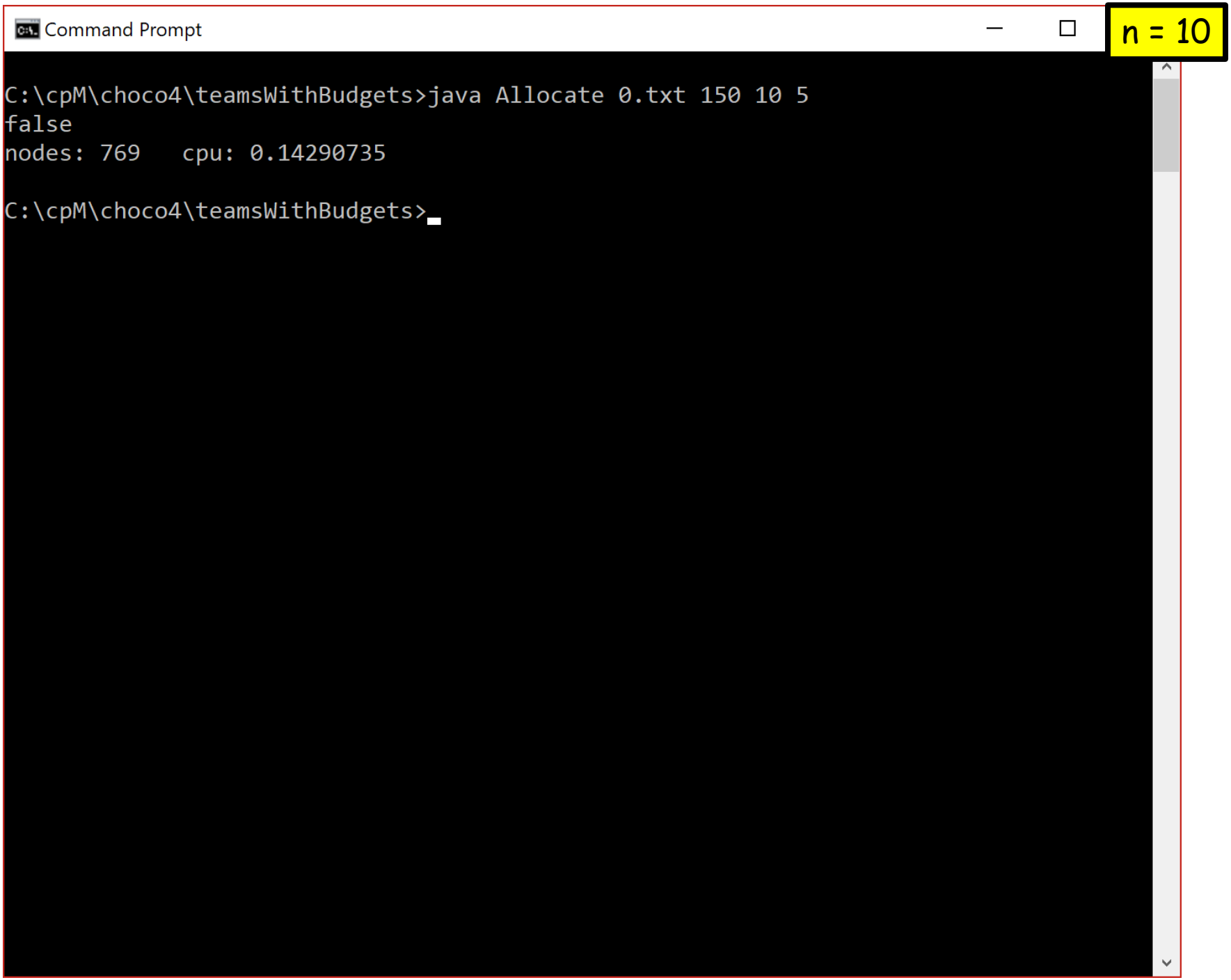
n = 10

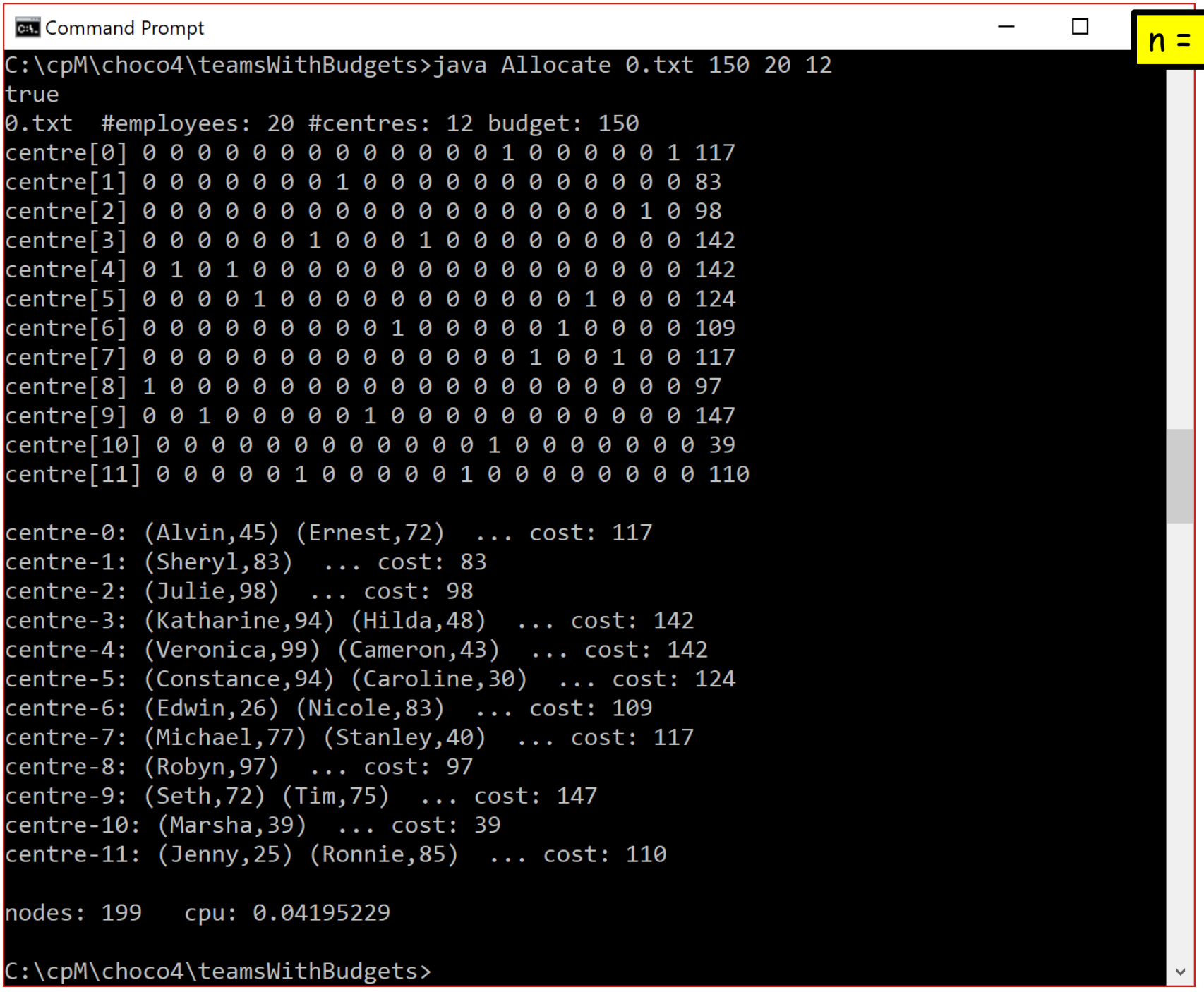
```
C:\cpM\choco4\teamsWithBudgets>java Allocate 0.txt 150 10 7
true
0.txt  #employees: 10 #centres: 7 budget: 150
centre[0] 0 0 0 0 0 1 1 0 0 0 119
centre[1] 0 0 0 1 0 0 0 0 1 0 118
centre[2] 0 1 0 0 0 0 0 0 0 0 99
centre[3] 0 0 1 0 0 0 0 0 0 0 72
centre[4] 0 0 0 0 1 0 0 0 0 1 120
centre[5] 1 0 0 0 0 0 0 0 0 0 97
centre[6] 0 0 0 0 0 0 0 1 0 0 83

centre-0: (Jenny,25) (Katharine,94)  ... cost: 119
centre-1: (Cameron,43) (Tim,75)  ... cost: 118
centre-2: (Veronica,99)  ... cost: 99
centre-3: (Seth,72)  ... cost: 72
centre-4: (Constance,94) (Edwin,26)  ... cost: 120
centre-5: (Robyn,97)  ... cost: 97
centre-6: (Sheryl,83)  ... cost: 83

nodes: 50   cpu: 0.02624389

C:\cpM\choco4\teamsWithBudgets>
```

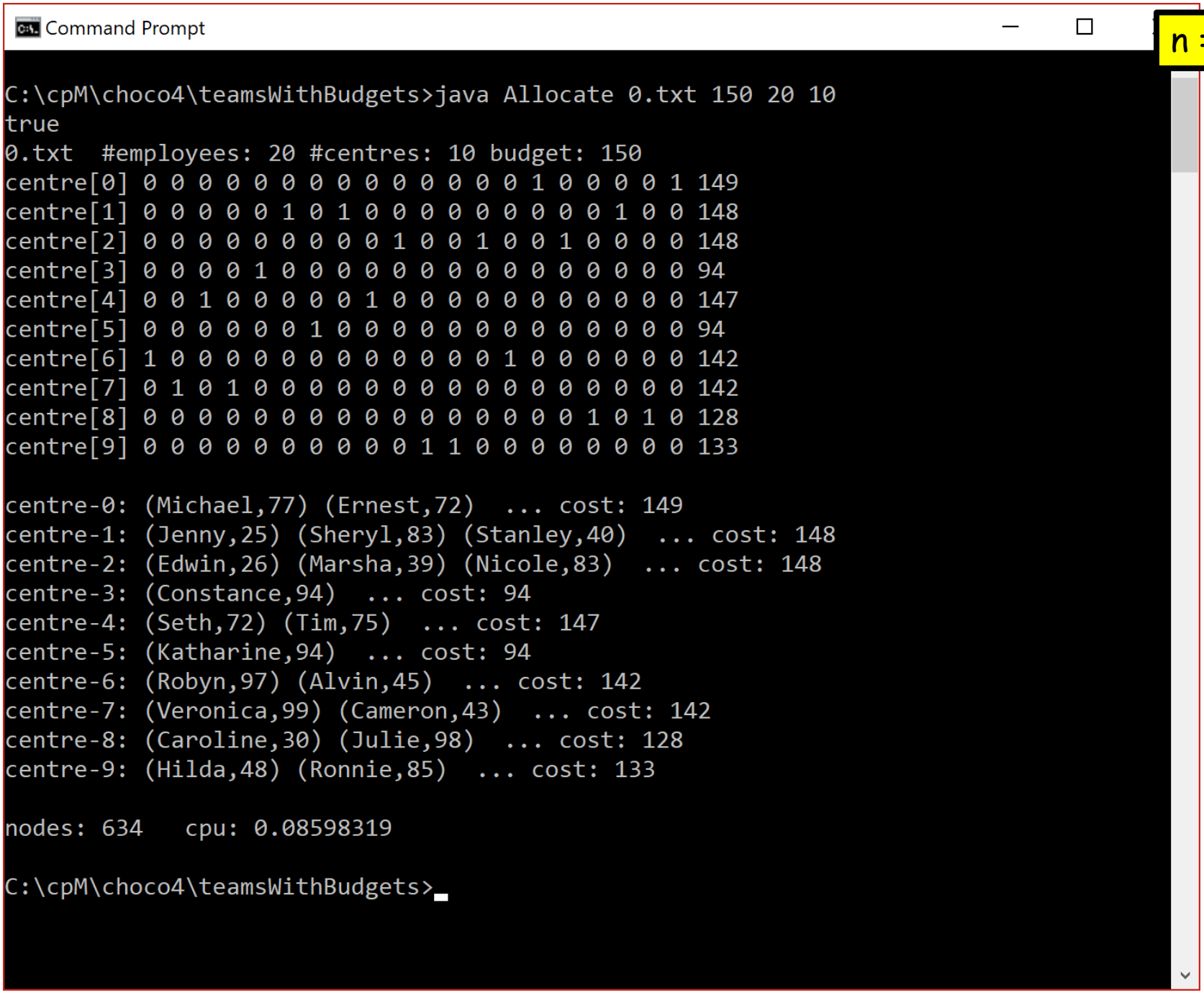
Command Prompt

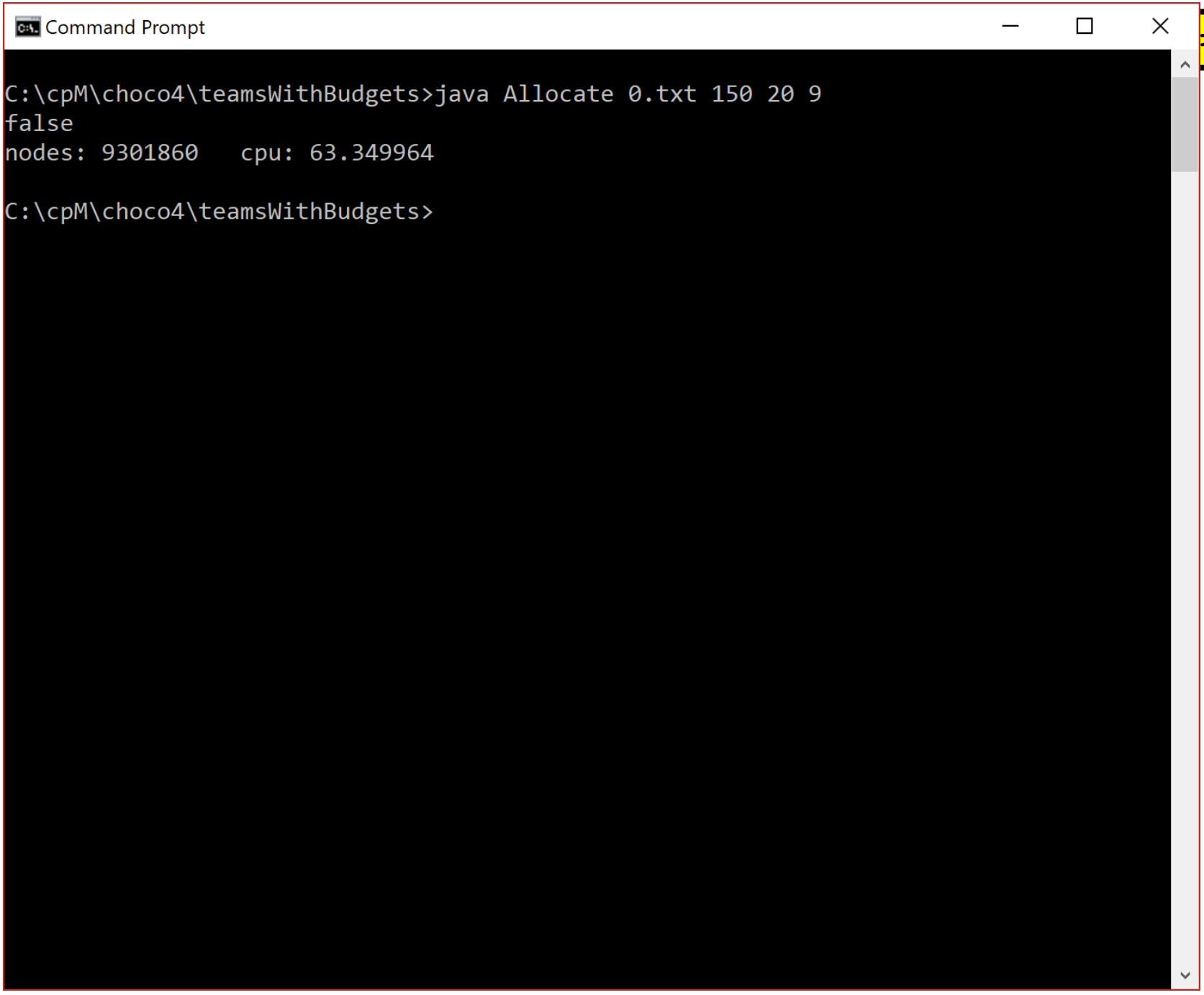
n = 10

```
C:\cpM\choco4\teamsWithBudgets>java Allocate 0.txt 150 10 6
true
0.txt  #employees: 10 #centres: 6 budget: 150
centre[0] 0 0 0 1 0 0 1 0 0 0 137
centre[1] 0 1 0 0 0 1 0 0 0 0 124
centre[2] 0 0 0 0 0 0 0 1 0 1 109
centre[3] 0 0 1 0 0 0 0 0 1 0 147
centre[4] 0 0 0 0 1 0 0 0 0 0 94
centre[5] 1 0 0 0 0 0 0 0 0 0 97

centre-0: (Cameron,43) (Katharine,94)  ... cost: 137
centre-1: (Veronica,99) (Jenny,25)  ... cost: 124
centre-2: (Sheryl,83) (Edwin,26)  ... cost: 109
centre-3: (Seth,72) (Tim,75)  ... cost: 147
centre-4: (Constance,94)  ... cost: 94
centre-5: (Robyn,97)  ... cost: 97

nodes: 53   cpu: 0.030337468

C:\cpM\choco4\teamsWithBudgets>
```


Command Prompt
n = 10
C:\cpM\choco4\teamsWithBudgets>java Allocate 0.txt 150 10 5
false
nodes: 769   cpu: 0.14290735
C:\cpM\choco4\teamsWithBudgets>


Command Prompt
n = 20
C:\cpM\choco4\teamsWithBudgets>java Allocate 0.txt 150 20 12
true
0.txt #employees: 20 #centres: 12 budget: 150
centre[0] 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 117
centre[1] 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 83
centre[2] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 98
centre[3] 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 142
centre[4] 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 142
centre[5] 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 124
centre[6] 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 109
centre[7] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 117
centre[8] 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 97
centre[9] 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 147
centre[10] 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 39
centre[11] 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 110
centre-0: (Alvin,45) (Ernest,72) ... cost: 117
centre-1: (Sheryl,83) ... cost: 83
centre-2: (Julie,98) ... cost: 98
centre-3: (Katharine,94) (Hilda,48) ... cost: 142
centre-4: (Veronica,99) (Cameron,43) ... cost: 142
centre-5: (Constance,94) (Caroline,30) ... cost: 124
centre-6: (Edwin,26) (Nicole,83) ... cost: 109
centre-7: (Michael,77) (Stanley,40) ... cost: 117
centre-8: (Robyn,97) ... cost: 97
centre-9: (Seth,72) (Tim,75) ... cost: 147
centre-10: (Marsha,39) ... cost: 39
centre-11: (Jenny,25) (Ronnie,85) ... cost: 110
nodes: 199   cpu: 0.04195229
C:\cpM\choco4\teamsWithBudgets>


Command Prompt
n = 20
C:\cpM\choco4\teamsWithBudgets>java Allocate 0.txt 150 20 10
true
0.txt #employees: 20 #centres: 10 budget: 150
centre[0] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 149
centre[1] 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 1 0 0 148
centre[2] 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 1 0 0 0 0 148
centre[3] 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 94
centre[4] 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 147
centre[5] 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 94
centre[6] 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 142
centre[7] 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 142
centre[8] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 128
centre[9] 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 133
centre-0: (Michael,77) (Ernest,72) ... cost: 149
centre-1: (Jenny,25) (Sheryl,83) (Stanley,40) ... cost: 148
centre-2: (Edwin,26) (Marsha,39) (Nicole,83) ... cost: 148
centre-3: (Constance,94) ... cost: 94
centre-4: (Seth,72) (Tim,75) ... cost: 147
centre-5: (Katharine,94) ... cost: 94
centre-6: (Robyn,97) (Alvin,45) ... cost: 142
centre-7: (Veronica,99) (Cameron,43) ... cost: 142
centre-8: (Caroline,30) (Julie,98) ... cost: 128
centre-9: (Hilda,48) (Ronnie,85) ... cost: 133
nodes: 634   cpu: 0.08598319
C:\cpM\choco4\teamsWithBudgets>


Command Prompt
C:\cpM\choco4\teamsWithBudgets>java Allocate 0.txt 150 20 9
false
nodes: 9301860   cpu: 63.349964
C:\cpM\choco4\teamsWithBudgets>

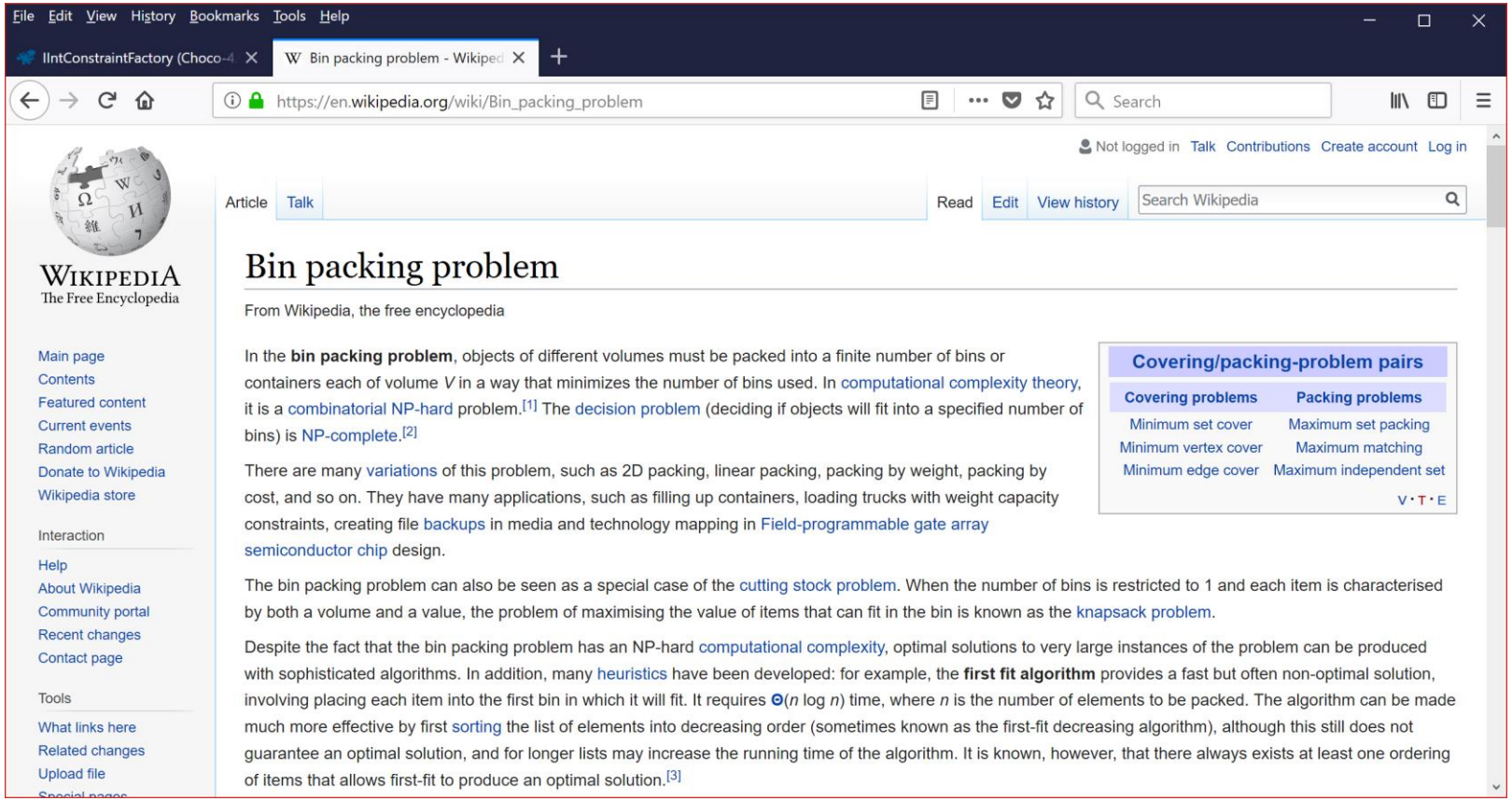
= 20

variable & value ordering

What are decision variables and what order are values picked?

```
// solve using value ordering over decision variables
boolean solve(){
    //solver.setSearch(Search.minDomLBSearch(ArrayUtils.flatten(inCentre)));
    //solver.setSearch(Search.minDomUBSearch(ArrayUtils.flatten(inCentre)));
    return solver.solve();
}
```

This is a classic problem …


File Edit View History Bookmarks Tools Help
IIntConstraintFactory (Choco-4
Bin packing problem - Wikiped
https://en.wikipedia.org/wiki/Bin_packing_problem
Search
Not logged in Talk Contributions Create account Log in
Article Talk
Read Edit View history
Search Wikipedia
WIKIPEDIA
The Free Encyclopedia
Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikipedia store
Interaction
Help
About Wikipedia
Community portal
Recent changes
Contact page
Tools
What links here
Related changes
Upload file
Bin packing problem
From Wikipedia, the free encyclopedia
In the bin packing problem, objects of different volumes must be packed into a finite number of bins or containers each of volume V in a way that minimizes the number of bins used. In computational complexity theory, it is a combinatorial NP-hard problem.[1] The decision problem (deciding if objects will fit into a specified number of bins) is NP-complete.[2]
There are many variations of this problem, such as 2D packing, linear packing, packing by weight, packing by cost, and so on. They have many applications, such as filling up containers, loading trucks with weight capacity constraints, creating file backups in media and technology mapping in Field-programmable gate array semiconductor chip design.
Covering/packing-problem pairs
Covering problems
Packing problems
Minimum set cover
Maximum set packing
Minimum vertex cover
Maximum matching
Minimum edge cover
Maximum independent set
V · T · E
The bin packing problem can also be seen as a special case of the cutting stock problem. When the number of bins is restricted to 1 and each item is characterised by both a volume and a value, the problem of maximising the value of items that can fit in the bin is known as the knapsack problem.
Despite the fact that the bin packing problem has an NP-hard computational complexity, optimal solutions to very large instances of the problem can be produced with sophisticated algorithms. In addition, many heuristics have been developed: for example, the first fit algorithm provides a fast but often non-optimal solution, involving placing each item into the first bin in which it will fit. It requires Θ(n log n) time, where n is the number of elements to be packed. The algorithm can be made much more effective by first sorting the list of elements into decreasing order (sometimes known as the first-fit decreasing algorithm), although this still does not guarantee an optimal solution, and for longer lists may increase the running time of the algorithm. It is known, however, that there always exists at least one ordering of items that allows first-fit to produce an optimal solution.[3]

[SR1] BIN PACKING

INSTANCE: Finite set U of items, a size s(u) in Z^+ for each u in U, a positive integer bin capacity B, and a positive integer K.

QUESTION: Is there a partition of U into disjoint sets U_1, U_2, …, U_k such that the sum of the sizes of the items in each U_i is B or less?

Garey & Johnson
"Computers and Intractability: A guide to the theory of NP-Completeness"

[SR1] BIN PACKING

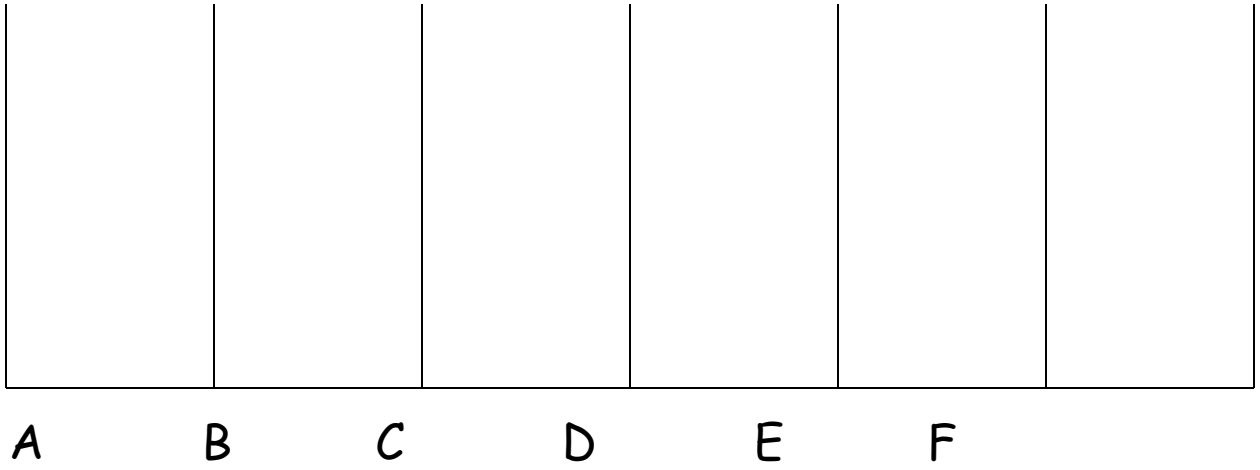
INSTANCE: Finite set U of items, a size s(u) in Z^+ for each u in U, a positive integer bin capacity B, and a positive integer K.

QUESTION: Is there a partition of U into disjoint sets U_1, U_2, …, U_k such that the sum of the sizes of the items in each U_i is B or less?

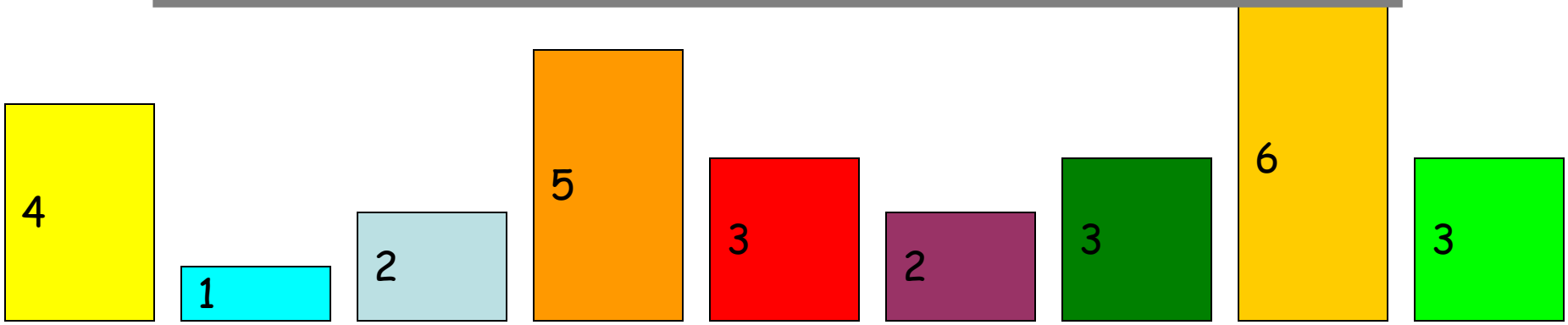
Garey & Johnson
"Computers and Intractability: A guide to the theory of NP-Completeness"

Is there a heuristic we might use?

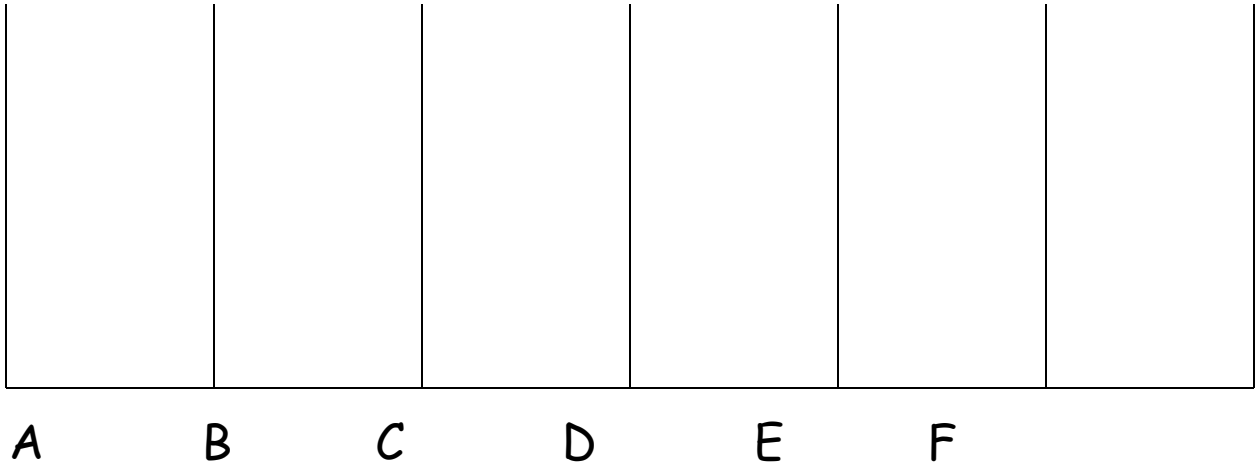
Bin Packing
First fit decreasing algorithm



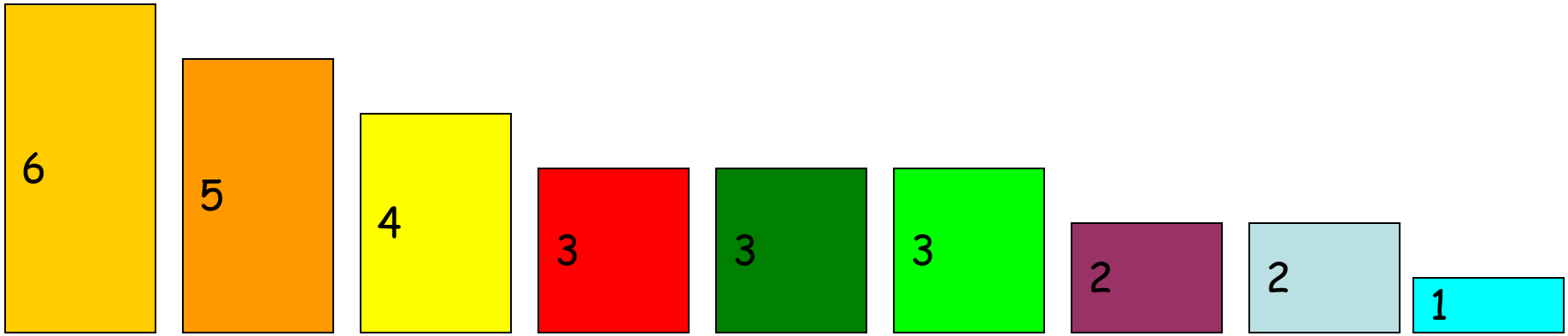
With the first fit decreasing algorithm we sort the blocks into descending order first.



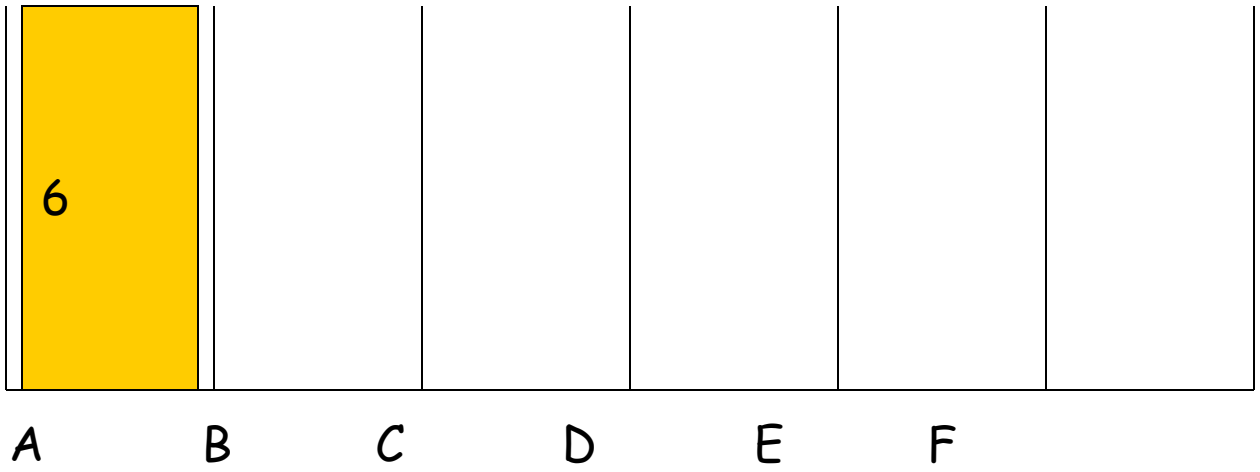
Bin Packing
First fit decreasing algorithm



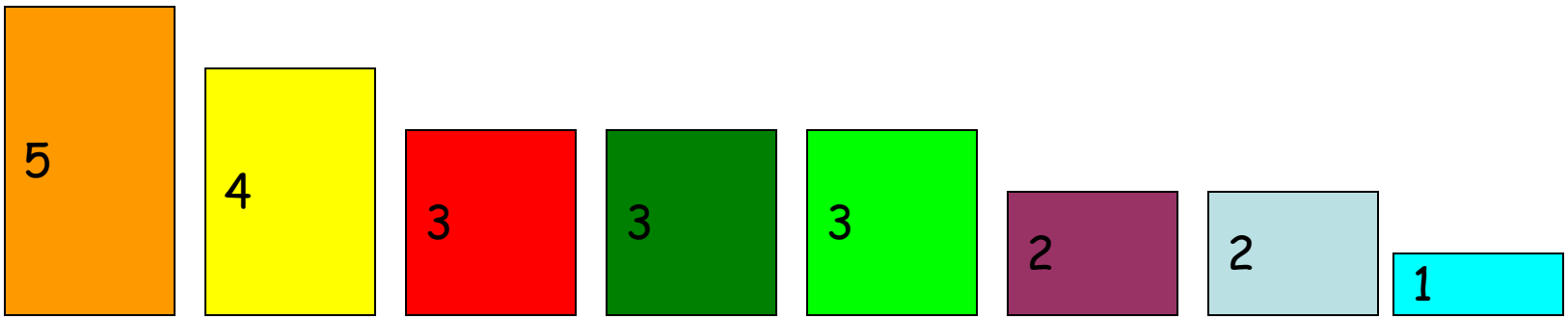
Now we use the first fit algorithm



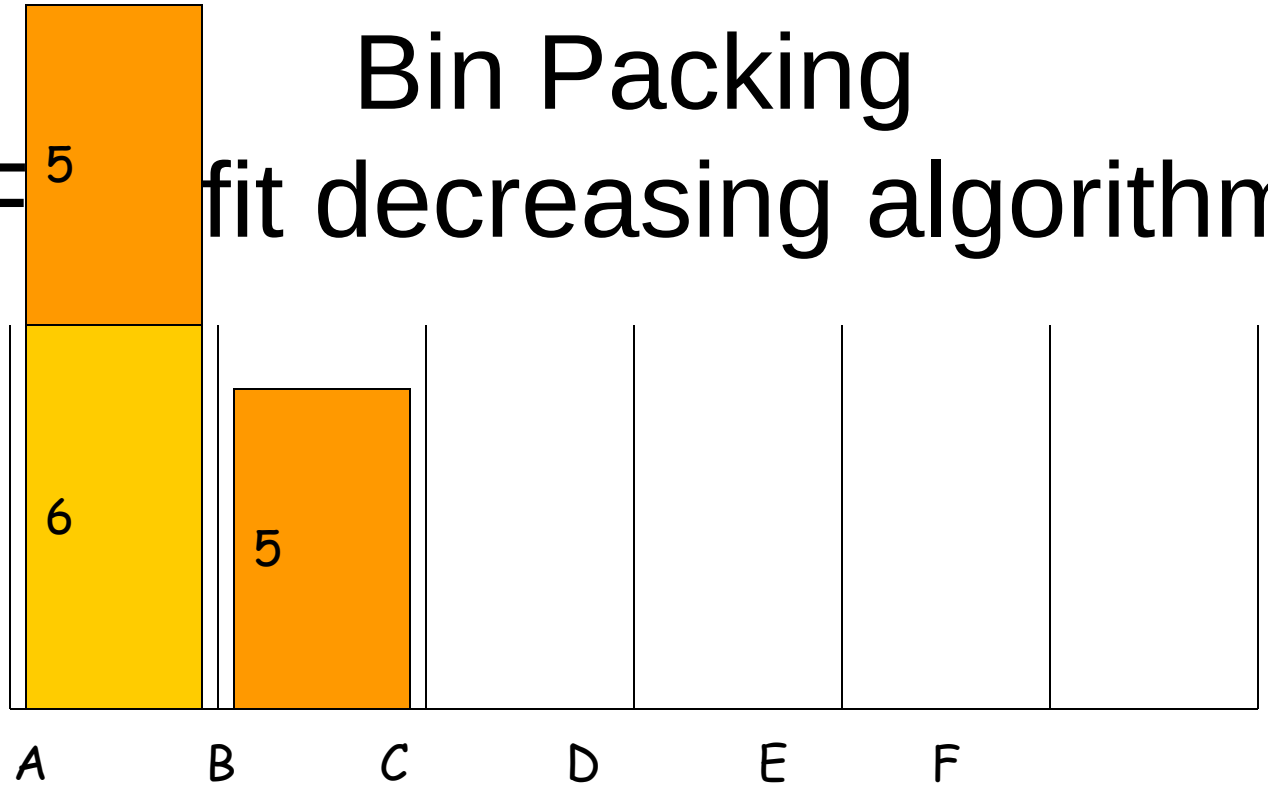
Bin Packing
First fit decreasing algorithm



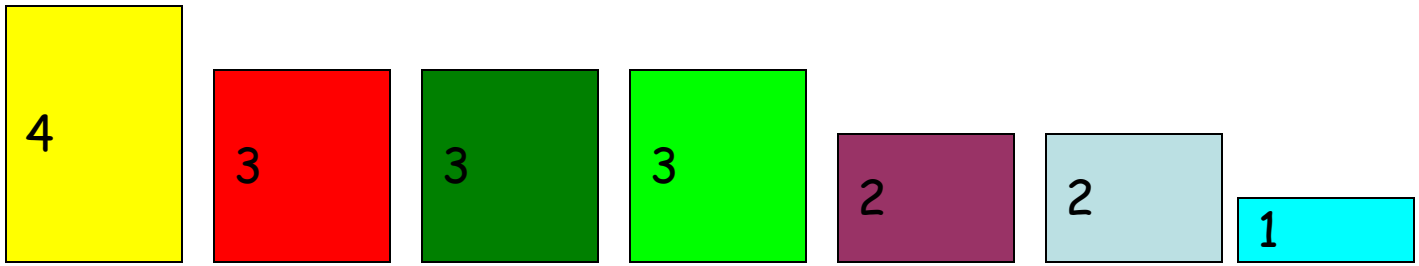
Now we use the first fit algorithm



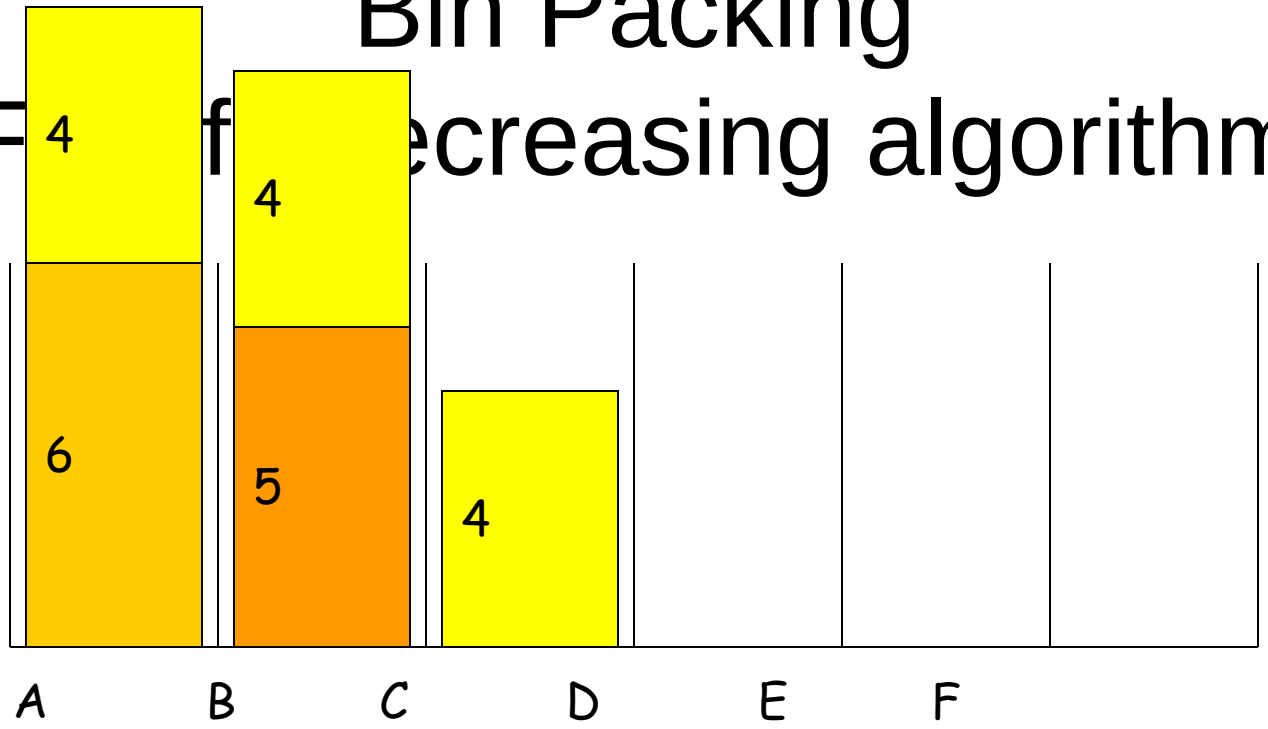
Bin Packing
First fit decreasing algorithm



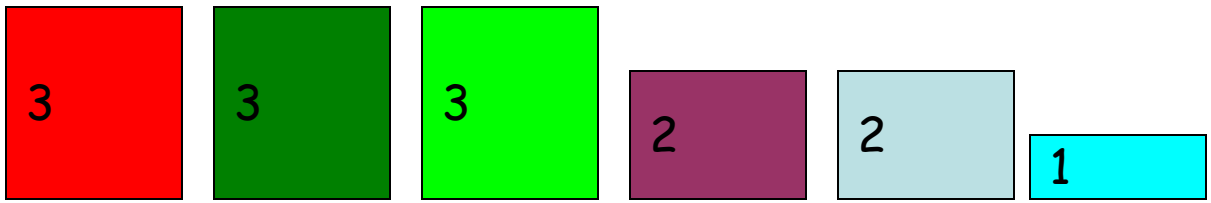
Now we use the first fit algorithm



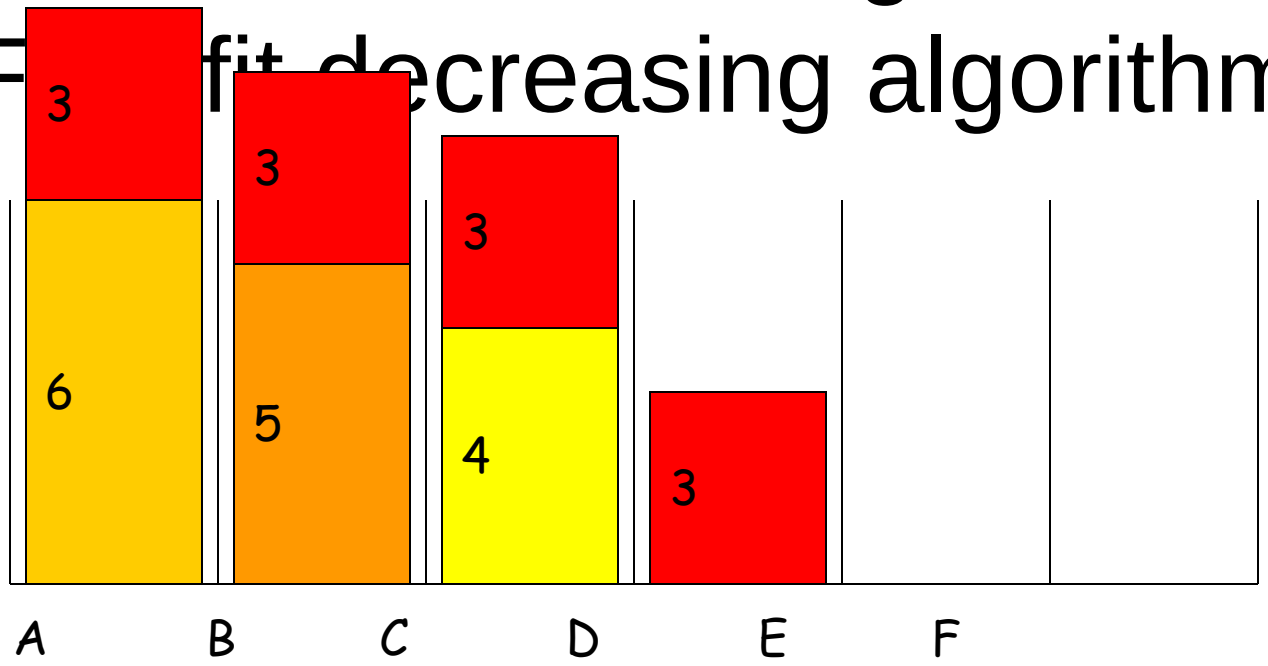
Bin Packing
First fit decreasing algorithm



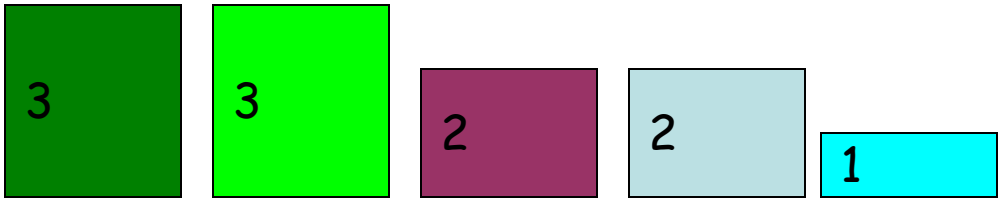
Now we use the first fit algorithm



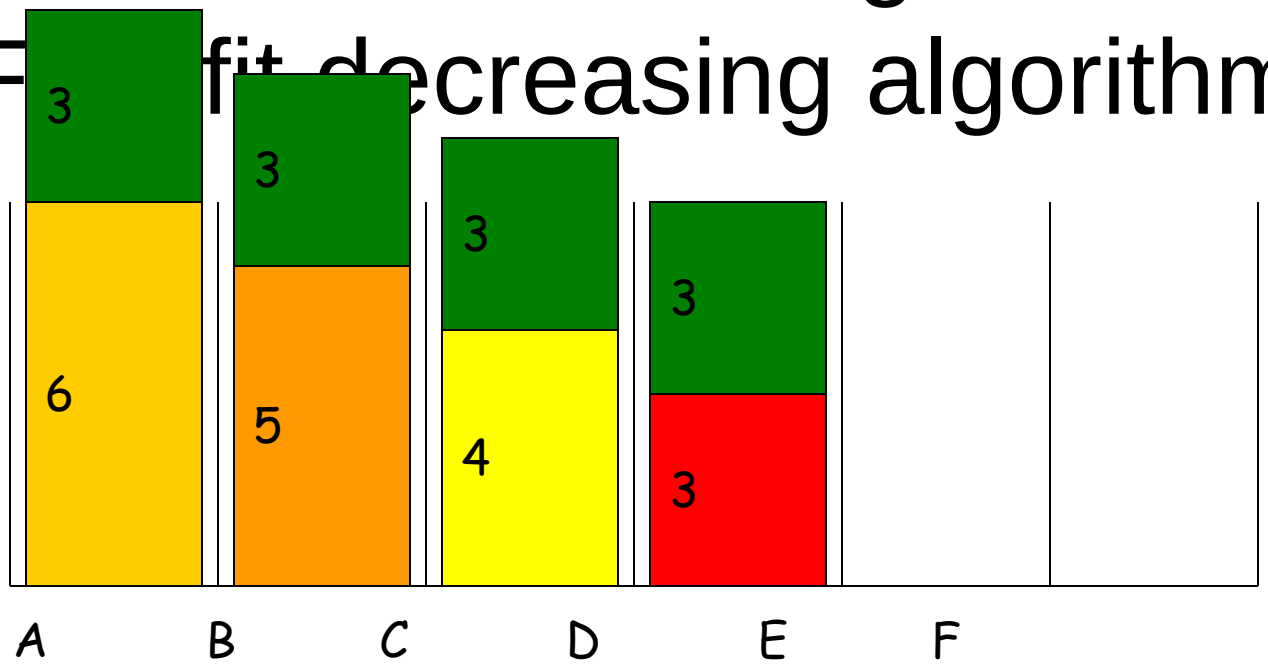
Bin Packing
First fit decreasing algorithm



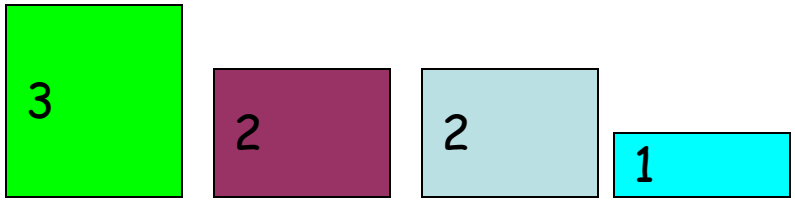
Now we use the first fit algorithm



Bin Packing
First fit decreasing algorithm

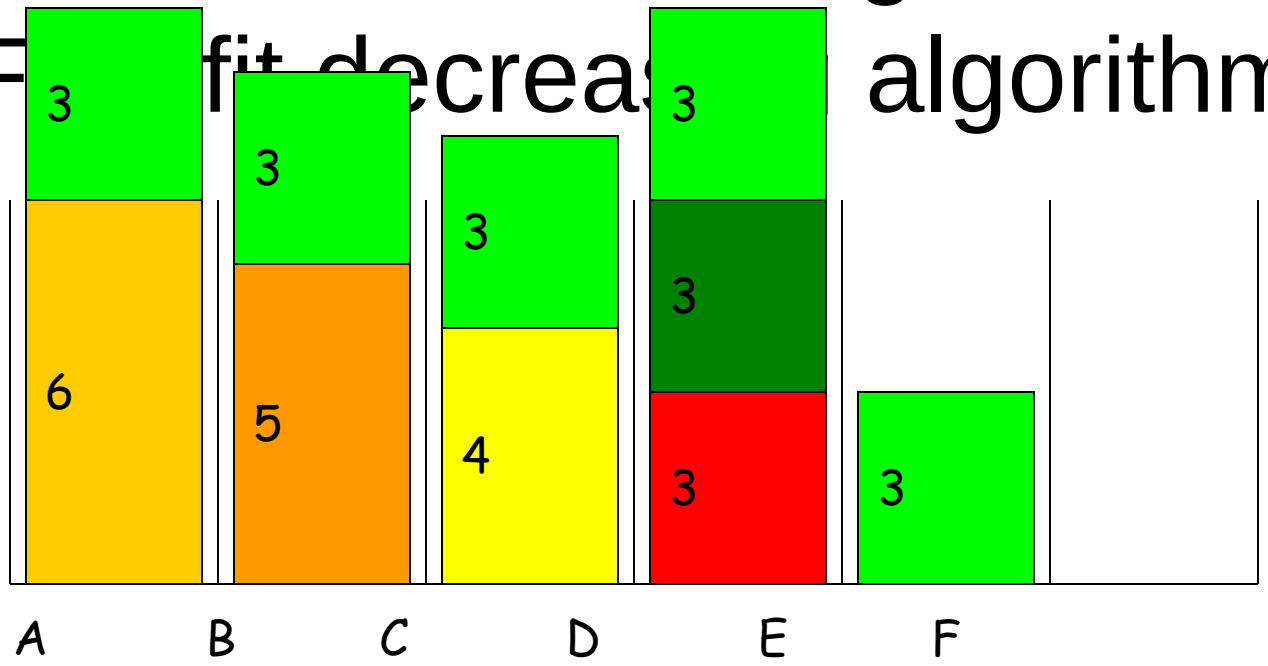


Now we use the first fit algorithm

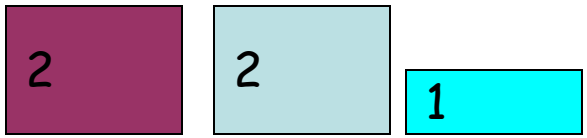


Bin Packing

First fit decreasing algorithm

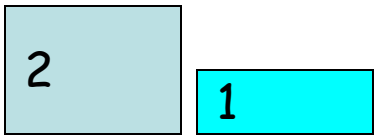


Now we use the first fit algorithm

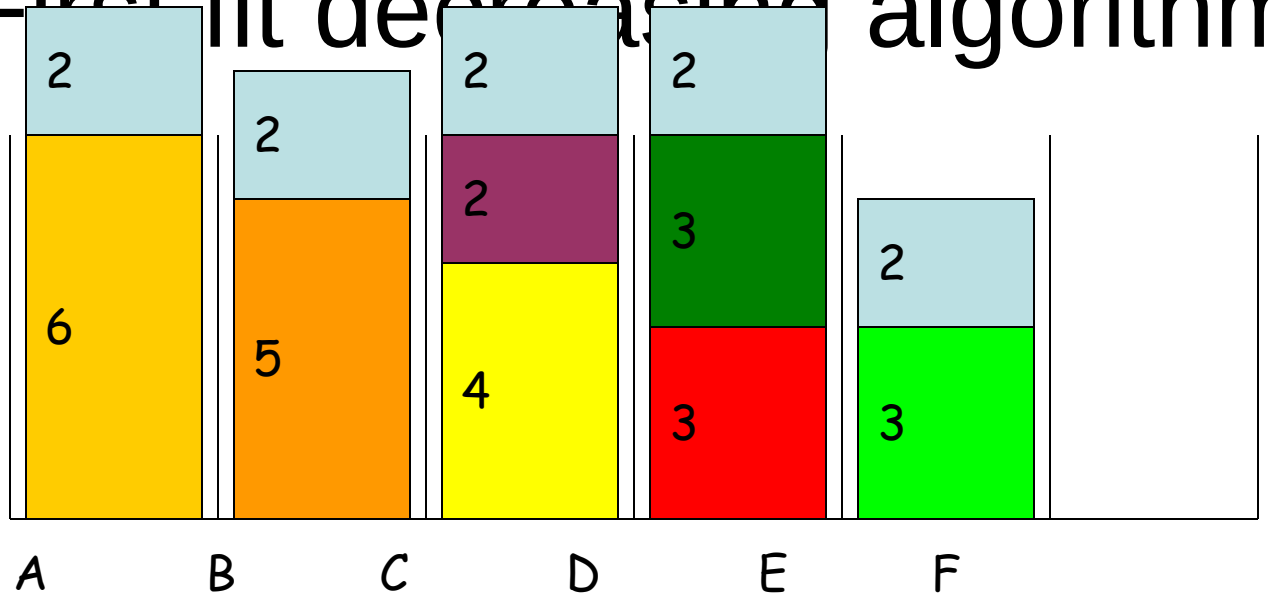


Bin Packing
First fit decreasing algorithm

Now we use the first fit algorithm

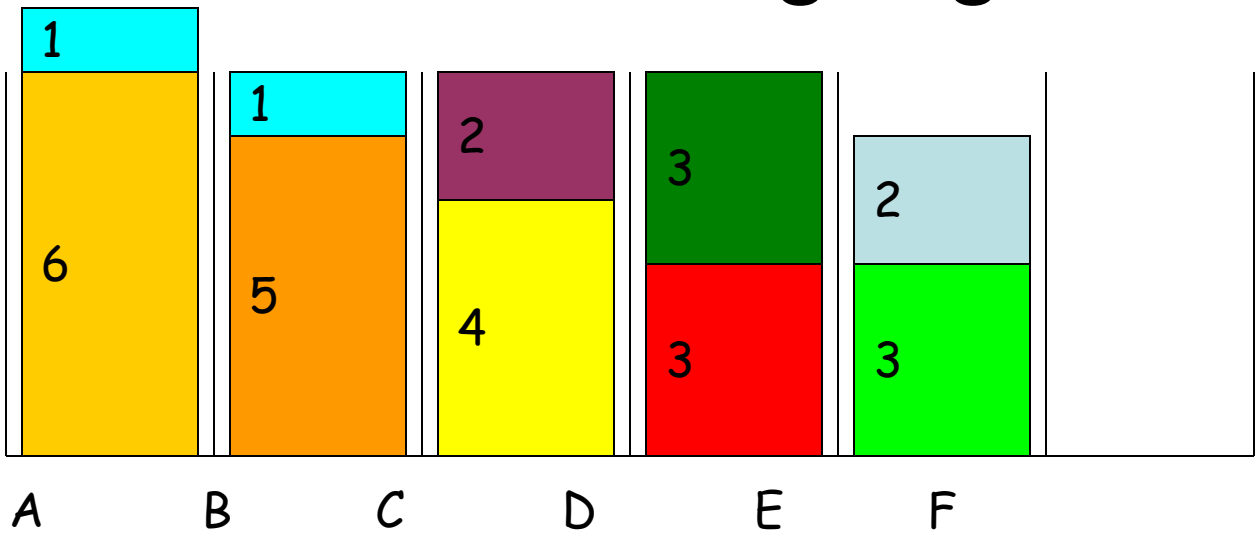


Bin Packing
First fit decreasing algorithm



Now we use the first fit algorithm

Bin Packing
First fit decreasing algorithm



We have packed them into 5 bins.

$(na)^2\sqrt{\text{MIC}}$

Home

About NANAMIC & membership

Contact NANAMIC

Forthcoming NANAMIC Events

Newsletter

Conferences

Consultations

Teaching & Learning Resources

Welcome to:

National Association for Numeracy and Mathematics in Colleges

Improving Learning in Mathematics

approaches that encourage active learning including group work, discussion and open questions.

Friday 23rd October 2009.

University of Wales, Caerleon campus, Newport

News:

```
Scanner sc = new Scanner(new File(fname));
for (int i=0;i<n;i++) person[i] = new Person(sc.next(),sc.nextInt());
sc.close();

// EDIT
// Arrays.sort(person);
//
// first fit decreasing
//

for (int i=0;i<n;i++) salary[i] = person[i].salary;
```

Try 1st fit decreasing (see Person)

Slow proving optimality

Don't have a test that sum of numbers over capacity is less than or equal to the number of bins available!

Symmetries?

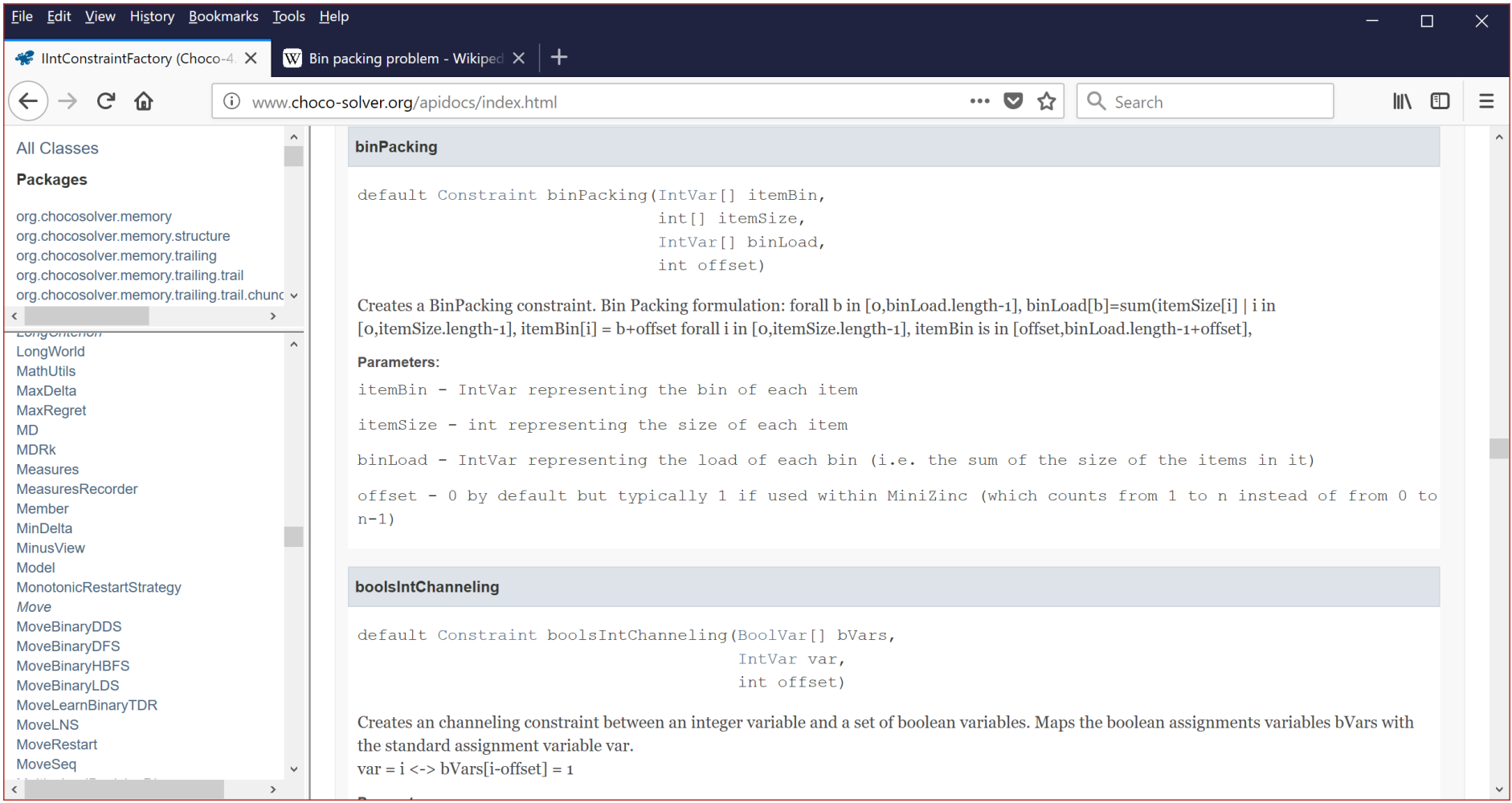
Are there any symmetries that are slowing down search?

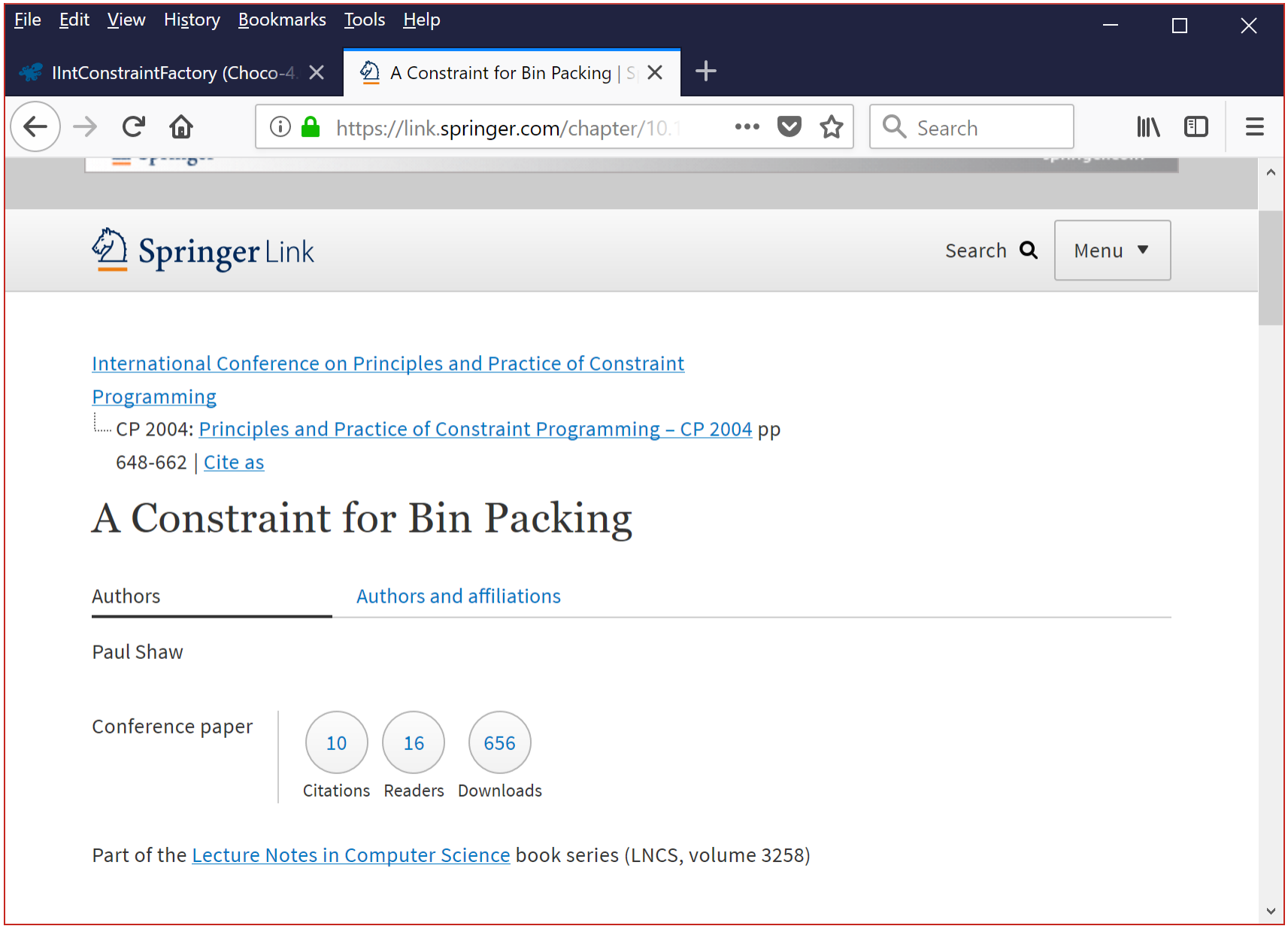
Can we remove those symmetries?

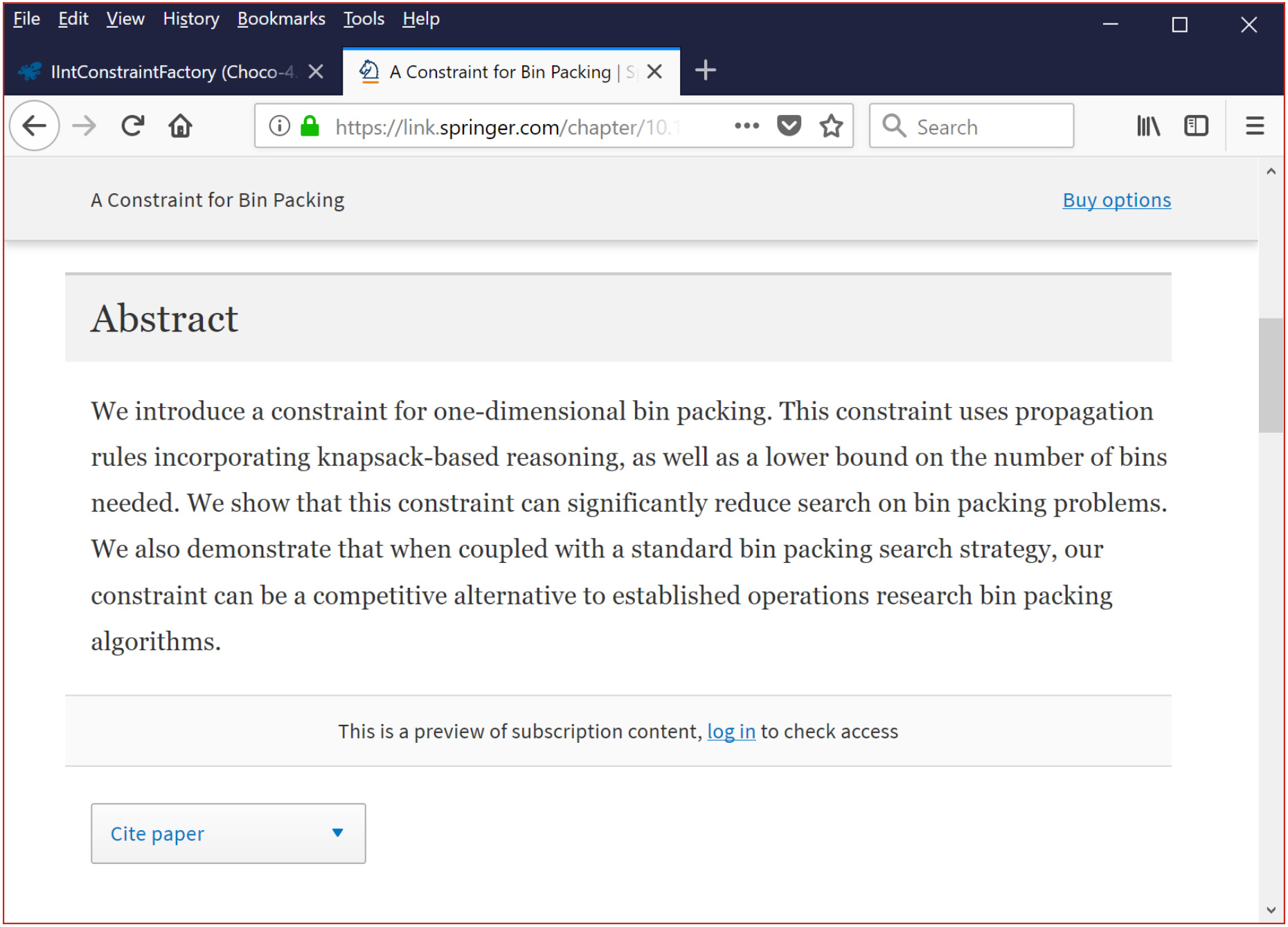
What are the symmetries in this problem?

```
// EDIT
//for (int i=0;i<m-1;i++)
// model.arithm(centreSalary[i],">=",centreSalary[i+1]).post();
//
// symmetry breaking consistent with first fit decreasing
// costliest cost centres have low index
//

// EDIT
//for (int centre=0;centre<m-1;centre++)
//    model.lexLessEq(inCentre[centre+1],inCentre[centre]).post();
//
// symmetry breaking such that inCentre[i] lex>= inCentre[i+1]
//
```

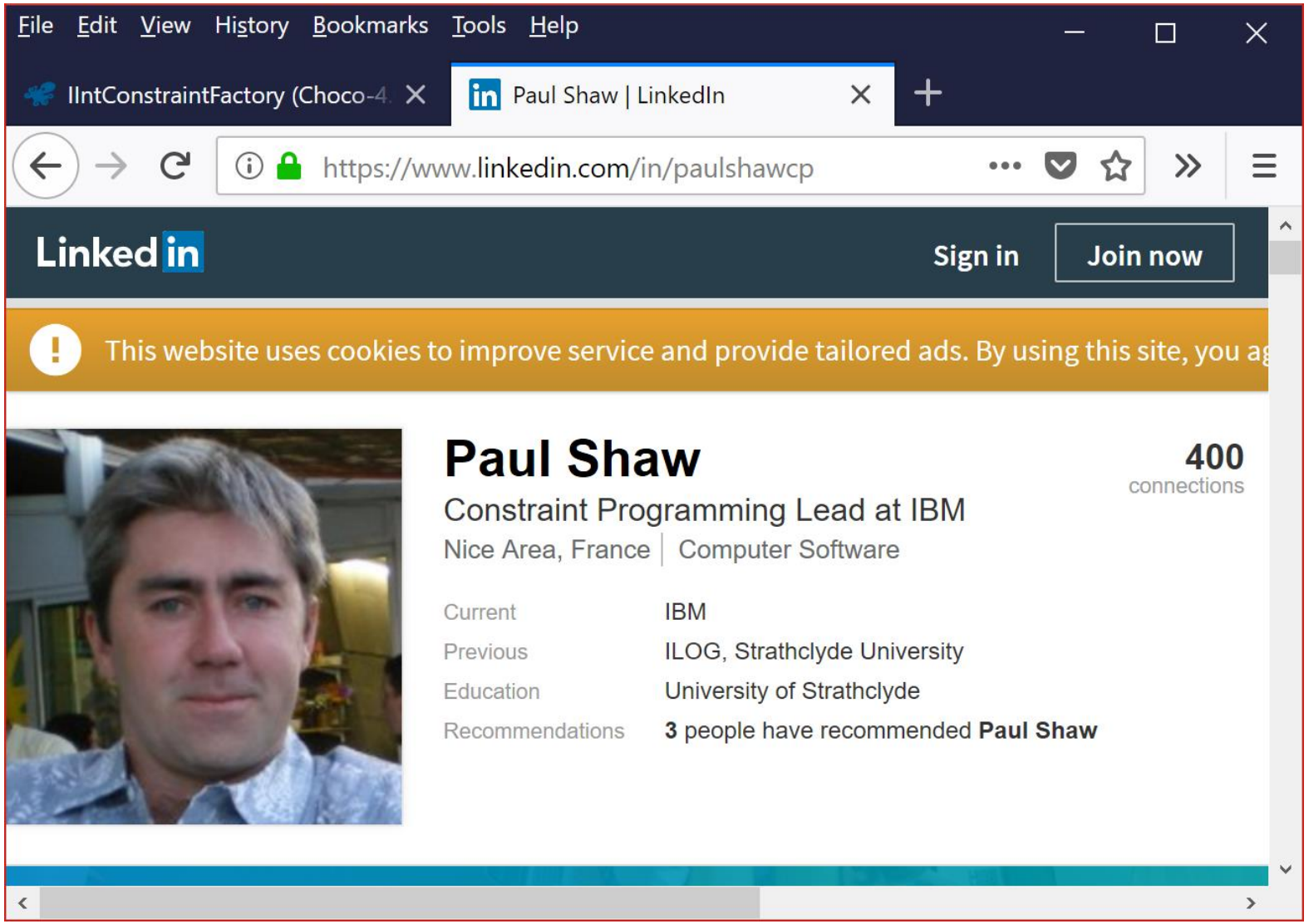
Is there another model?


File Edit View History Bookmarks Tools Help
IIntConstraintFactory (Choco-4
Bin packing problem - Wikiped
www.choco-solver.org/apidocs/index.html
Search
All Classes
Packages
org.chocosolver.memory
org.chocosolver.memory.structure
org.chocosolver.memory.trailing
org.chocosolver.memory.trailing.trail
org.chocosolver.memory.trailing.trail.chunc
LongWorld
MathUtils
MaxDelta
MaxRegret
MD
MDRk
Measures
MeasuresRecorder
Member
MinDelta
MinusView
Model
MonotonicRestartStrategy
Move
MoveBinaryDDS
MoveBinaryDFS
MoveBinaryHBFS
MoveBinaryLDS
MoveLearnBinaryTDR
MoveLNS
MoveRestart
MoveSeq
binPacking
default Constraint binPacking(IntVar[] itemBin,
int[] itemSize,
IntVar[] binLoad,
int offset)
Creates a BinPacking constraint. Bin Packing formulation: forall b in [0,binLoad.length-1], binLoad[b]=sum(itemSize[i] | i in [0,itemSize.length-1], itemBin[i] = b+offset forall i in [0,itemSize.length-1], itemBin is in [offset,binLoad.length-1+offset],
Parameters:
itemBin - IntVar representing the bin of each item
itemSize - int representing the size of each item
binLoad - IntVar representing the load of each bin (i.e. the sum of the size of the items in it)
offset - 0 by default but typically 1 if used within MiniZinc (which counts from 1 to n instead of from 0 to n-1)
boolsIntChanneling
default Constraint boolsIntChanneling(BoolVar[] bVars,
IntVar var,
int offset)
Creates an channeling constraint between an integer variable and a set of boolean variables. Maps the boolean assignments variables bVars with the standard assignment variable var.
var = i <-> bVars[i-offset] = 1


File Edit View History Bookmarks Tools Help
IIntConstraintFactory (Choco-4.
A Constraint for Bin Packing | S
https://link.springer.com/chapter/10.1
Search
Springer Link
Search
Menu
International Conference on Principles and Practice of Constraint Programming
CP 2004: Principles and Practice of Constraint Programming – CP 2004 pp 648-662 | Cite as
A Constraint for Bin Packing
Authors
Authors and affiliations
Paul Shaw
Conference paper
10
Citations
16
Readers
656
Downloads
Part of the Lecture Notes in Computer Science book series (LNCS, volume 3258)


File Edit View History Bookmarks Tools Help
IIntConstraintFactory (Choco-4.
A Constraint for Bin Packing | S
https://link.springer.com/chapter/10.1
Search
A Constraint for Bin Packing
Buy options
Abstract
We introduce a constraint for one-dimensional bin packing. This constraint uses propagation rules incorporating knapsack-based reasoning, as well as a lower bound on the number of bins needed. We show that this constraint can significantly reduce search on bin packing problems. We also demonstrate that when coupled with a standard bin packing search strategy, our constraint can be a competitive alternative to established operations research bin packing algorithms.
This is a preview of subscription content, log in to check access
Cite paper

Thanks me ol' mucka!



File Edit View History Bookmarks Tools Help

IIntConstraintFactory (Choco-4. | Paul Shaw - IBM

https://researcher.wats

Search

IBM

Marketplace

Search for people

IBM Research

Research areas | Work with us | About us | Blog

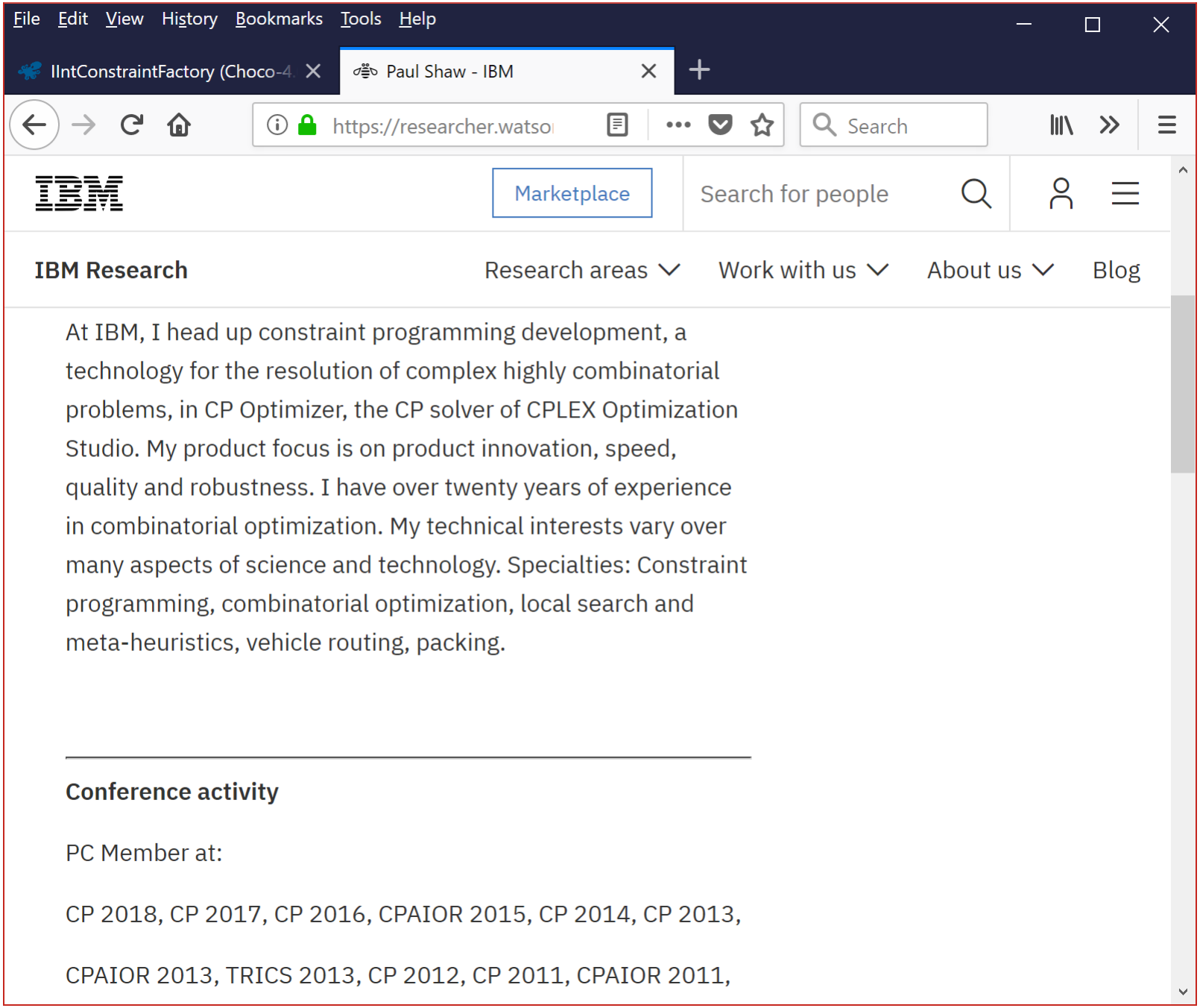
Paul Shaw

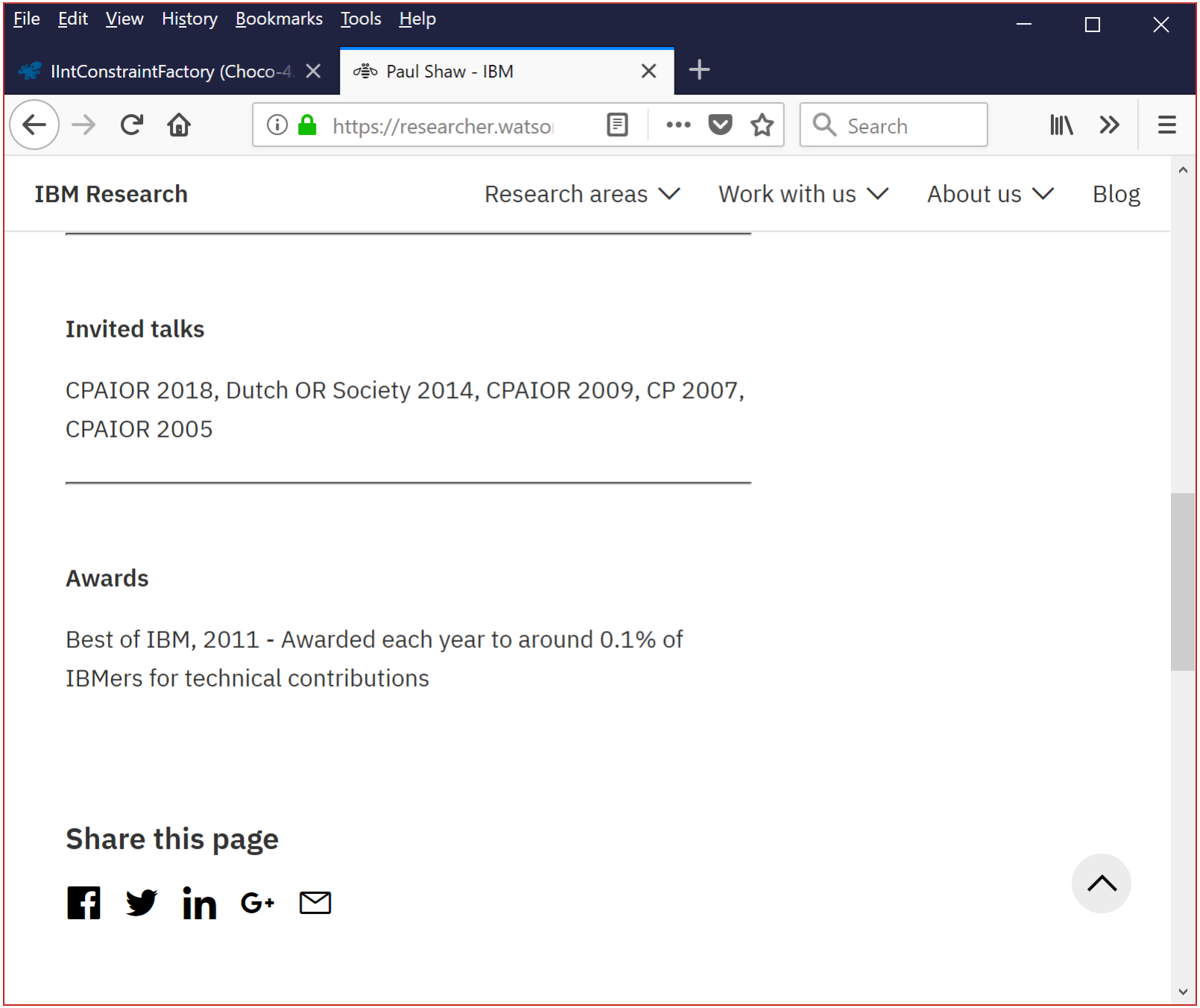
feedback

STSM, Optimization Technology. CP Optimizer Development Manager
paul.shaw@fr.ibm.com +33-4-9296-6225

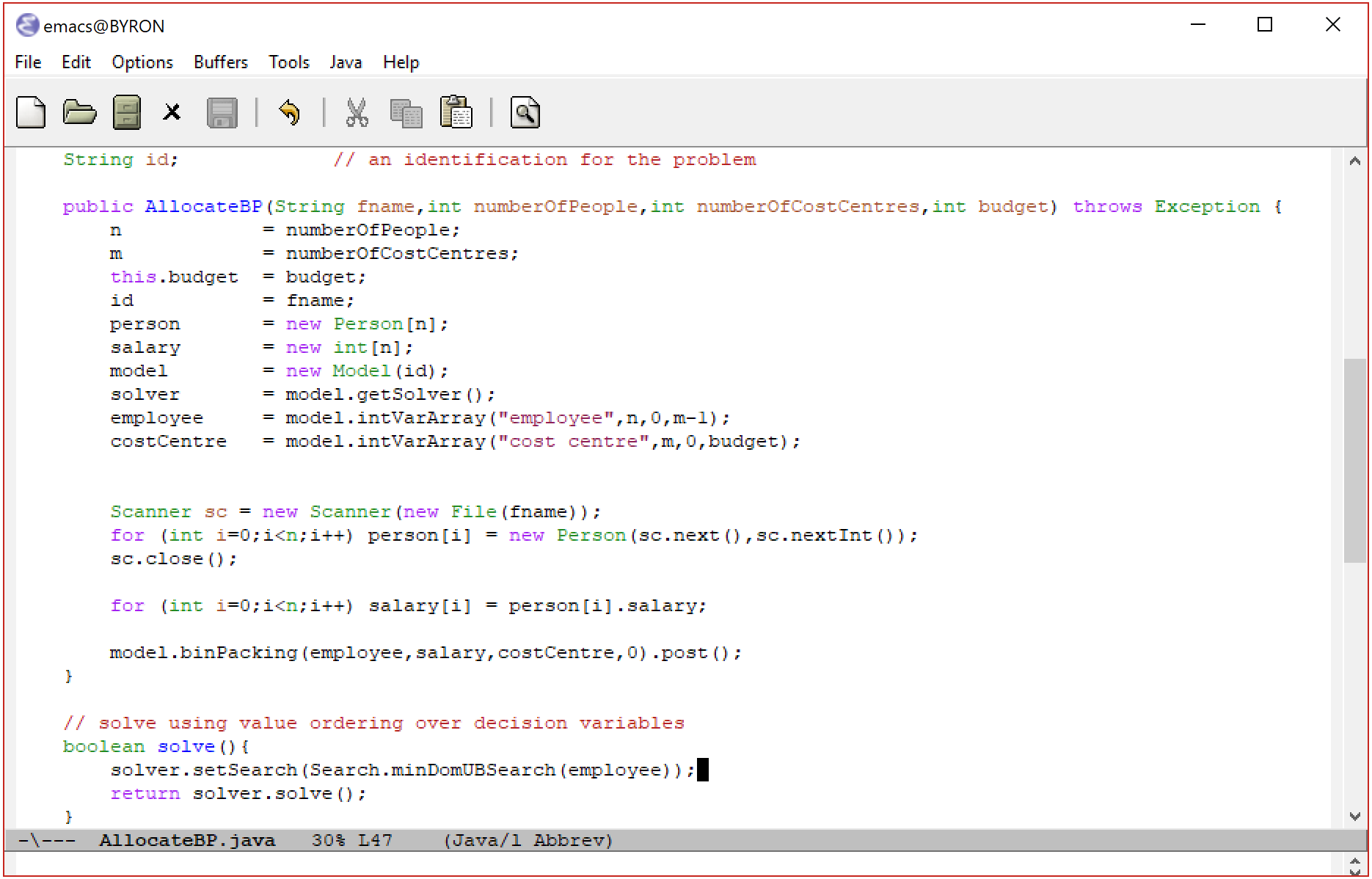
Profile | Publications | Patents

At IBM, I head up constraint programming development, a technology for the resolution of complex highly combinatorial problems, in CP Optimizer, the CP solver of CPLEX Optimization Studio. My product focus is on product innovation, speed


File Edit View History Bookmarks Tools Help
IIntConstraintFactory (Choco-4
Paul Shaw - IBM
https://researcher.watso
Search
IBM
Marketplace
Search for people
IBM Research
Research areas
Work with us
About us
Blog
At IBM, I head up constraint programming development, a technology for the resolution of complex highly combinatorial problems, in CP Optimizer, the CP solver of CPLEX Optimization Studio. My product focus is on product innovation, speed, quality and robustness. I have over twenty years of experience in combinatorial optimization. My technical interests vary over many aspects of science and technology. Specialties: Constraint programming, combinatorial optimization, local search and meta-heuristics, vehicle routing, packing.
Conference activity
PC Member at:
CP 2018, CP 2017, CP 2016, CPAIOR 2015, CP 2014, CP 2013,
CPAIOR 2013, TRICS 2013, CP 2012, CP 2011, CPAIOR 2011,


File Edit View History Bookmarks Tools Help
IIntConstraintFactory (Choco-4
Paul Shaw - IBM
https://researcher.watso
Search
IBM Research
Research areas
Work with us
About us
Blog
Invited talks
CPAIOR 2018, Dutch OR Society 2014, CPAIOR 2009, CP 2007, CPAIOR 2005
Awards
Best of IBM, 2011 - Awarded each year to around 0.1% of IBMers for technical contributions
Share this page

See AllocateBP



So?

What have we learned?

1. Identify the decision variables
2. What is value ordering doing to the search?
3. Can we use any heuristics?
4. Are there symmetries that we can break?
5. Are there any simple/redundant tests/constraints overlooked?
6. Is there an alternative model?

- How would we modify our model to address
  - Two people must be in same cost centre
  - Two people must not be in same cost centre
  - Cost centres have a limit on
    - Sum of salaries and …
    - Number of employees in cost centre

end