JUDCon

JBoss Users & Developers Conference

London:2011

Bin packing with Drools Planner Scheduling processes on cloud servers

Geoffrey De Smet

Every organization has planning problems.

What is a planning problem?

- Complete goals
- With limited resources
- Under constraints

Hospital bed scheduling

http://www.flickr.com/photos/markhillary/2227726759/

- Assign each
  - Patient
- To
  - Bed
- Constraints
  - Length of stay
  - Room requirements
  - Room preferences

Hospital nurse rostering

http://www.flickr.com/photos/glenpooh/709704564/

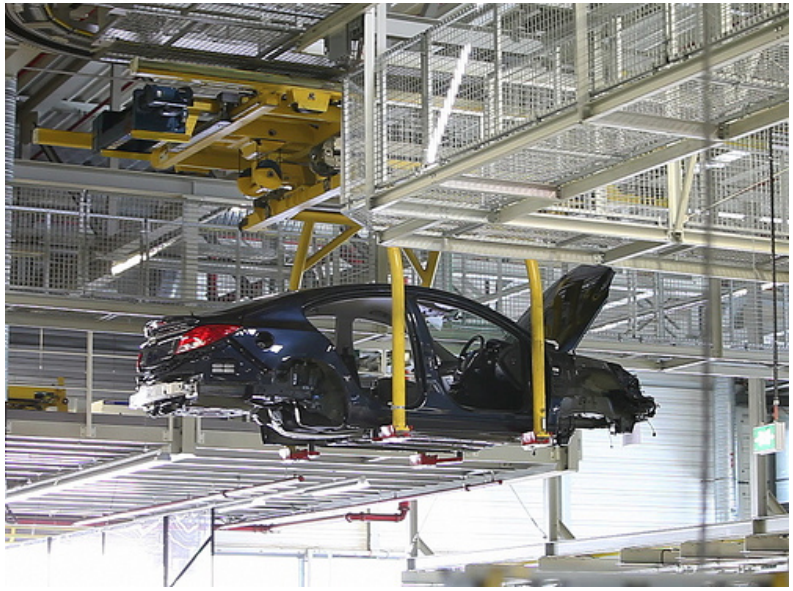
- Assign each
  - Shift
- To
  - Nurse
- Constraints
  - Employment contract
  - Free time preferences
  - Skills

School timetabling

http://www.flickr.com/photos/phelyan/2281095105/

- Assign each
  - Course
- To
  - Room
  - Timeslot
- Constraints
  - No simultaneous
    - Per room
    - Per teacher
    - Per student

Car factory order scheduling



http://www.flickr.com/photos/52248755@N03/4816681486/

- Assign each
  - Car order
- To
  - Assembly line
- Constraints
  - Assembly line specialisation

Airline routing

http://www.flickr.com/photos/yorickr/3674349657/

- Assign each
  - Flight
- To
  - Airplane
  - Crew
- Constraints
  - Airplane/crew depart from where they arrive
  - Minimize mileage

Bin packing in the cloud

http://www.flickr.com/photos/torkildr/3462607995/

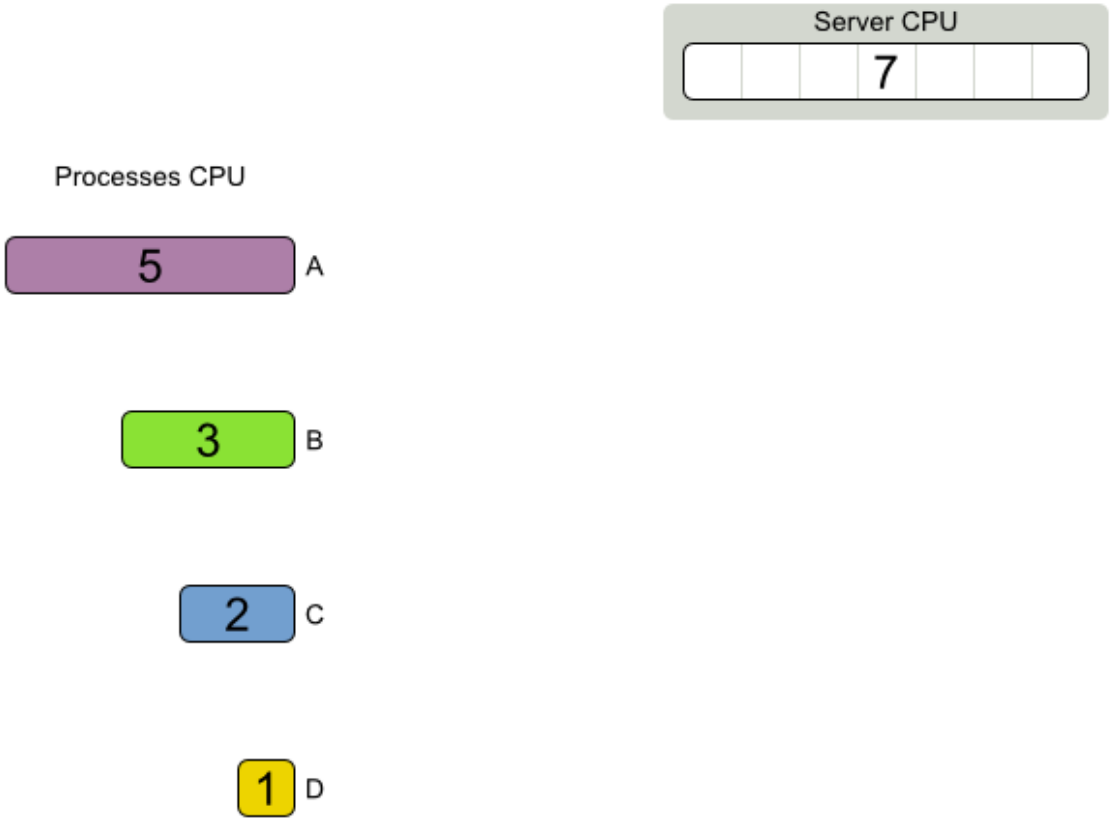
- Assign each
  - Process
- To
  - Server
- Constraints
  - Hardware requirements
  - Minimize server cost

Bin packing in the cloud

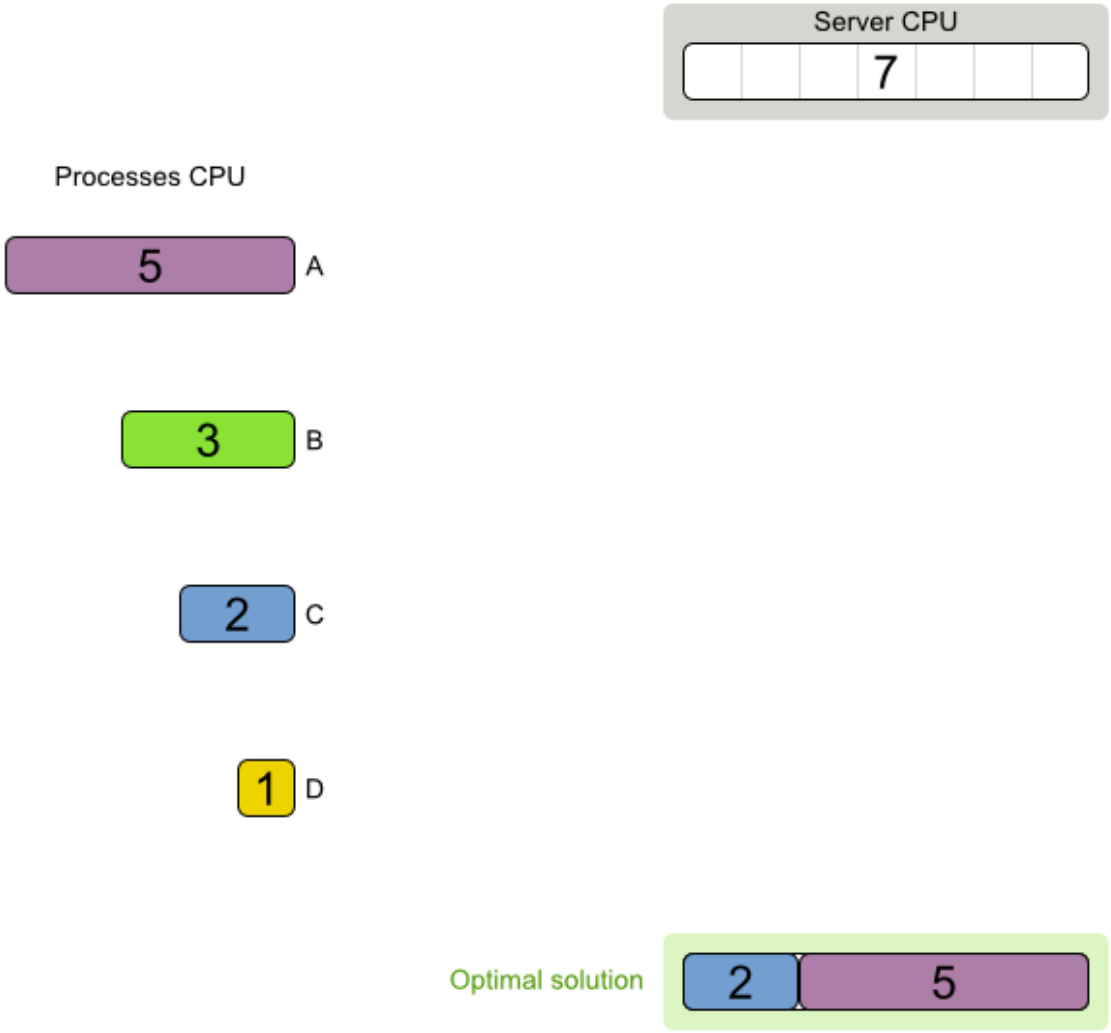
http://www.flickr.com/photos/torkildr/3462607995/

- Assign each
  - Process
- To
  - Server
- Constraints
  - Hardware requirements
  - Minimize server cost

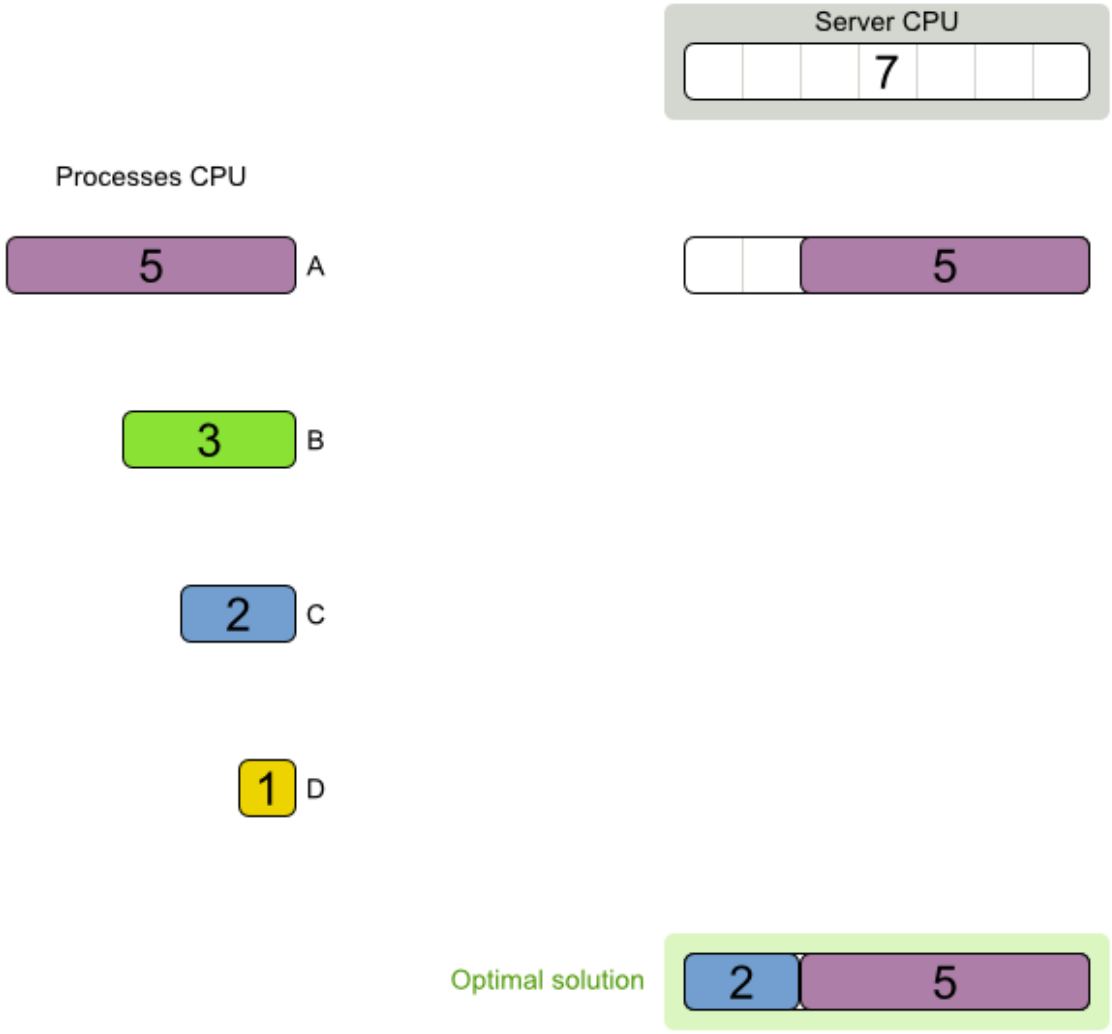
Which processes do we assign to this server?



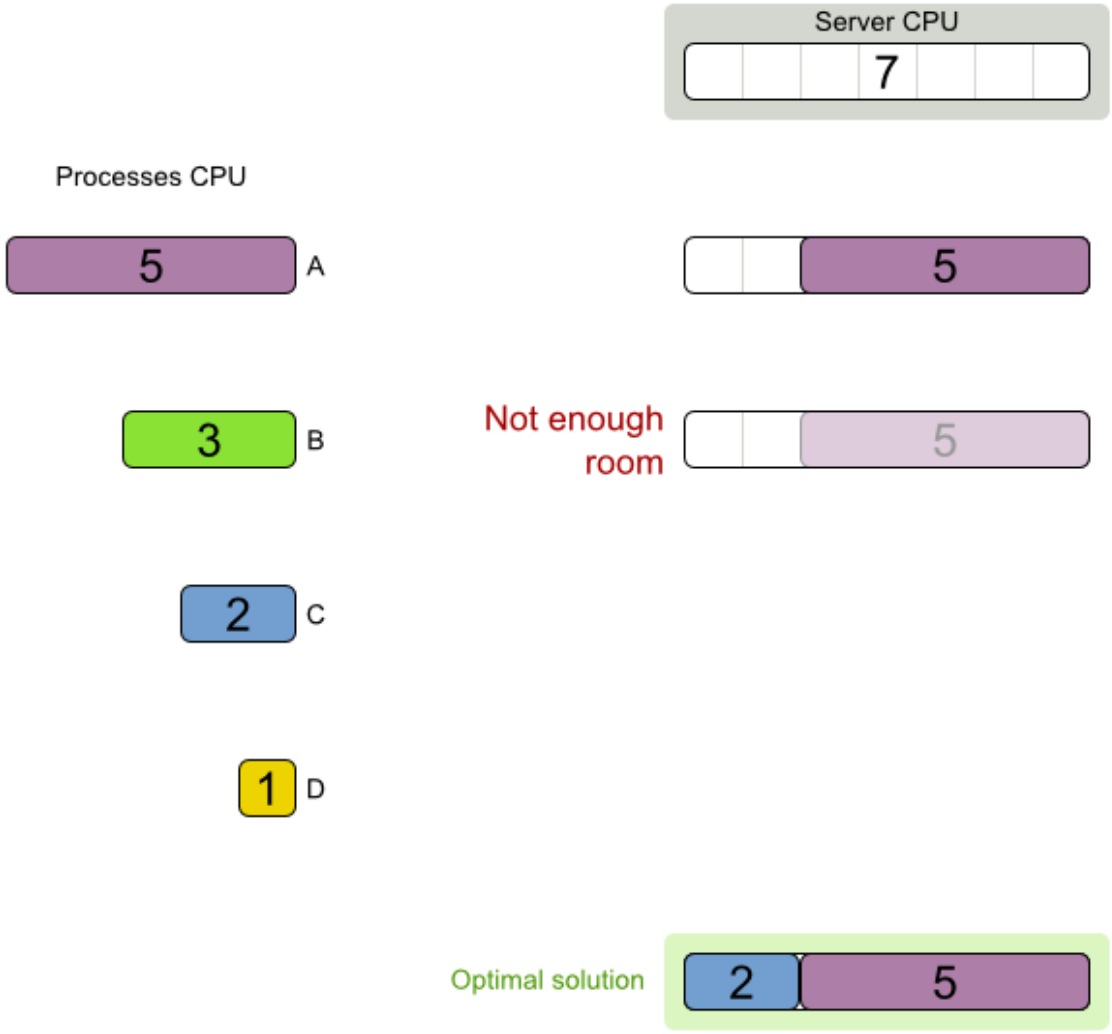
How did we find that solution?



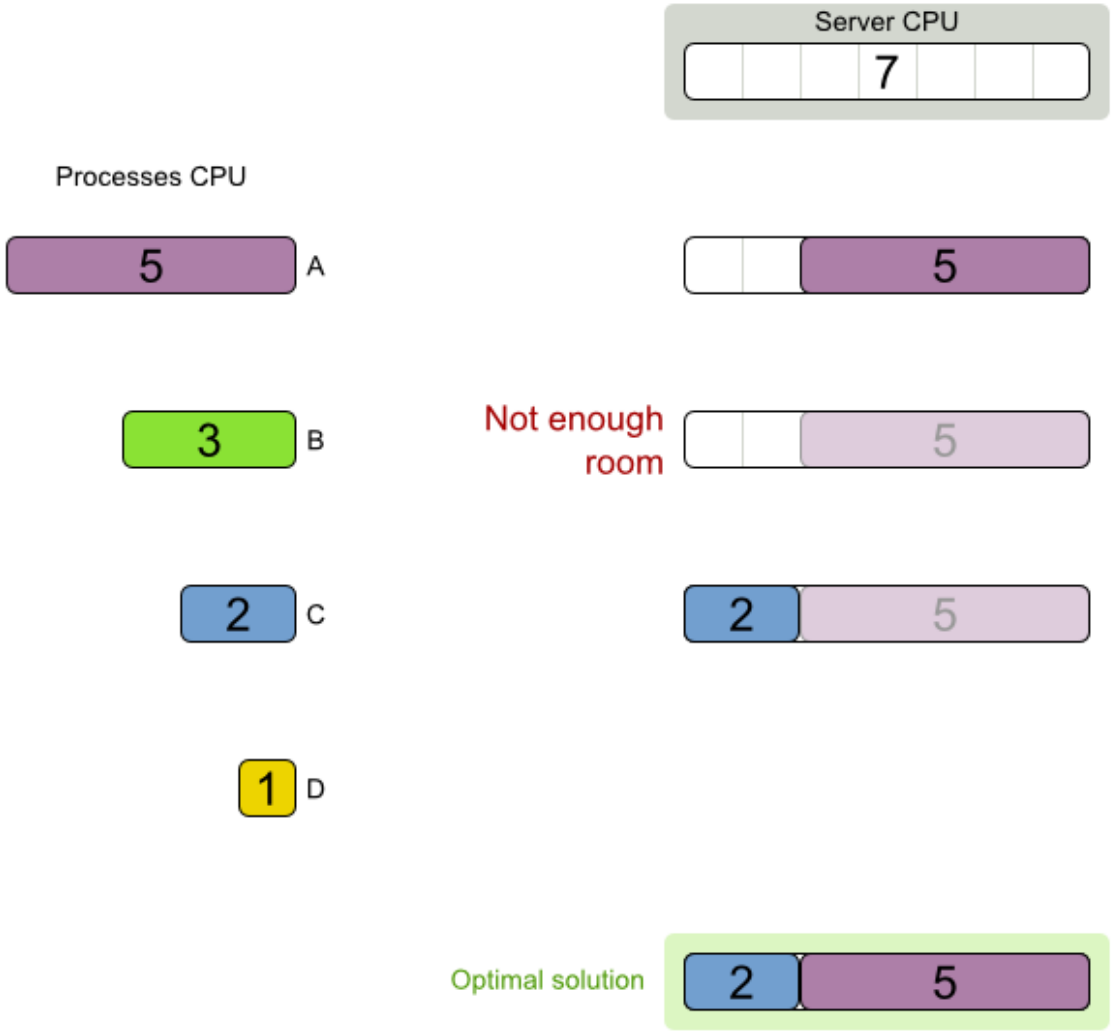
First fit by decreasing size



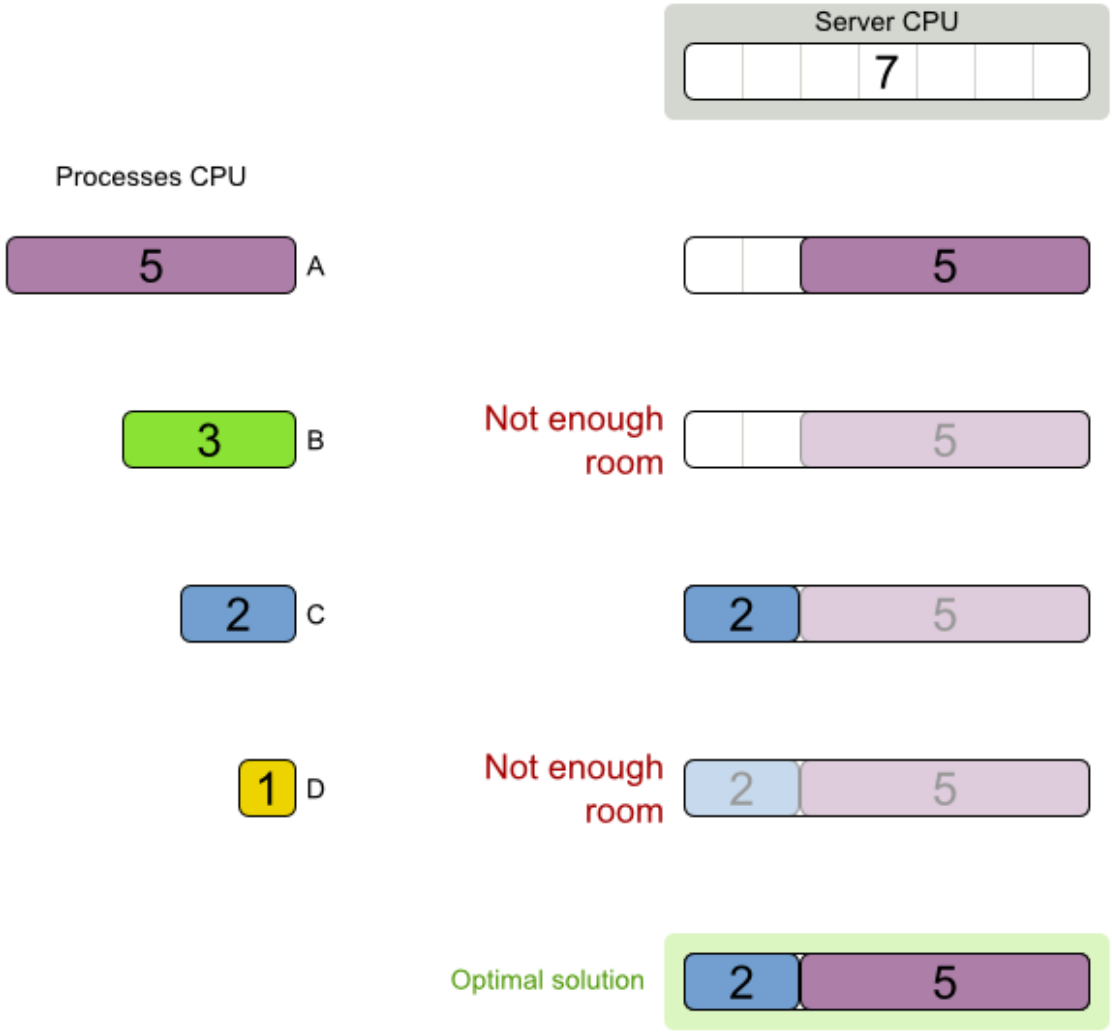
First fit by decreasing size



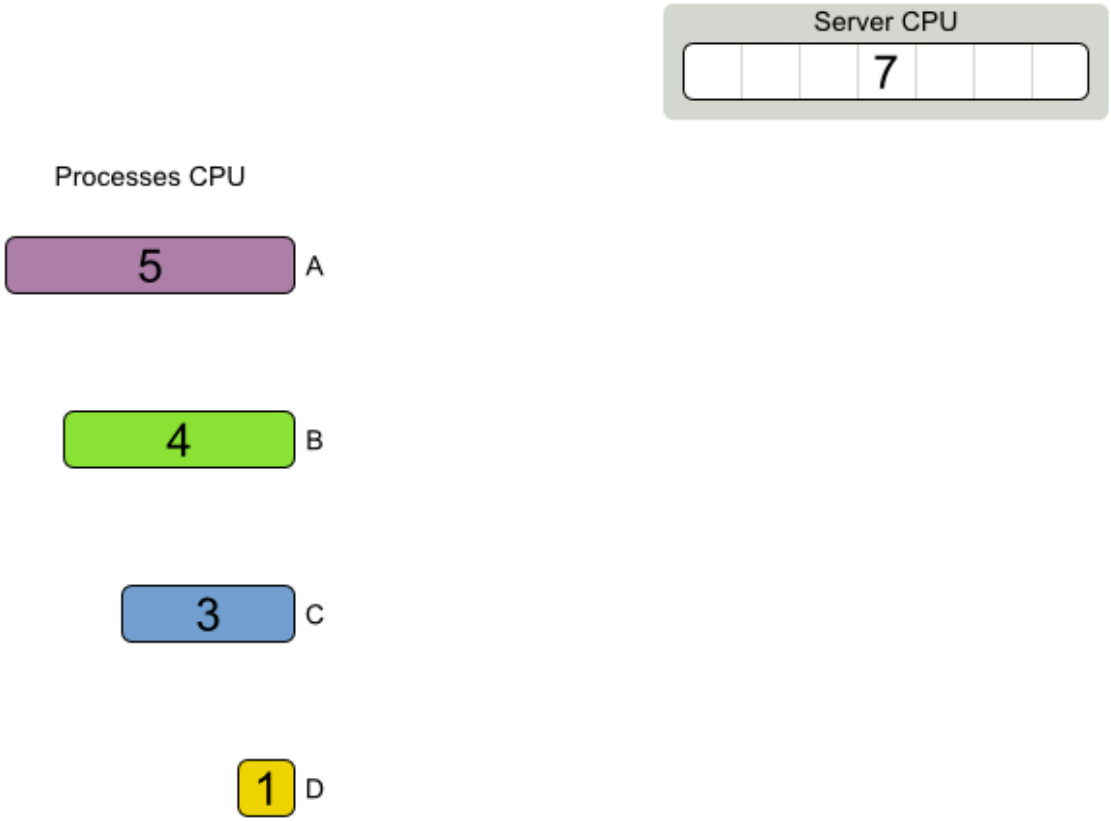
First fit by decreasing size



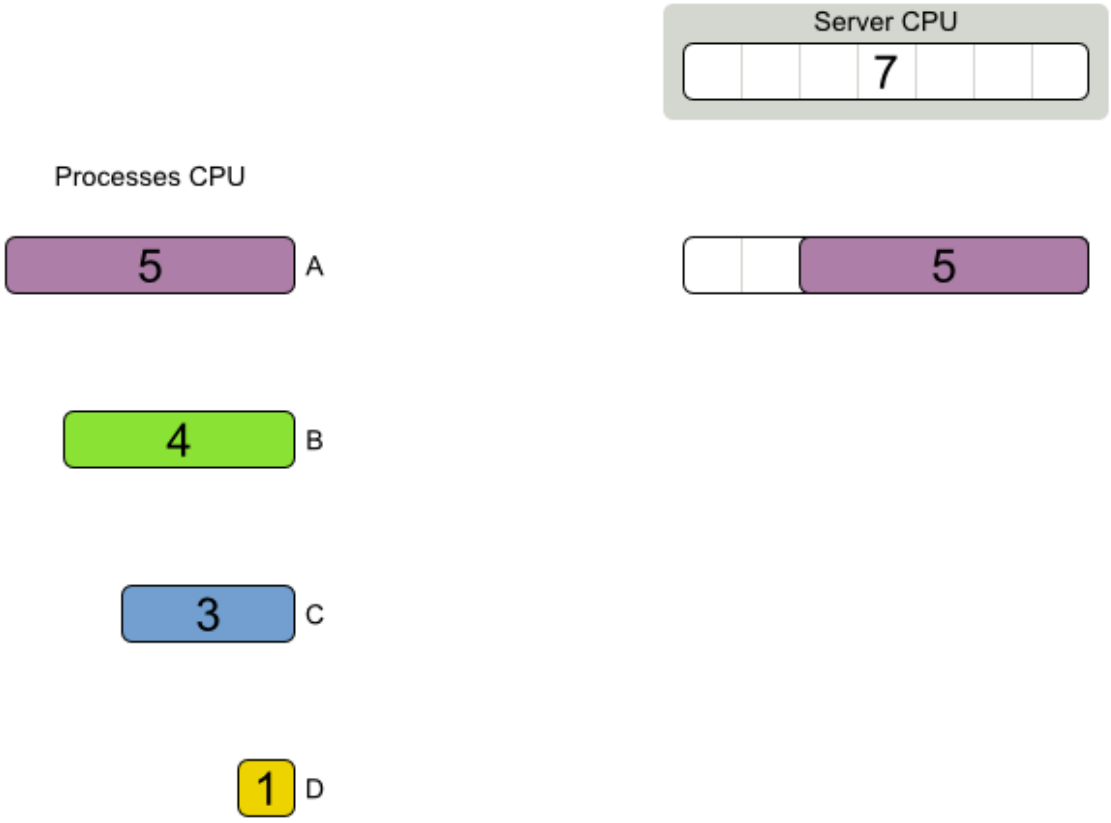
First fit by decreasing size



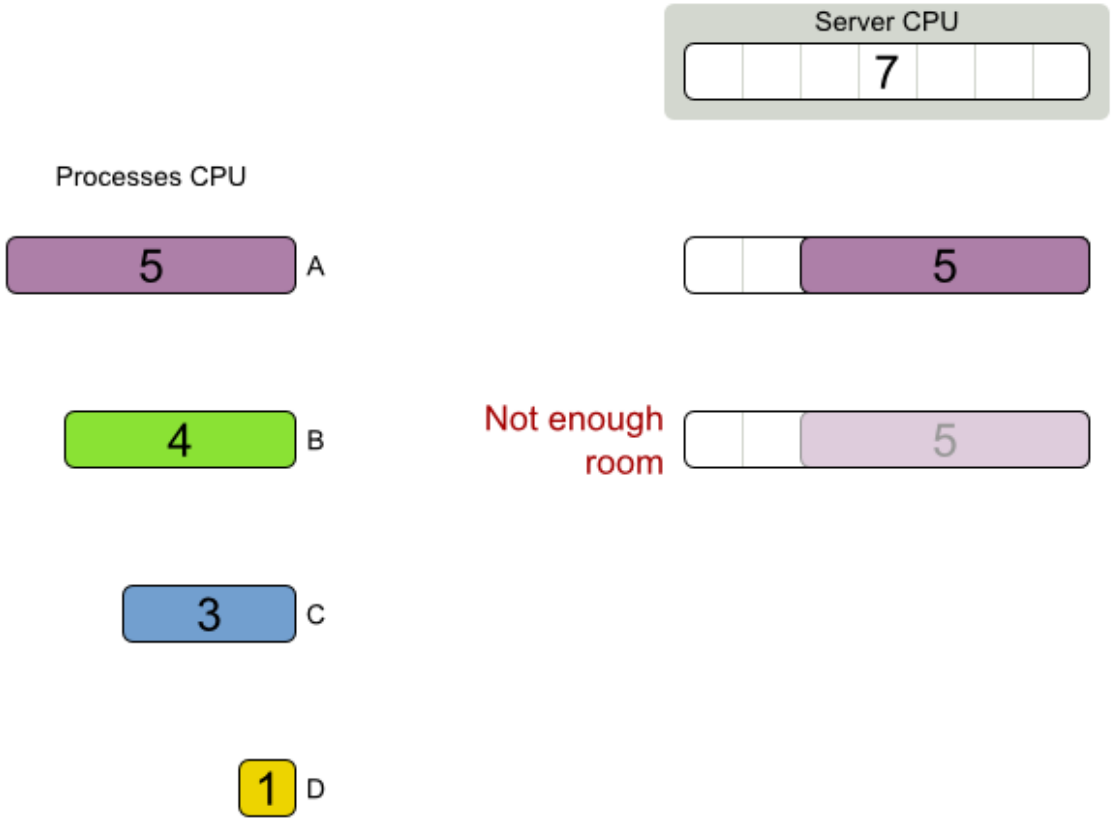
Another case



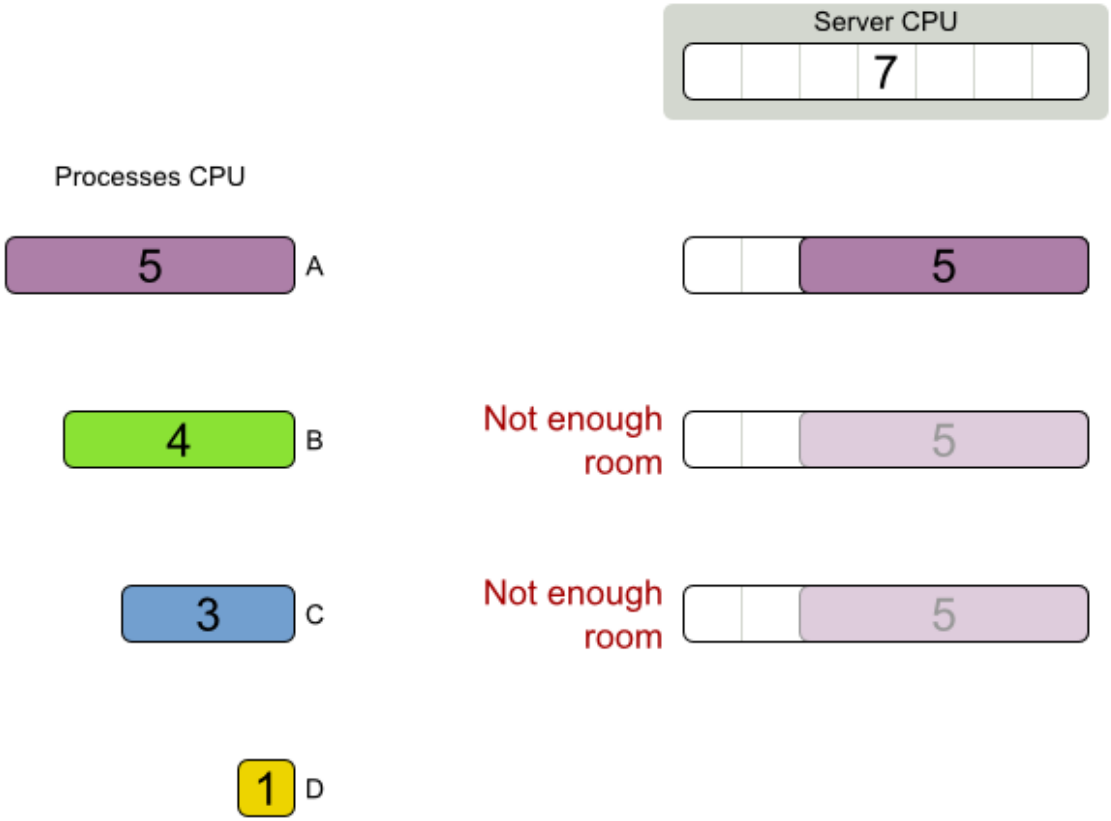
Try FFD again



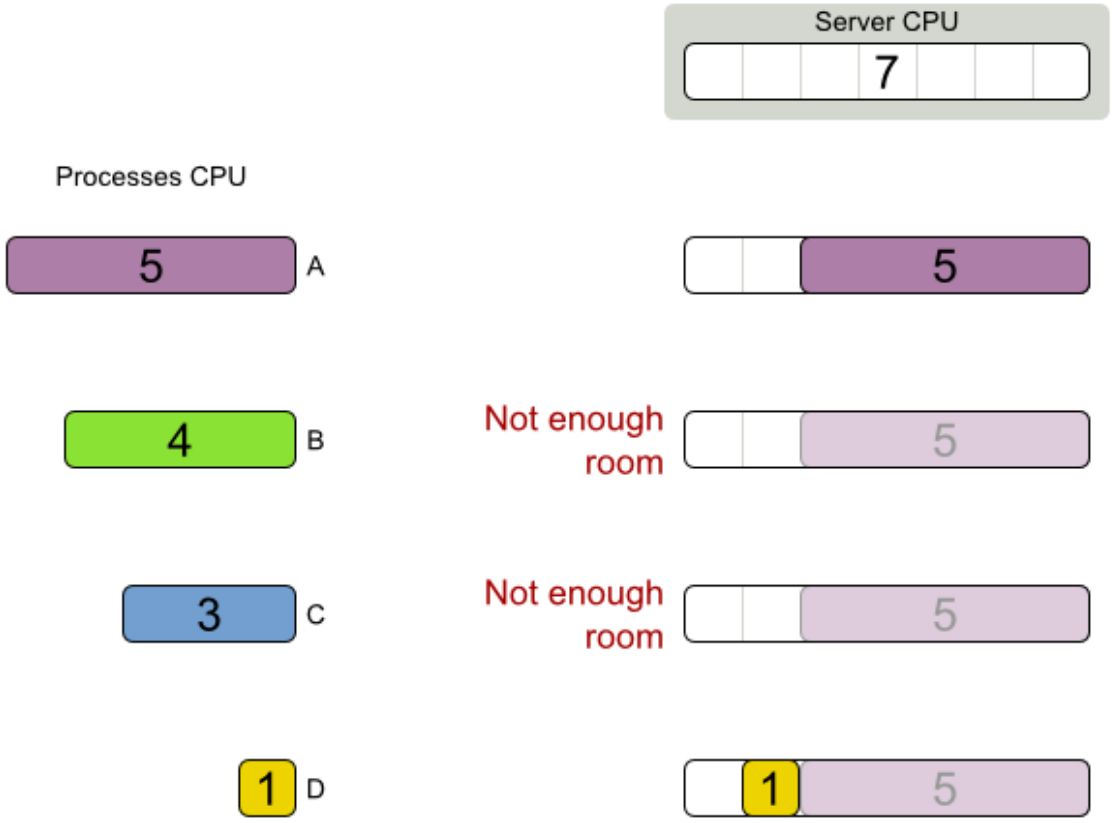
Try FFD again



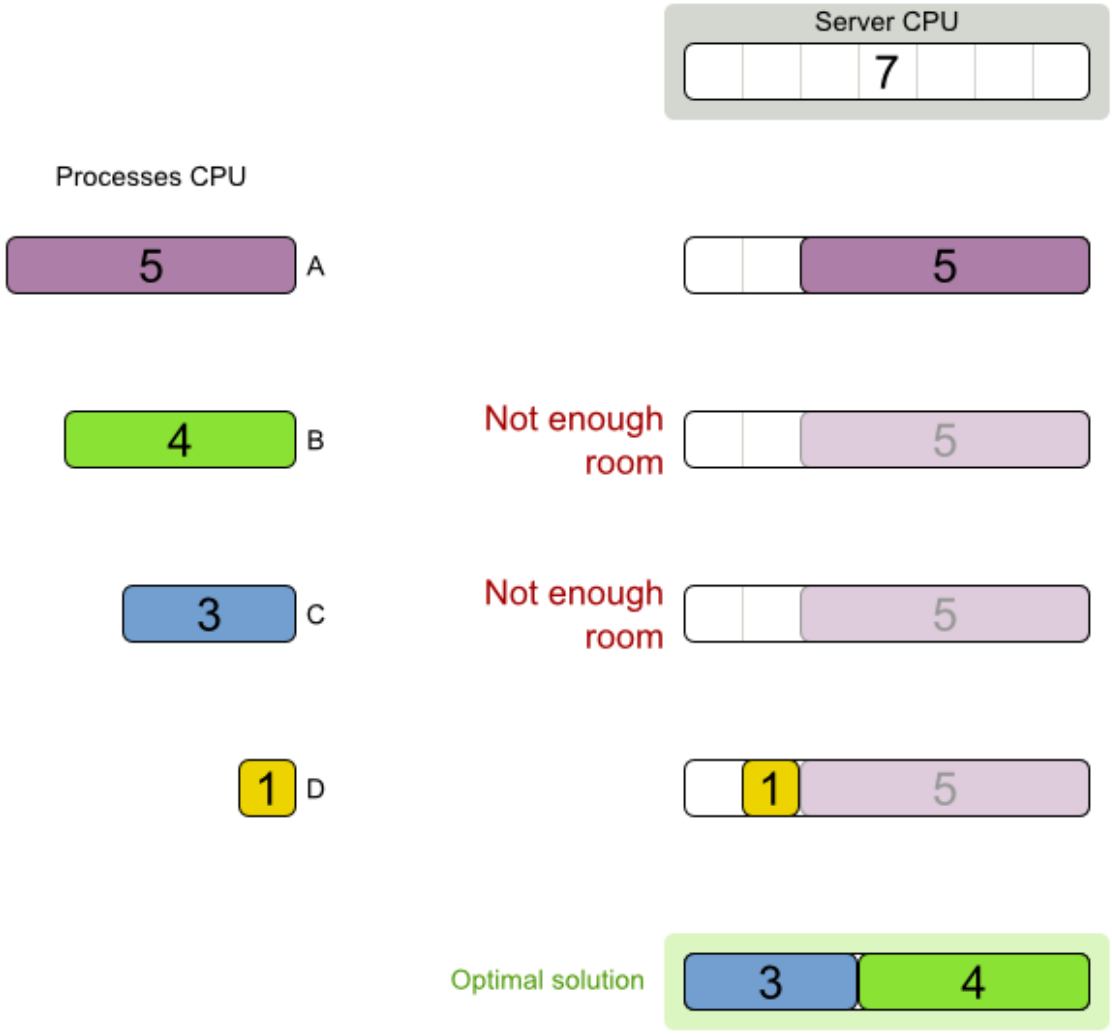
Try FFD again



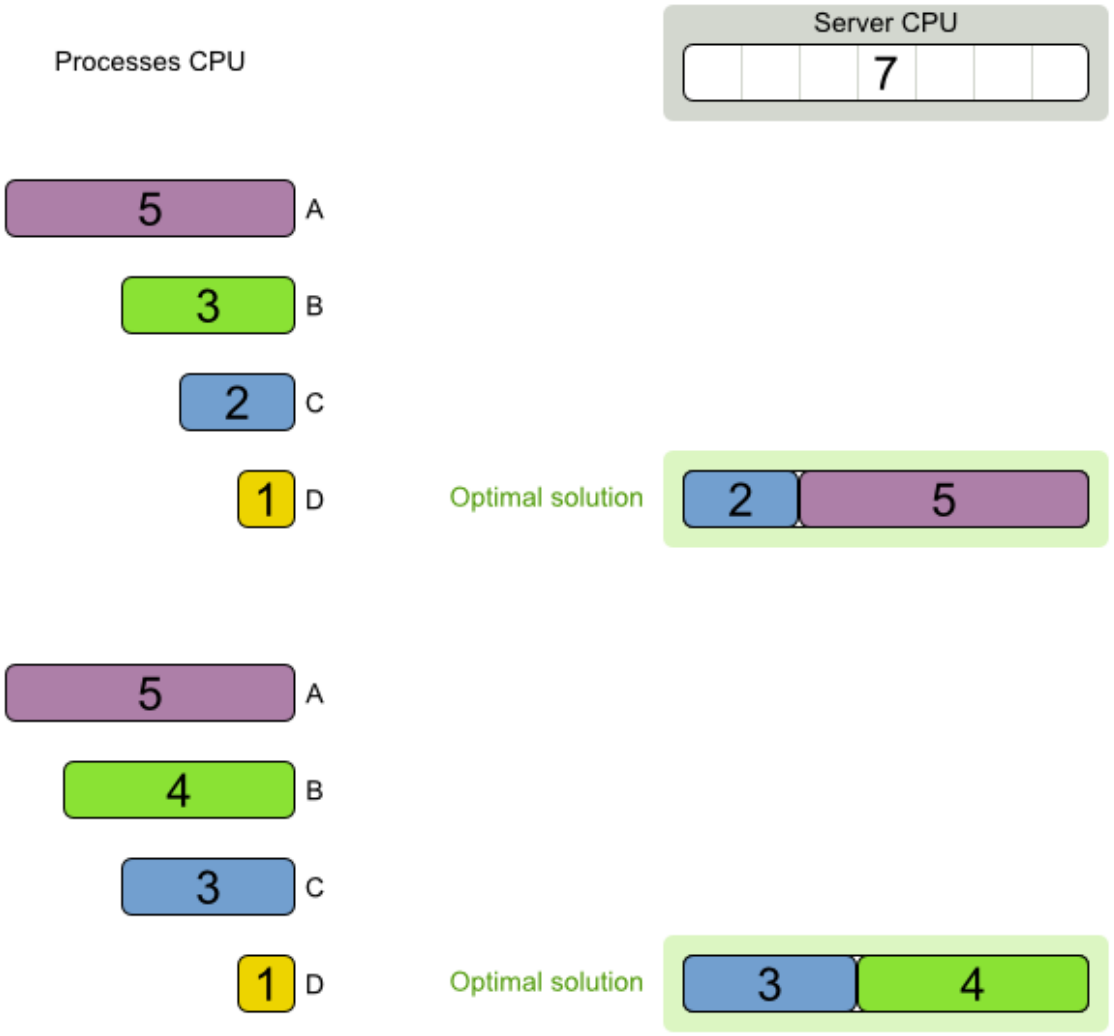
Try FFD again



FFD failure



NP complete


Processes CPU
Server CPU
7
5
A
3
B
2
C
1
D
Optimal solution
2
5
5
A
4
B
3
C
1
D
Optimal solution
3
4

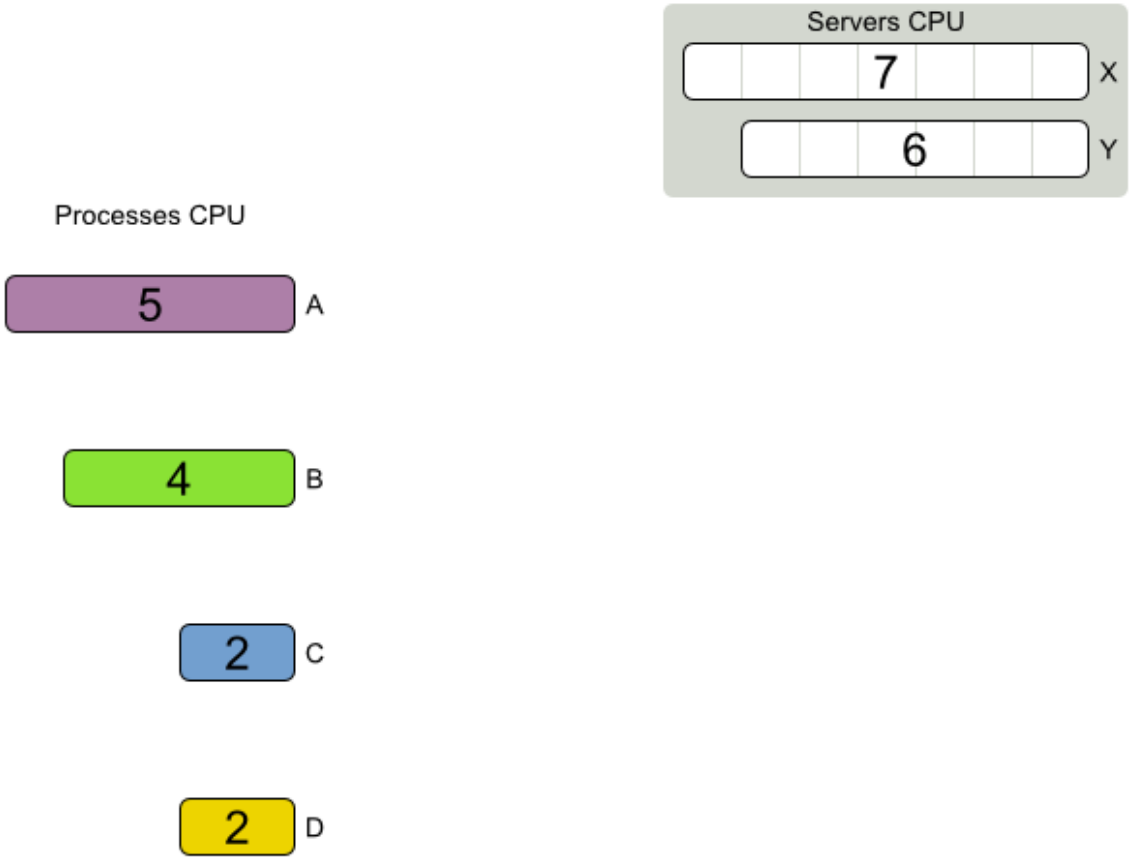
NP complete

http://www.flickr.com/photos/annguyenphotography/3267723713/

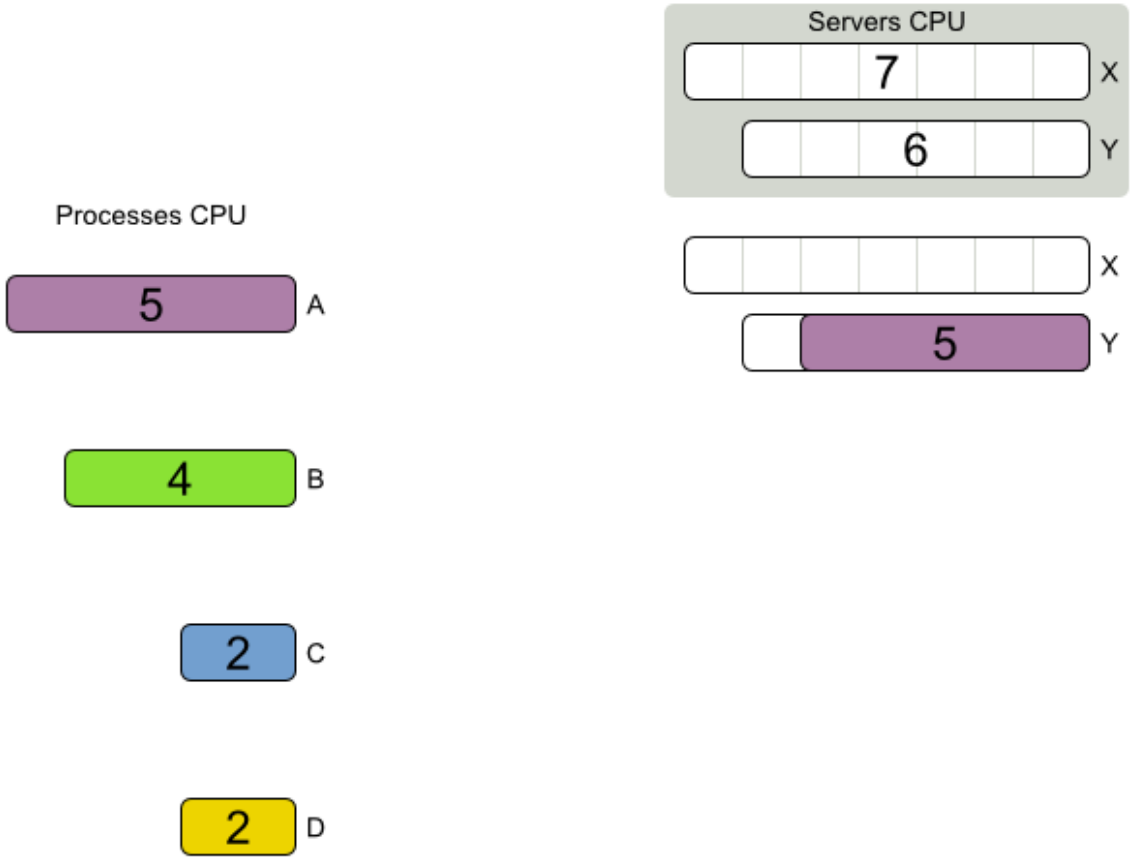
- No silver bullet known
  - Holy grail of computer science (P == NP)
  - Probably does not exist (P != NP)
- Root problem of all planning problems

Organizations rarely optimize planning problems.

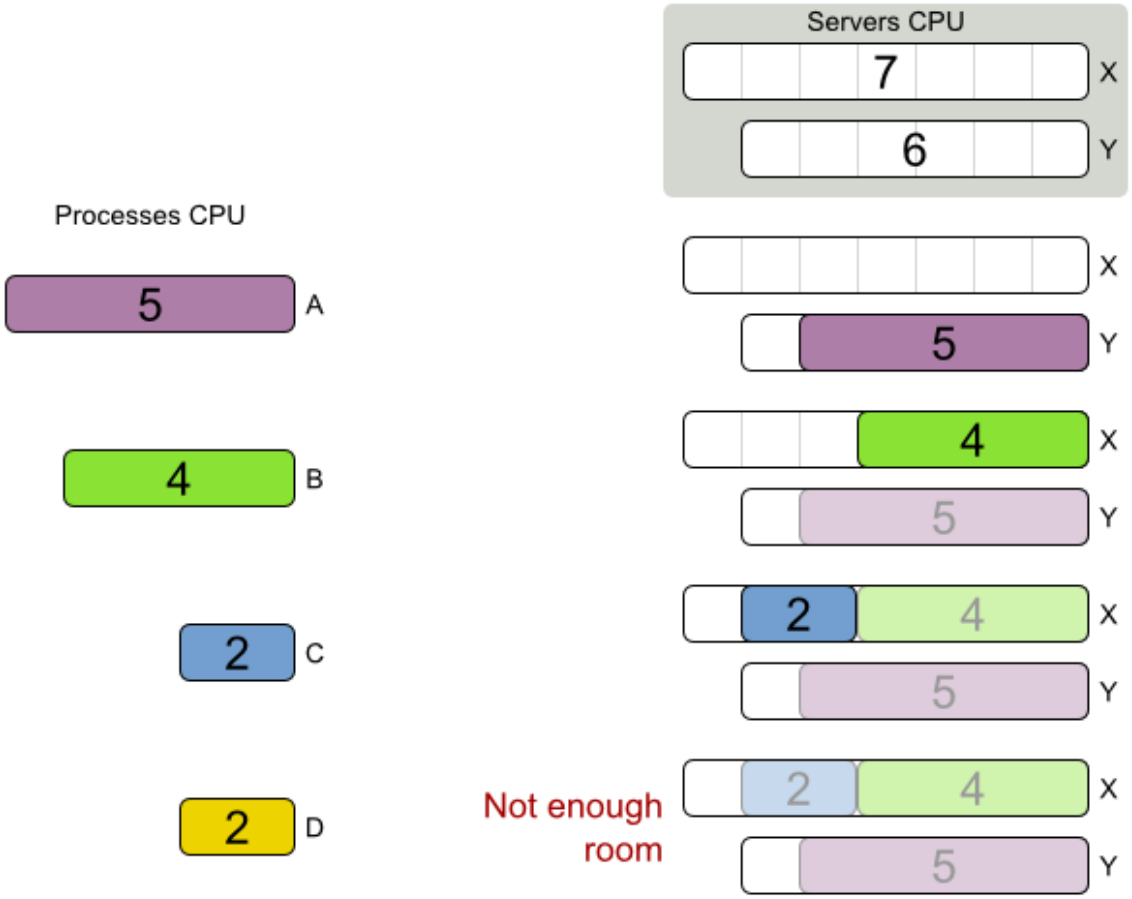
Multiple servers...



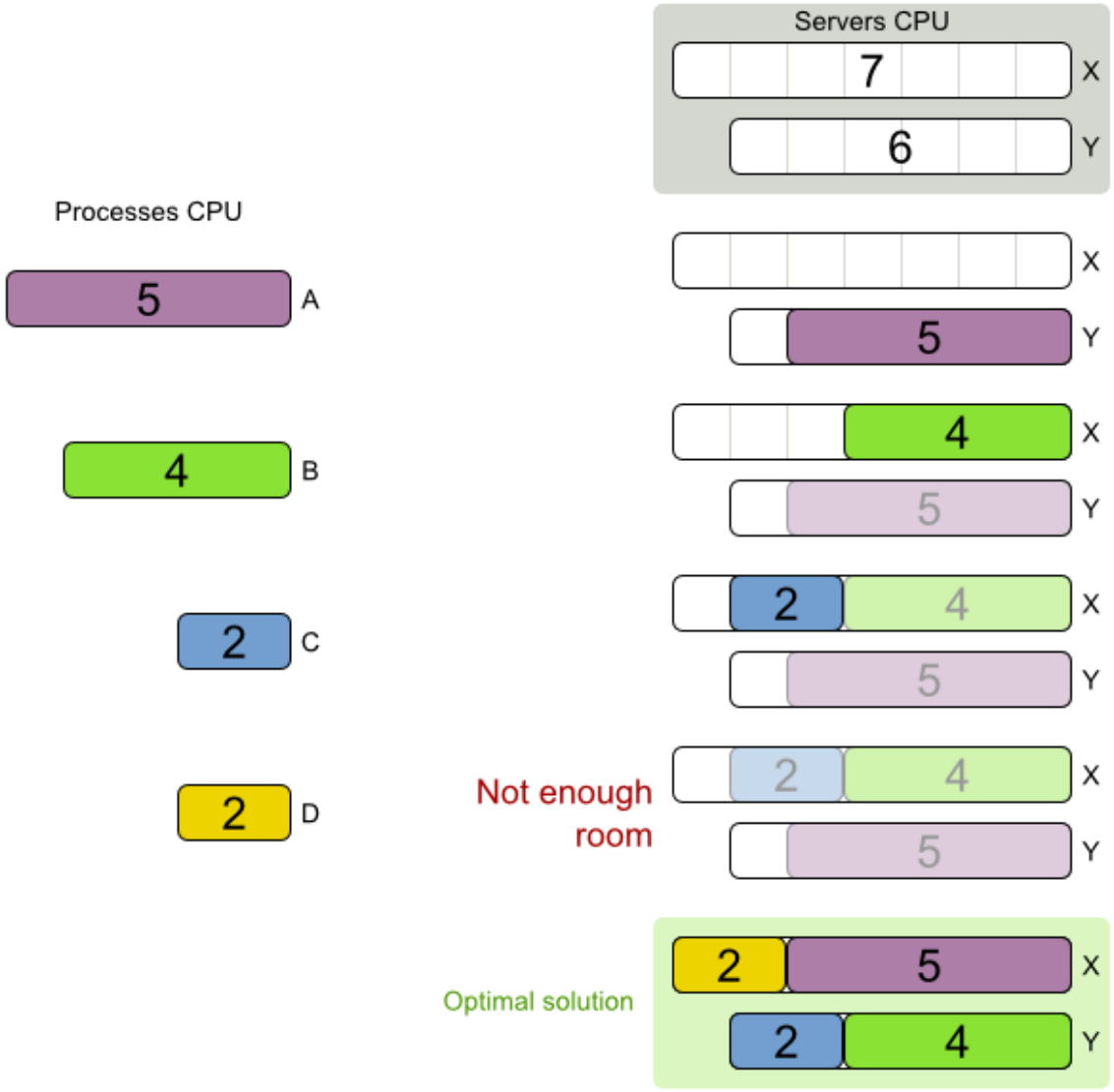
Multiple servers...



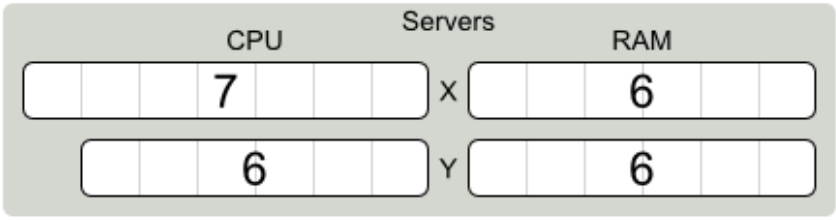
Multiple servers...

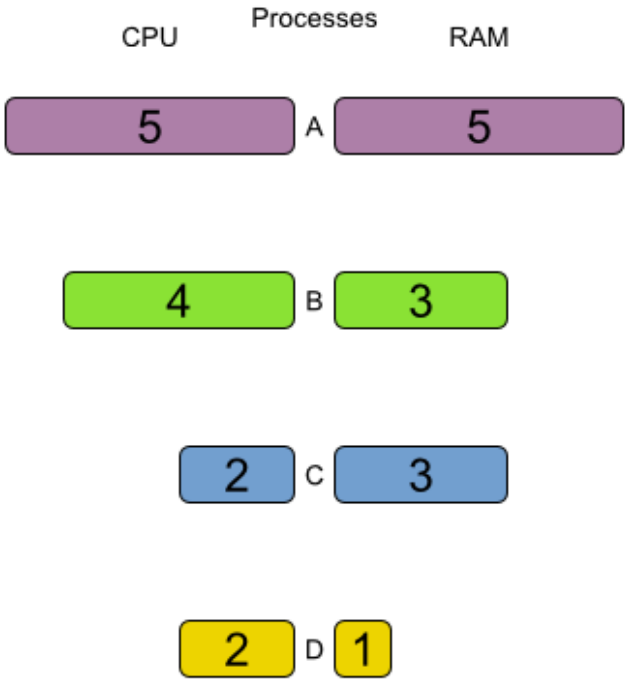


Multiple servers...

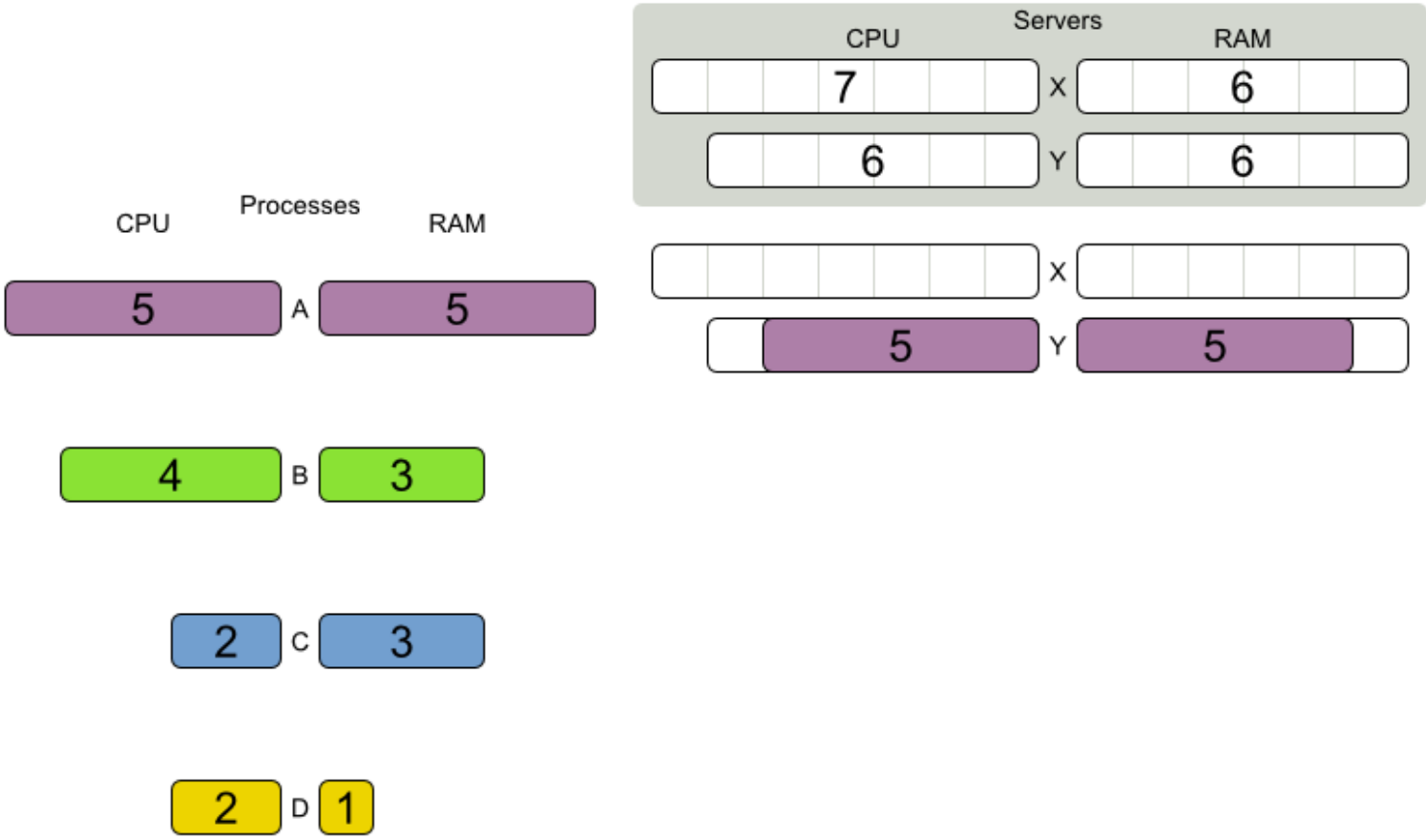


Multiple constraints...

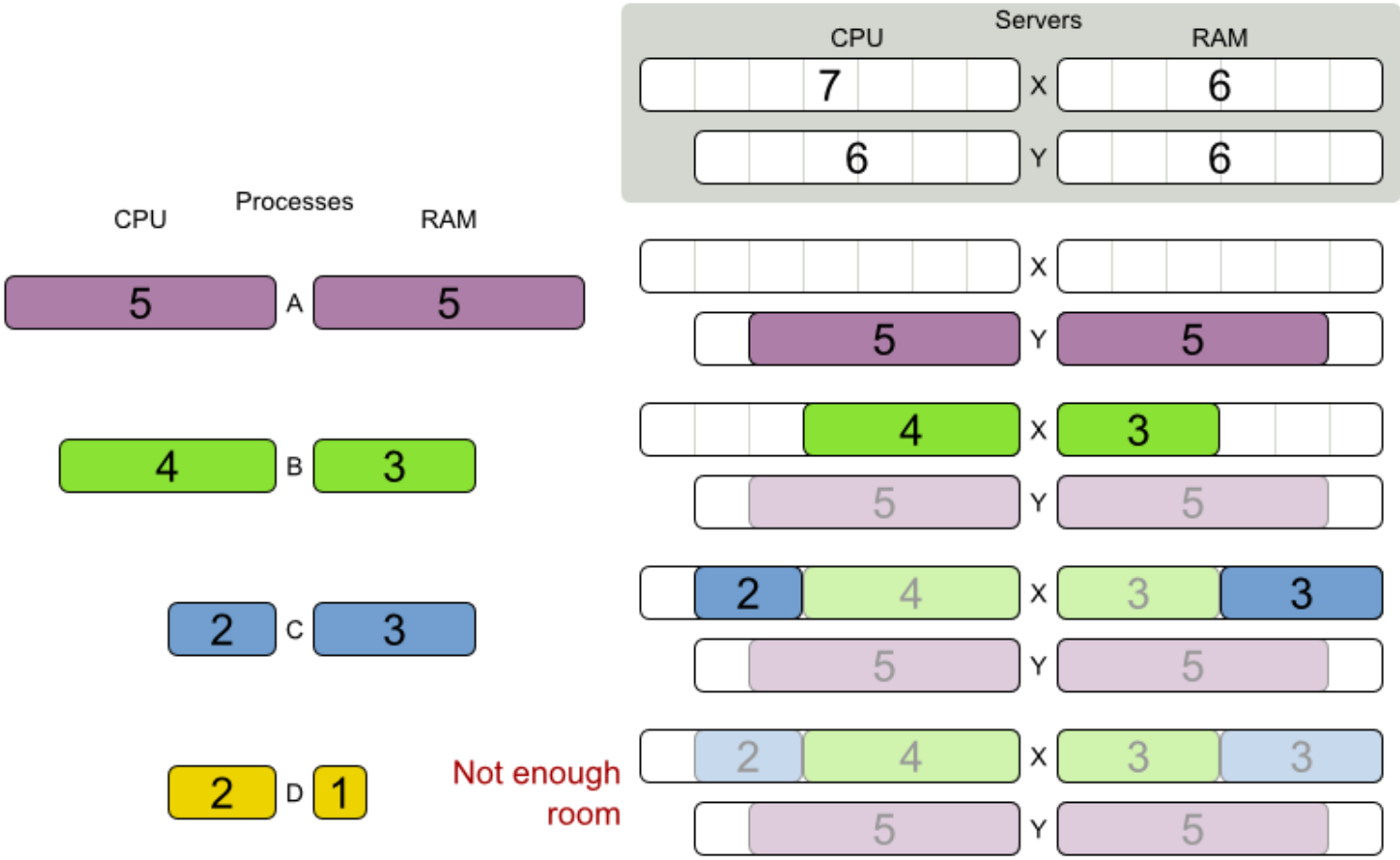




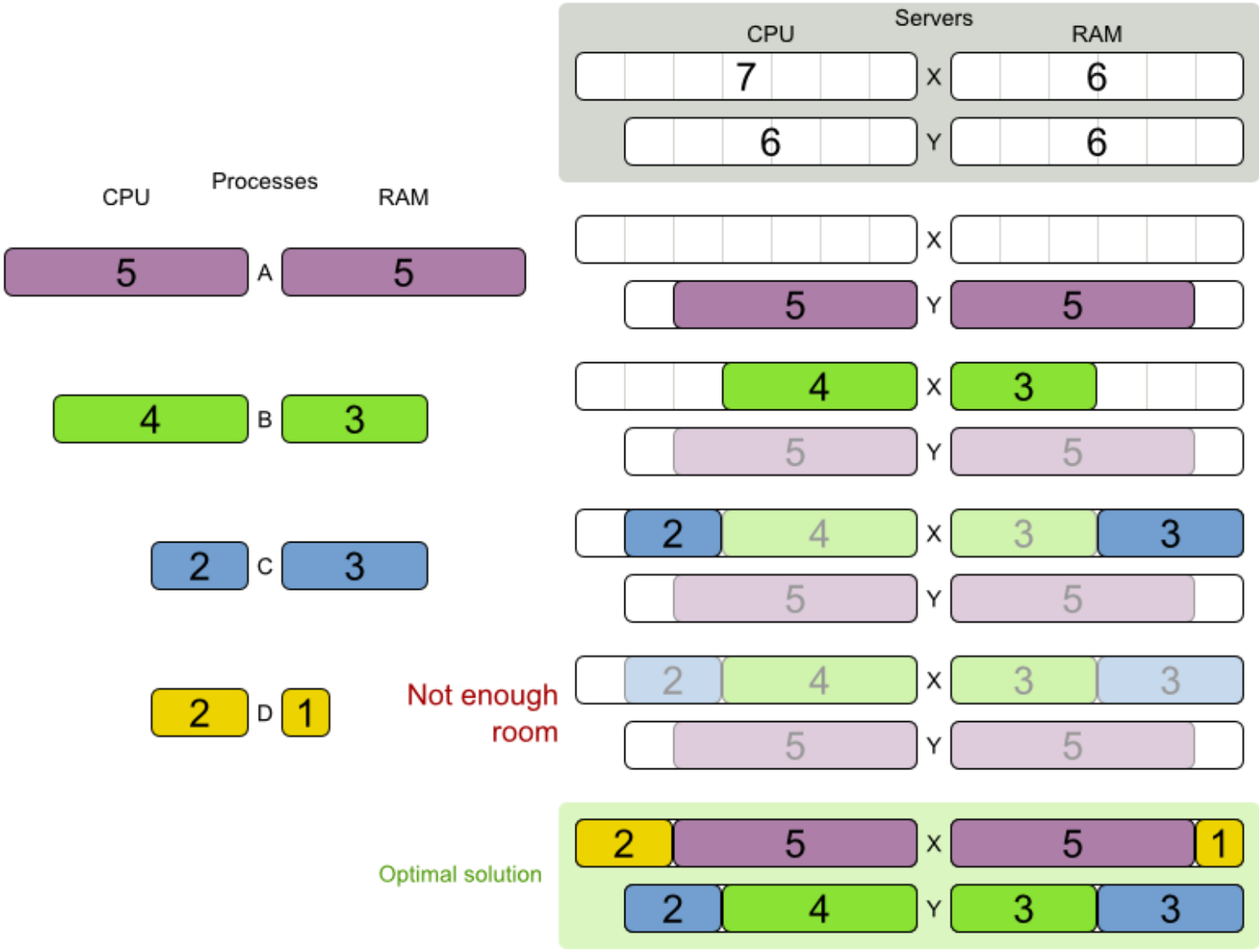
Multiple constraints...



Multiple constraints...



Multiple constraints...



Reuse optimization algorithms?

Drools Planner

- Optimization algorithms for normal Java programmers
- Open source (ASL 2.0)
- Manual + examples

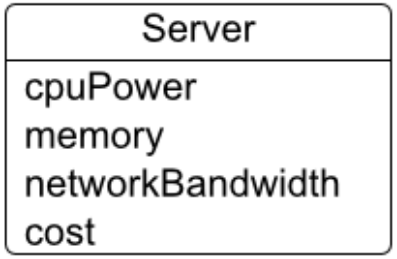
Demo

Drools Planner

CloudBalance example

Domain model

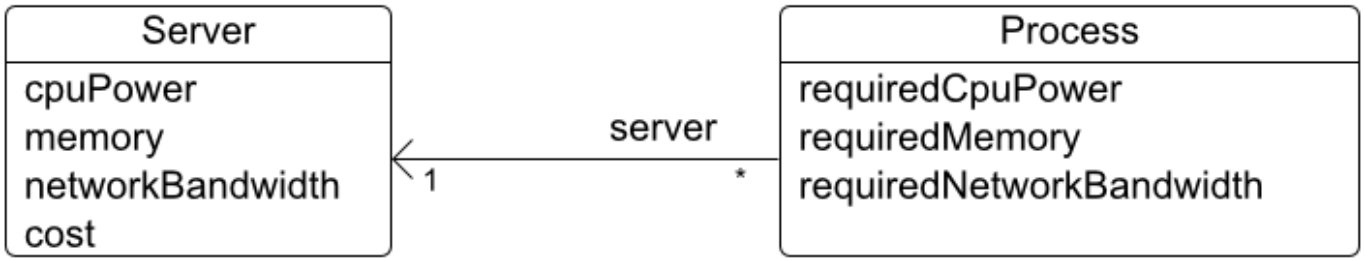
Server



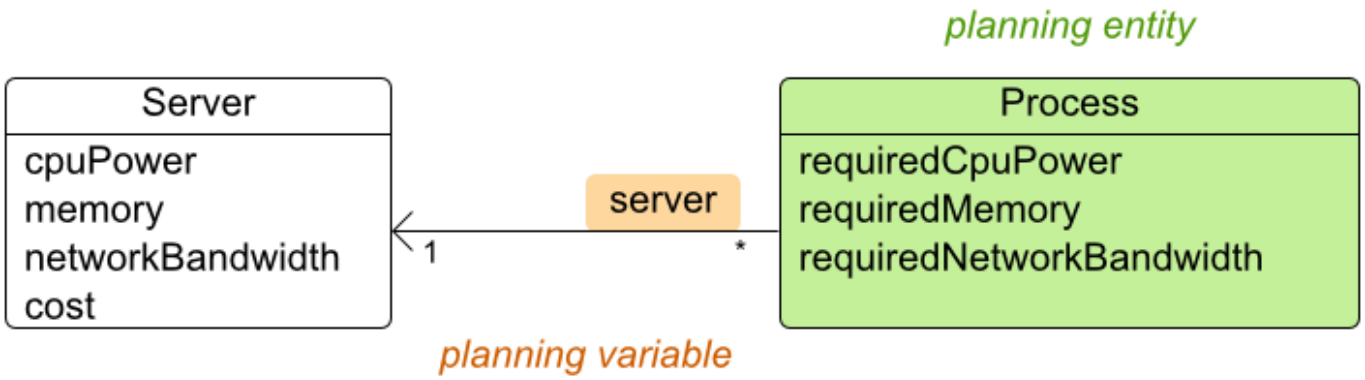
Server

```
public class Server {

   private int cpuPower;
   private int memory;
   private int networkBandwidth;

   private int cost;

   // getters
}
```

Process



Process



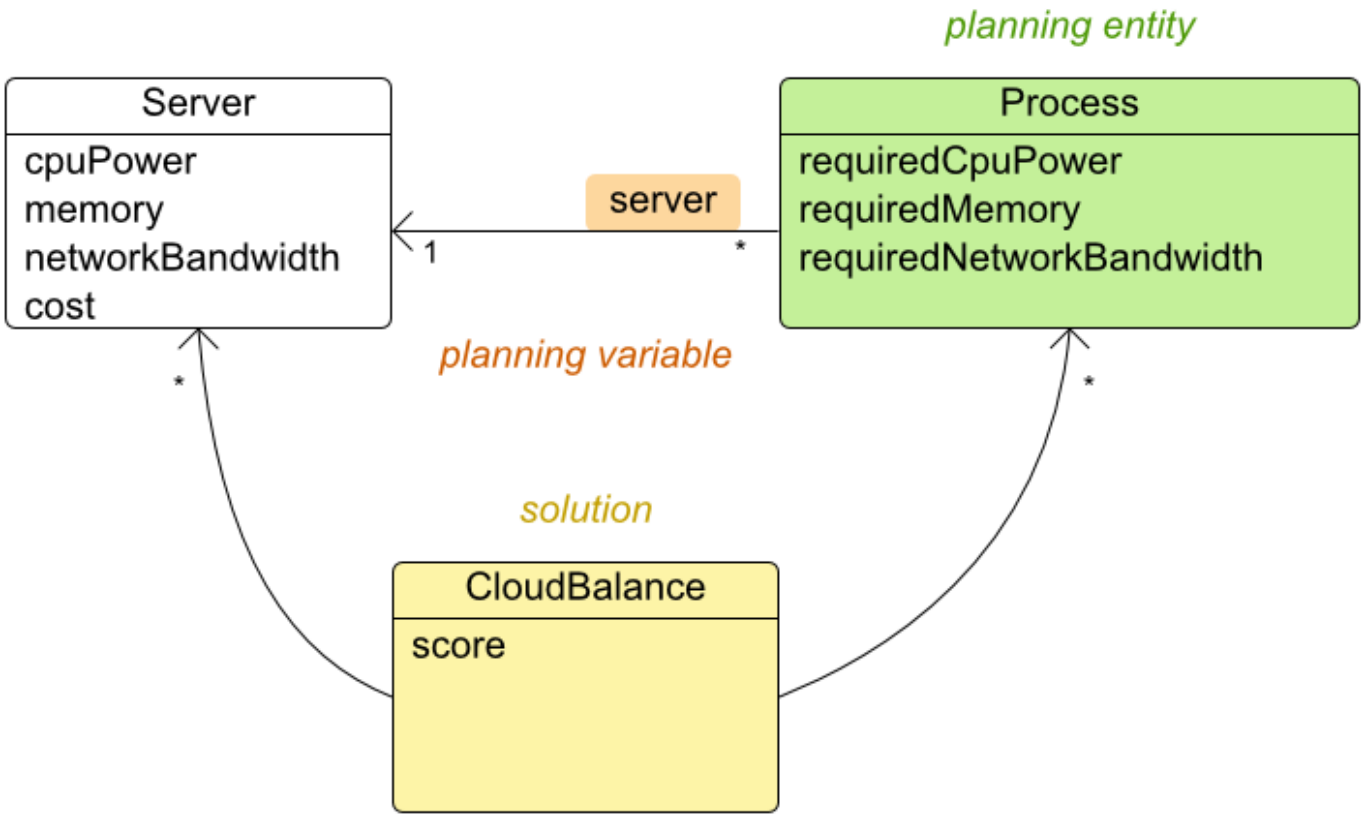
Process is a planning entity

```
@PlanningEntity
public class Process {

  private int requiredCpuPower;
  private int requiredMemory;
  private int requiredNetworkBandwidth;

  ...

  // getters, clone, equals, hashcode
```

Process has a planning variable

```
@PlanningEntity
public class Process {
  ...
  private Server server;

  @PlanningVariable
  @ValueRangeFromSolutionProperty(
    propertyName = "serverList")
  public Server getServer() {return server;}
  public void setServer(Server server) {...}

}
```

CloudBalance



Solution CloudBalance: problem facts

```
public class CloudBalance
    implements Solution<HardAndSoftScore> {

  private List<Server> serverList;

  public List<Server> getServerList() {
    return serverList;
  }

  ...

  // clone, equals, hashcode
```

Solution CloudBalance: planning entities

```
public class CloudBalance
    implements Solution<HardAndSoftScore> {
  ...

  private List<Process> processList;

  @PlanningEntityCollectionProperty
  public List<Process> getProcessList() {
    return processList;
  }

  ...
```

Solution CloudBalance: getProblemFacts

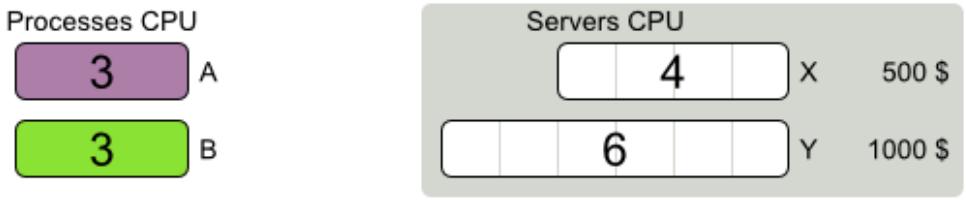
```
public class CloudBalance
    implements Solution<HardAndSoftScore> {
  ...

  // Used in score constraints
  public Collection<Object> getProblemFacts() {
    List<Object> facts = new ArrayList<Object>();
    facts.addAll(serverList);
    return facts;
  }

  ...
```

Solution CloudBalance: score

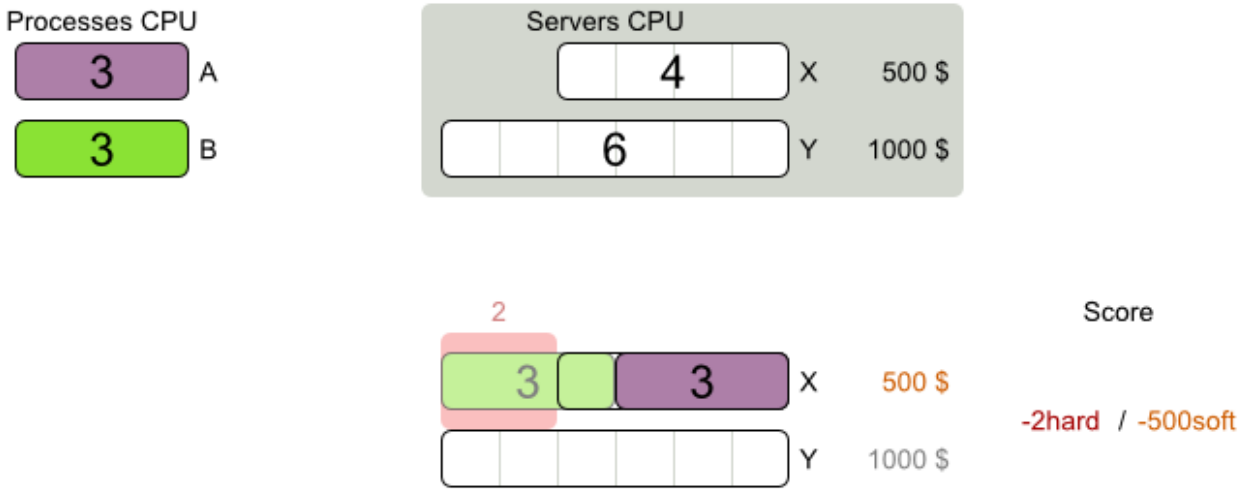
```
public class CloudBalance
    implements Solution<HardAndSoftScore> {
  ...

  private HardAndSoftScore score;

  public HardAndSoftScore getScore() {...}
  public void setScore(HardAndSoftScore score) {...}

}
```

Score constraints

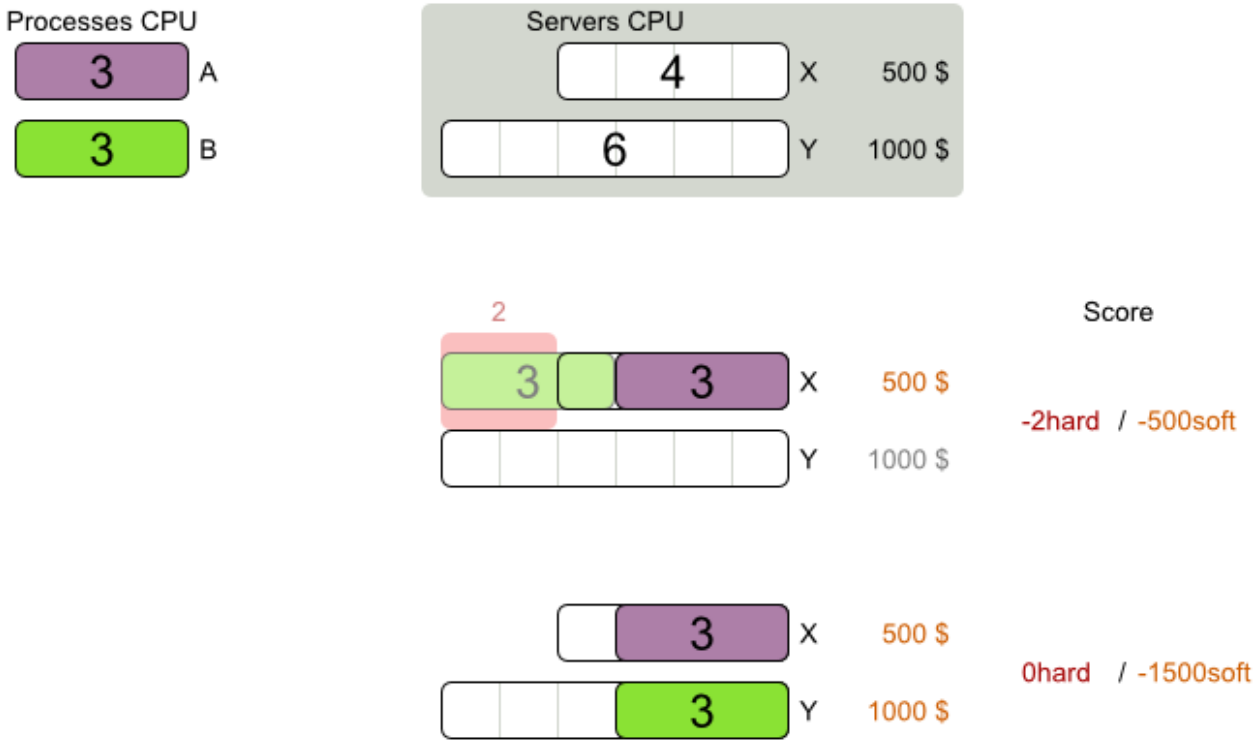
Each Solution has 1 Score



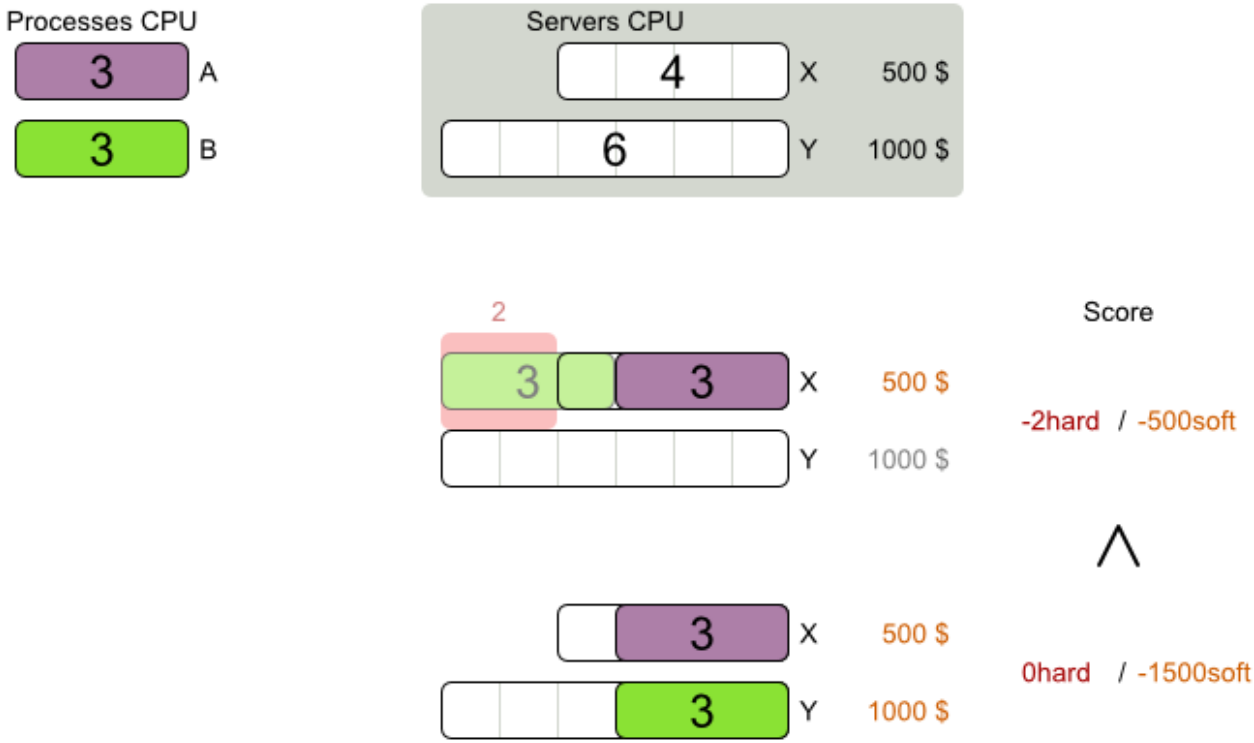
Each Solution has 1 Score



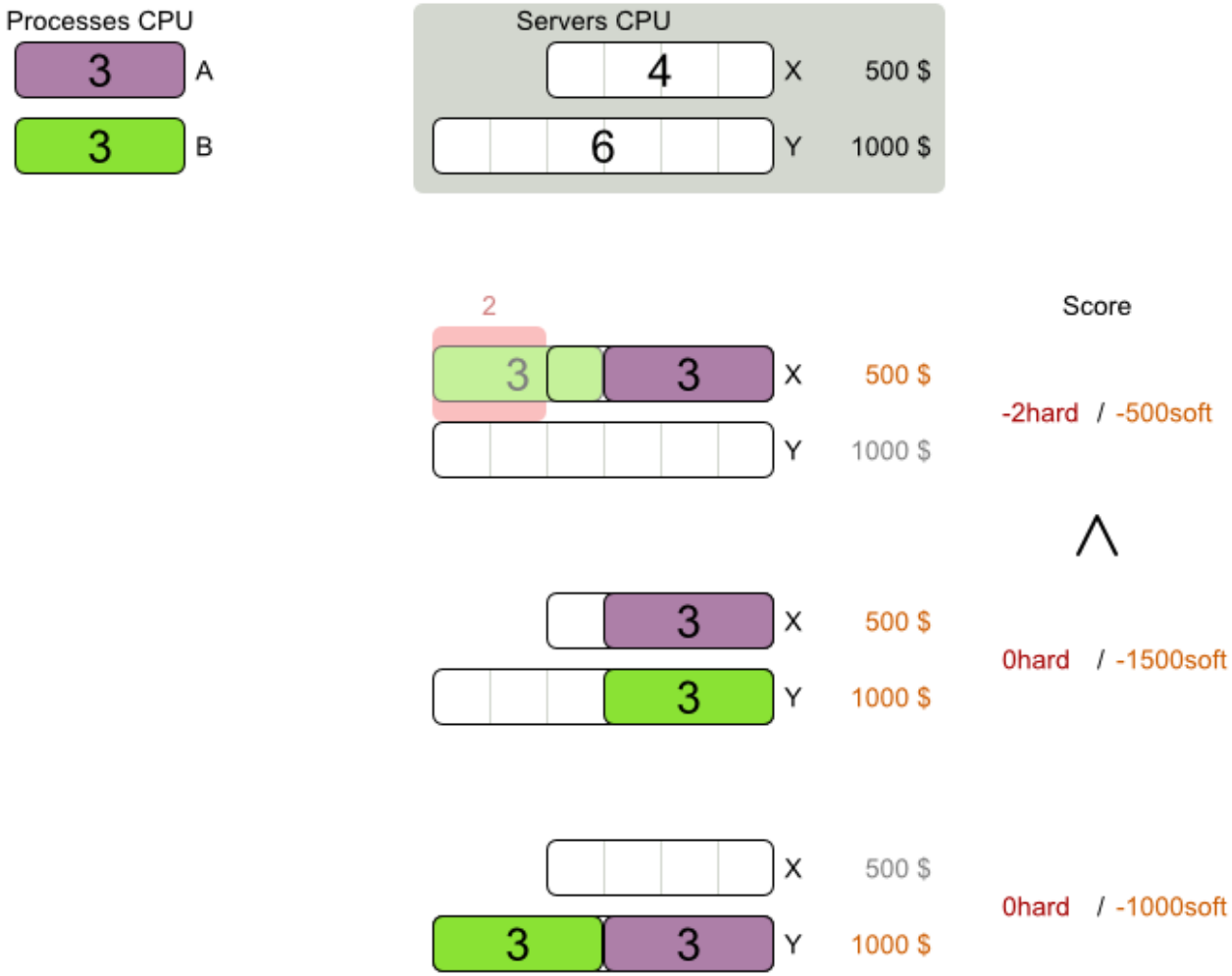
Better score => better solution



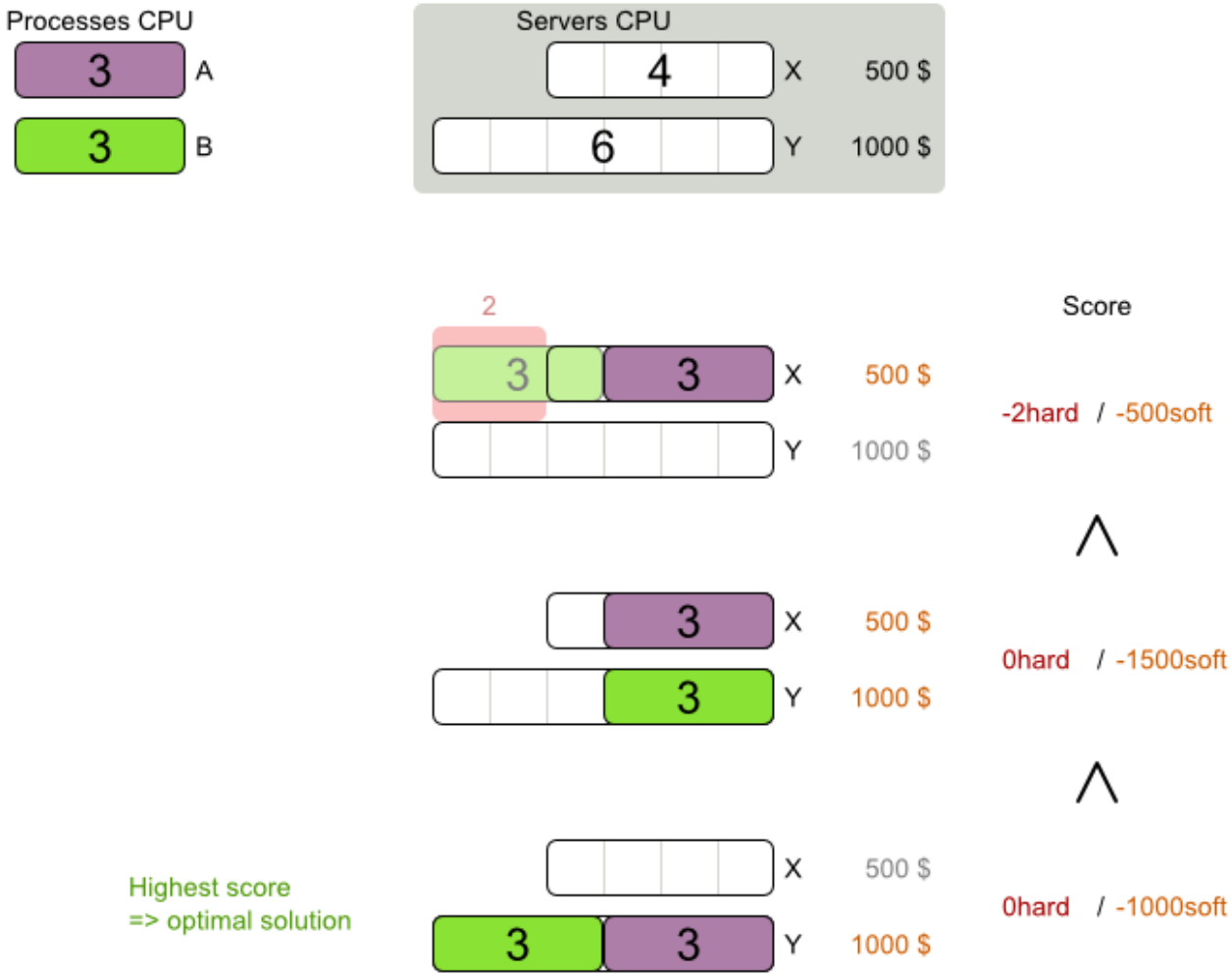
Better score => better solution



Best score => best solution



Best score => best solution



Score calculation with Drools rule engine

- DRL
  - Declarative (not imperative)
  - Like SQL, regular expressions
- Performance + scalability
  - Indexing, ReteOO, ...
  - Delta based score calculation
    - Solution changes => only recalculate part of score

Soft constraint: server cost

```
rule "serverCost"
  when
    // there is a server
    $s : Server($c : cost)
    // there is a processes on that server
    exists Process(server == $s)
  then
    // lower soft score by $c
    insertLogical(new IntConstraintOccurrence(
        "serverCost", ConstraintType.NEGATIVE_SOFT,
        $c,
        $s));
end
```

Hard constraint: CPU power

```
rule "requiredCpuPowerTotal"
  when
    // there is a server
    $s : Server($cpu : cpuPower)
    // with too little cpu for its processes
    $total : Number(intValue > $cpu) from accumulate(
      Process(server == $s,
          $requiredCpu : requiredCpuPower),
      sum($requiredCpu)
    )
  then
    // lower hard score by ($total - $cpu)
end
```

Solver configuration by XML

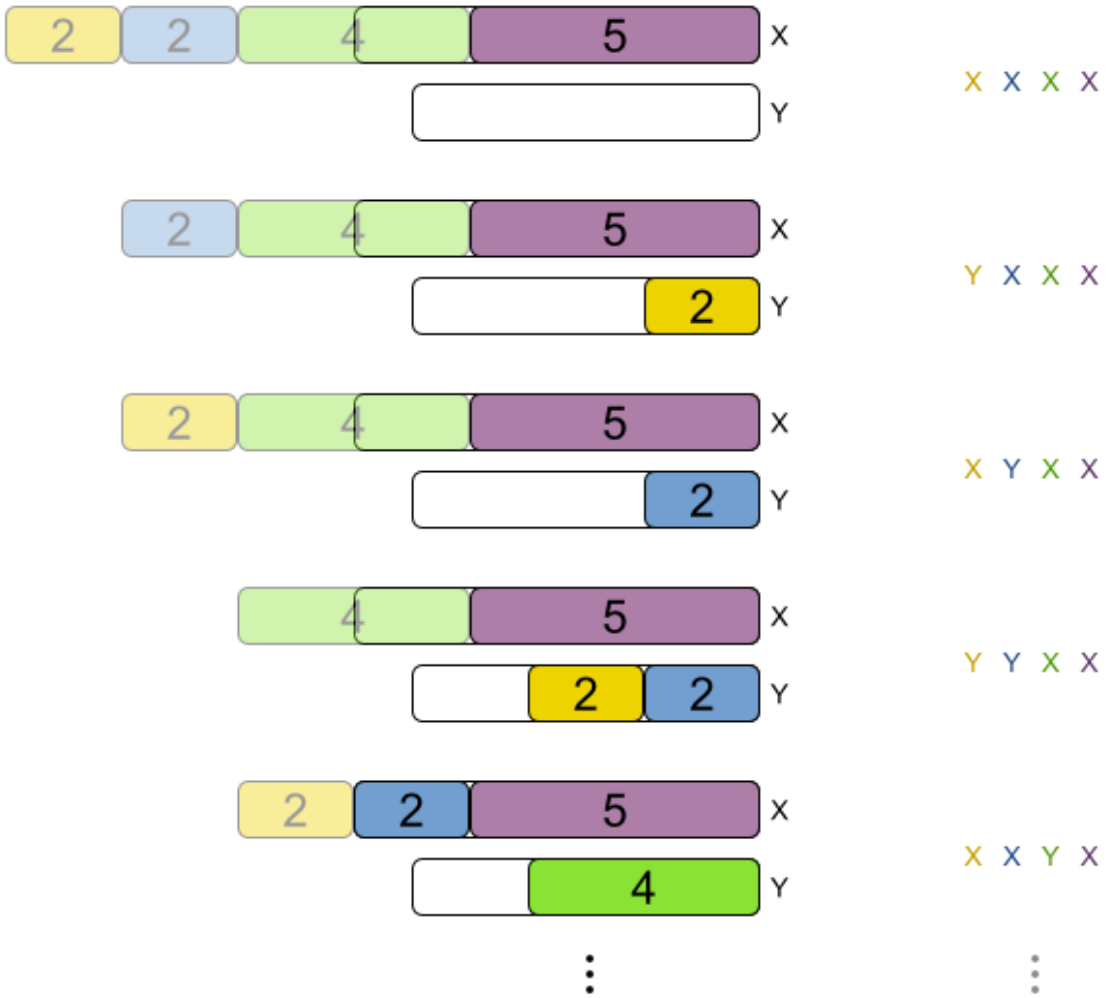
```
<solver>
  <solutionClass>...CloudBalance</solutionClass>
  <planningEntityClass>...Process</>

  <scoreDrl>...ScoreRules.drl</scoreDrl>
  <scoreDefinition>
    <scoreDefinitionType>HARD_AND_SOFT</>
  </scoreDefinition>

  <!-- optimization algorithms →
  ...
</solver>
```

Solving

```
XmlSolverConfigurer config = new XmlSolverConfigurer(
    "...SolverConfig.xml");
Solver solver = config.buildSolver();


solver.setPlanningProblem(cloudBalance);
solver.solve();
cloudBalance = (CloudBalance)
solver.getBestSolution();
```

Optimization algorithms

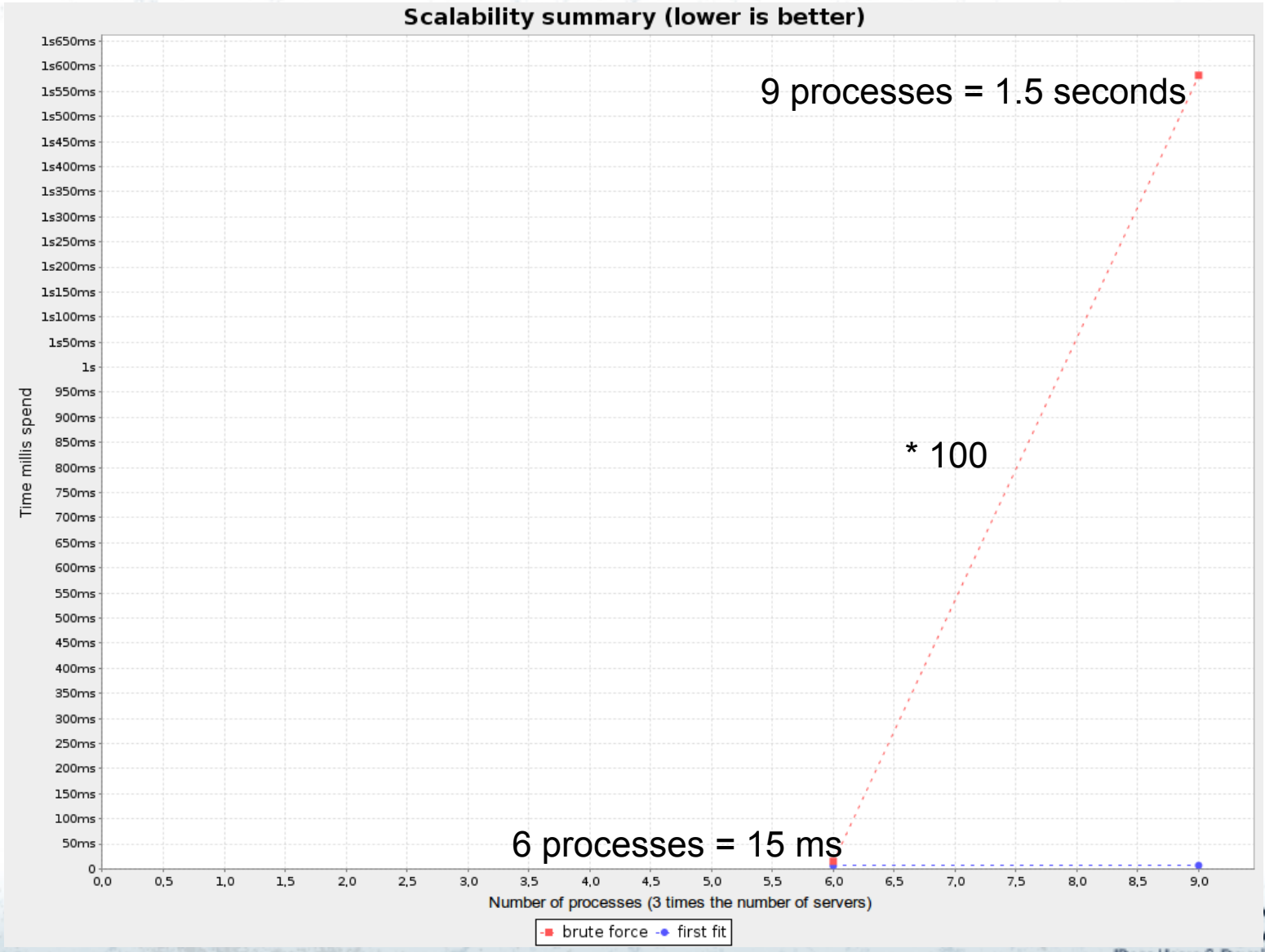
Brute Force



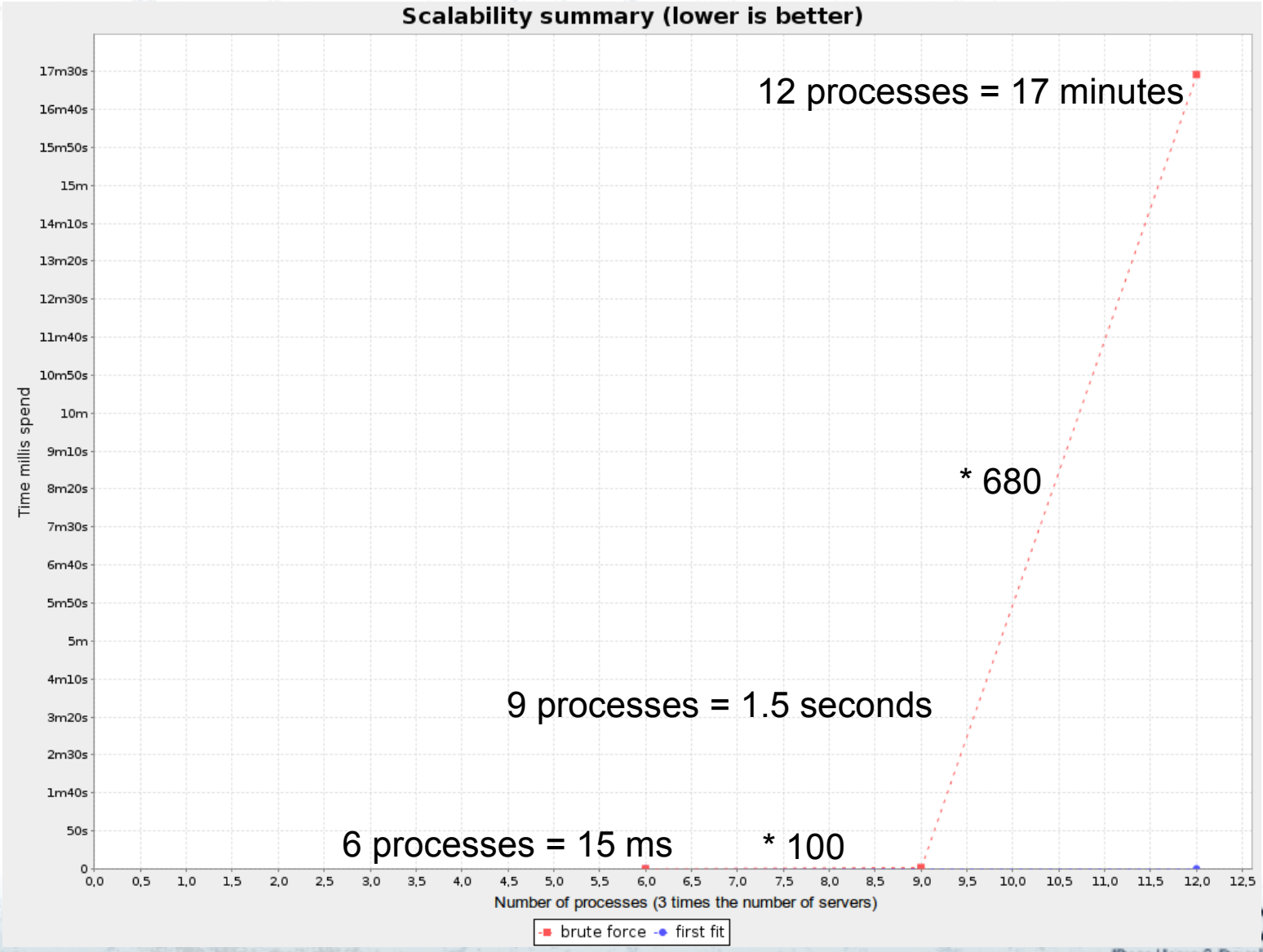
Brute Force config

```
<solver>
  ...

  <bruteForce />
</solver>
```

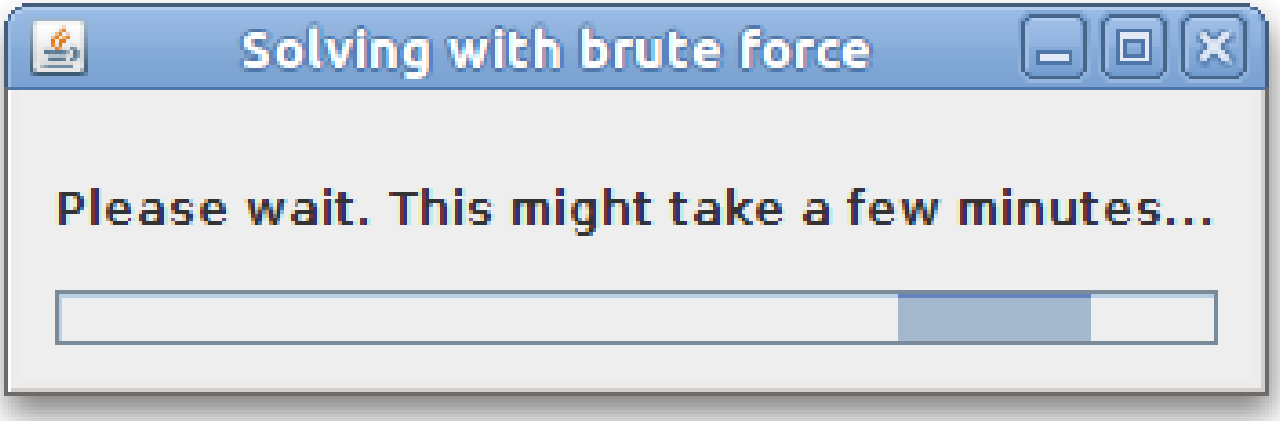
Brute force scalability



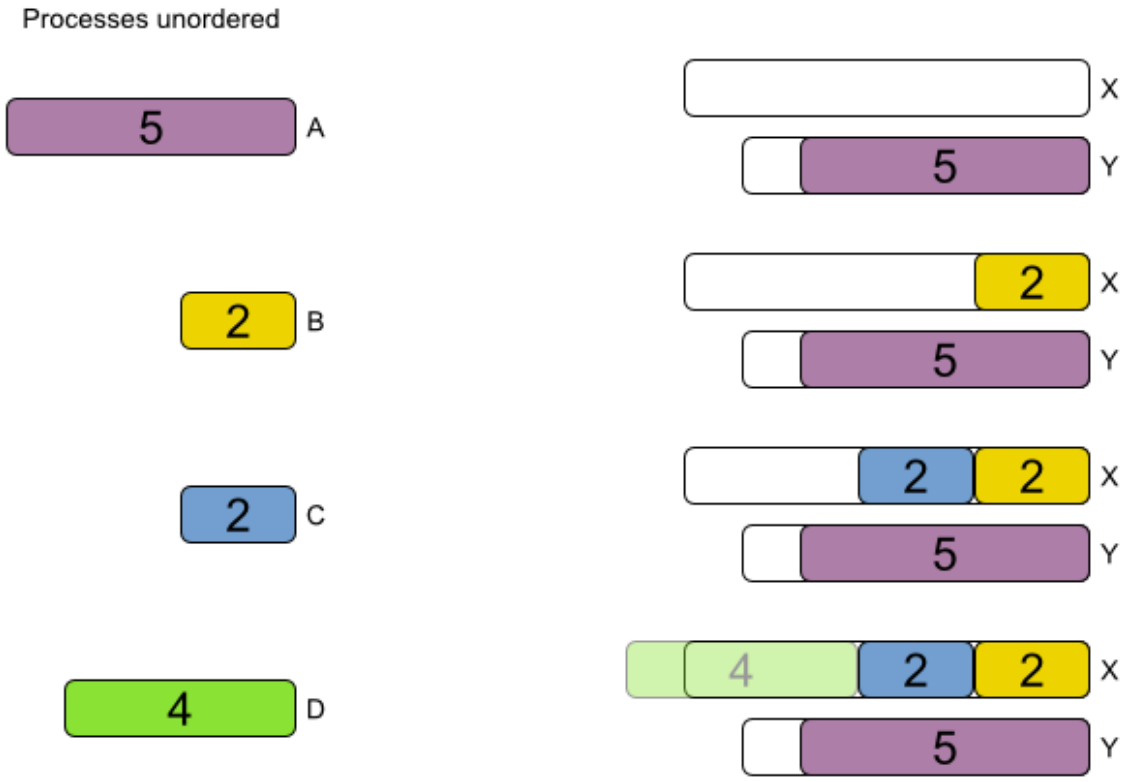
Brute force scalability



Plan 1200 processes with brute force?



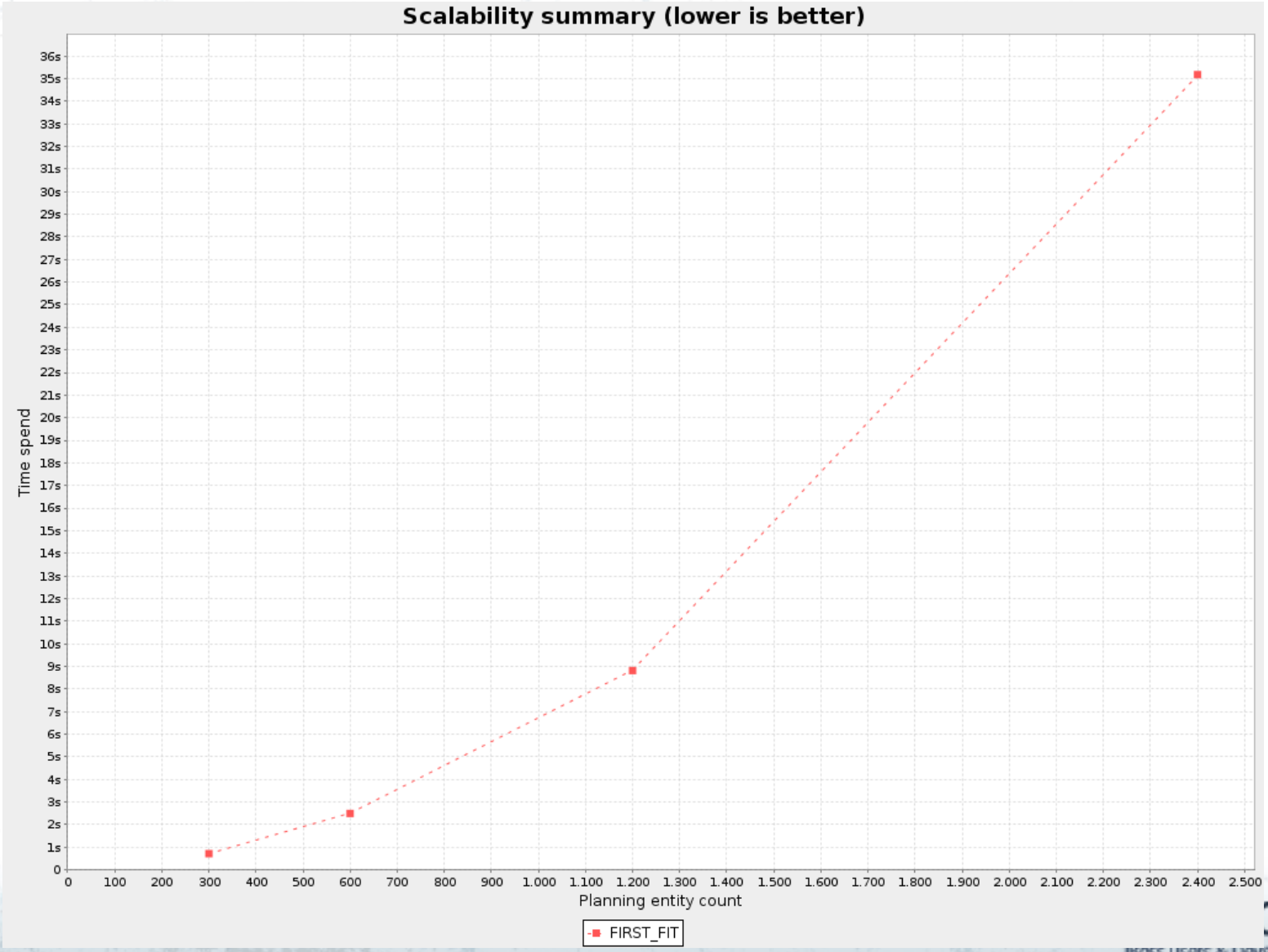
First Fit



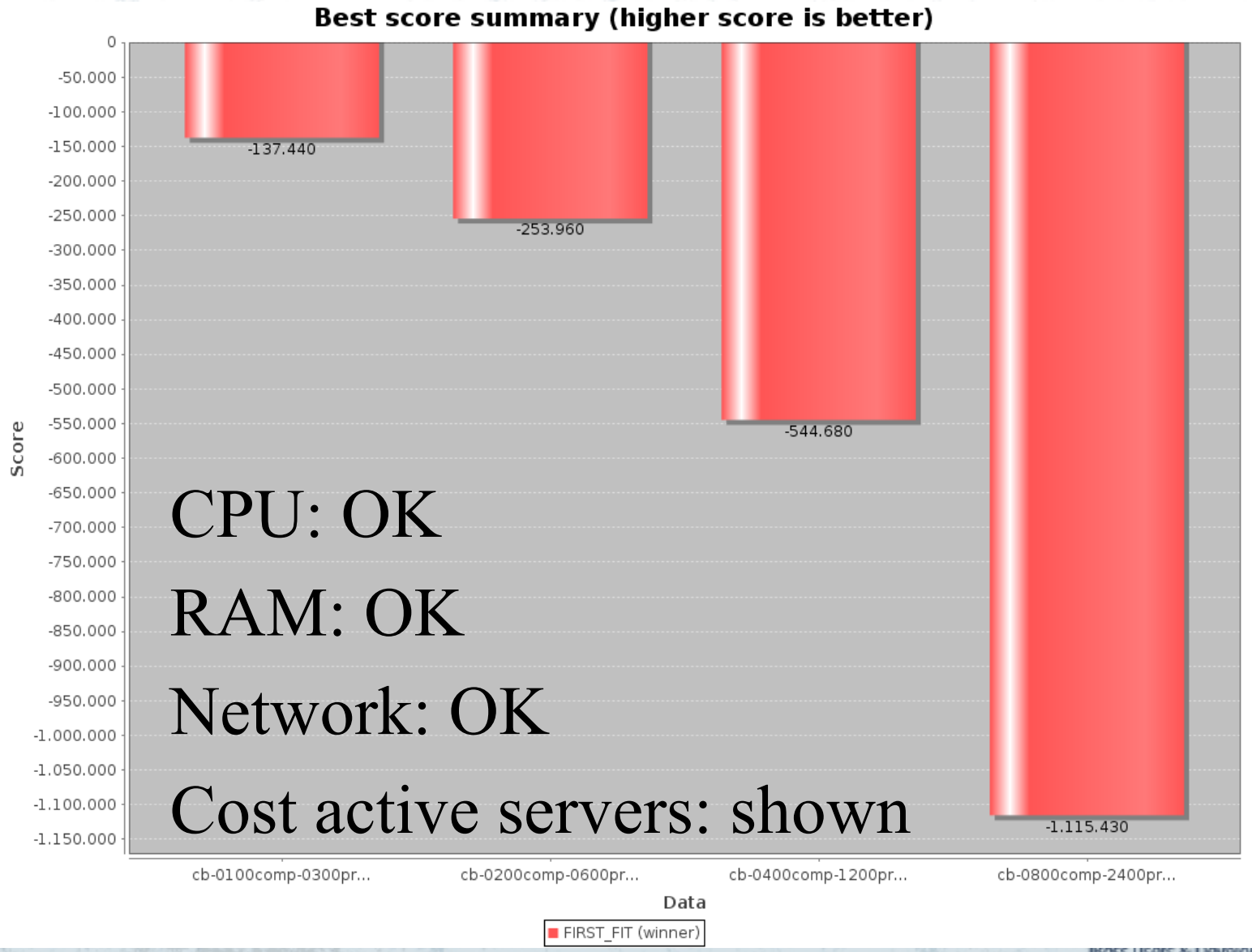
First Fit config

```
<solver>
  ...

  <constructionHeuristic>
    <constructionHeuristicType>FIRST_FIT</>
  </constructionHeuristic>
</solver>
```

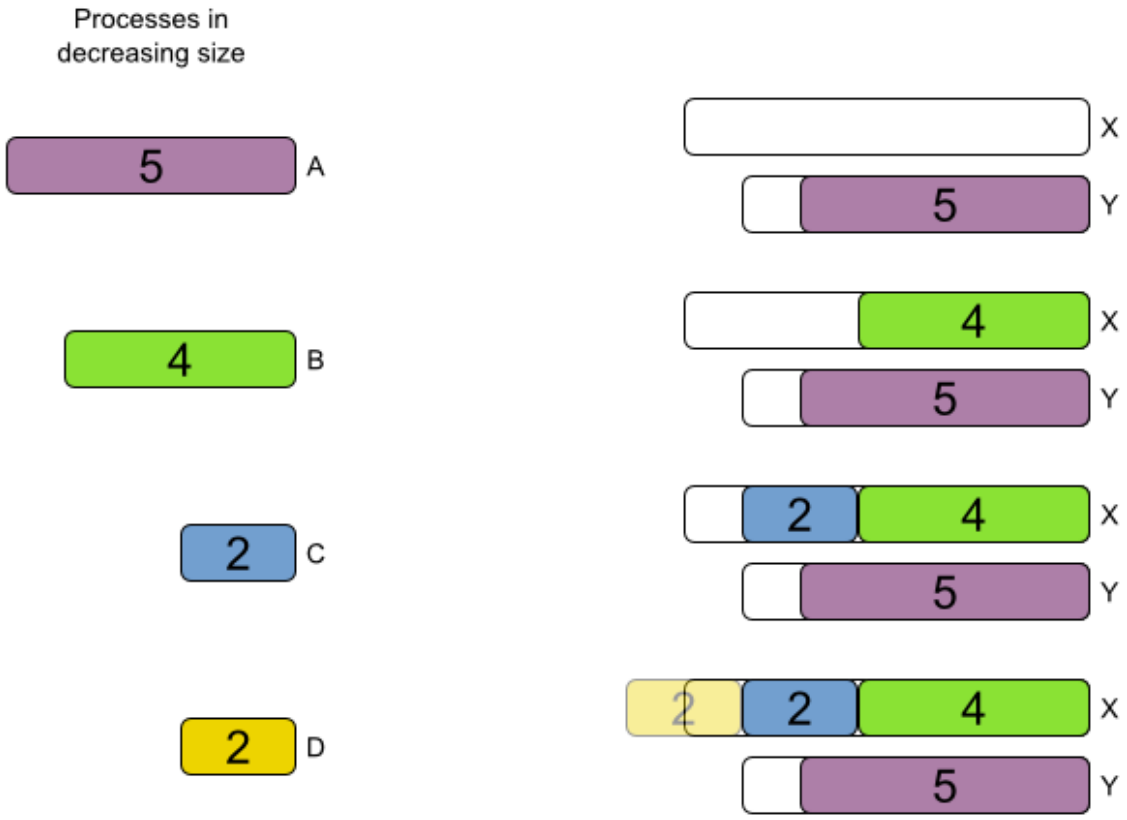
First Fit scalability



First Fit results



First Fit Decreasing



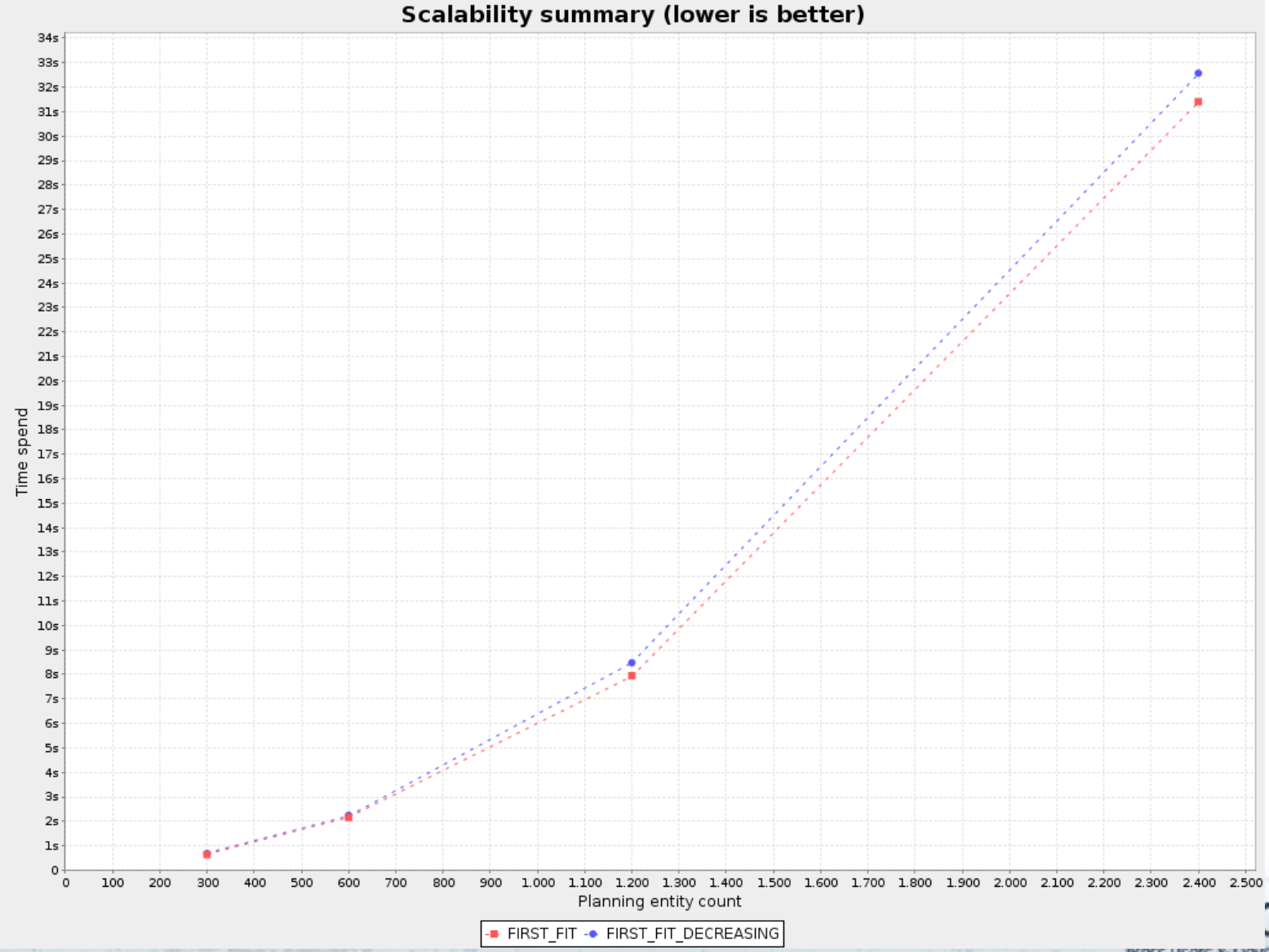
First Fit Decreasing config

```
<solver>
  ...

  <constructionHeuristic>
    <constructionHeuristicType>FIRST_FIT_DECREASING</>
  </constructionHeuristic>
</solver>
```

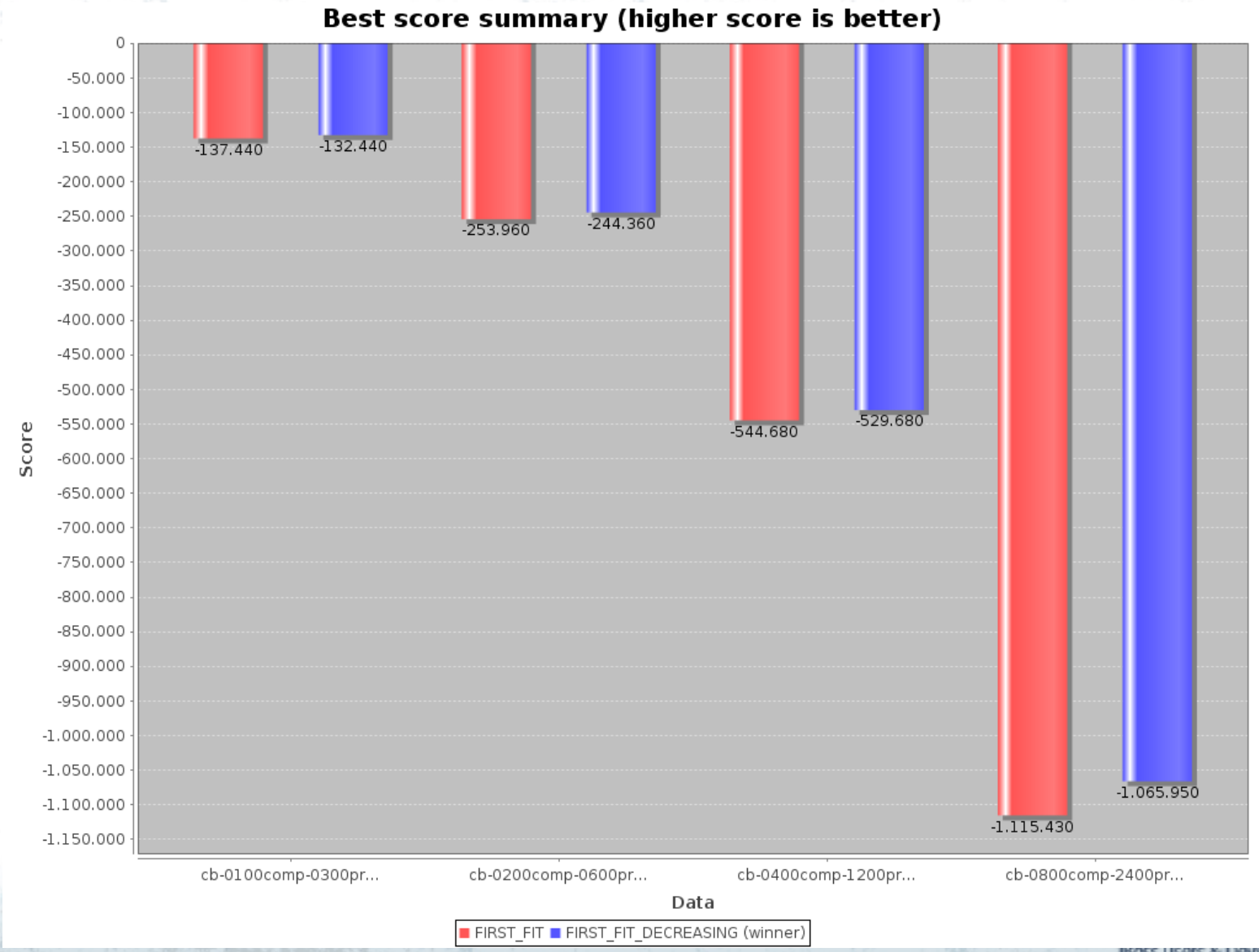
DifficultyComparator

```
public class ProcessDifficultyComparator
    implements Comparator<Process> {
  public int compare(Process a, Process b) {
    // Compare on requiredCpuPower * requiredMemory
    //     * requiredNetworkBandwidth
  }
}

@PlanningEntity(difficultyComparatorClass
    = ProcessDifficultyComparator.class)
public class Process  {
  ...
}
```

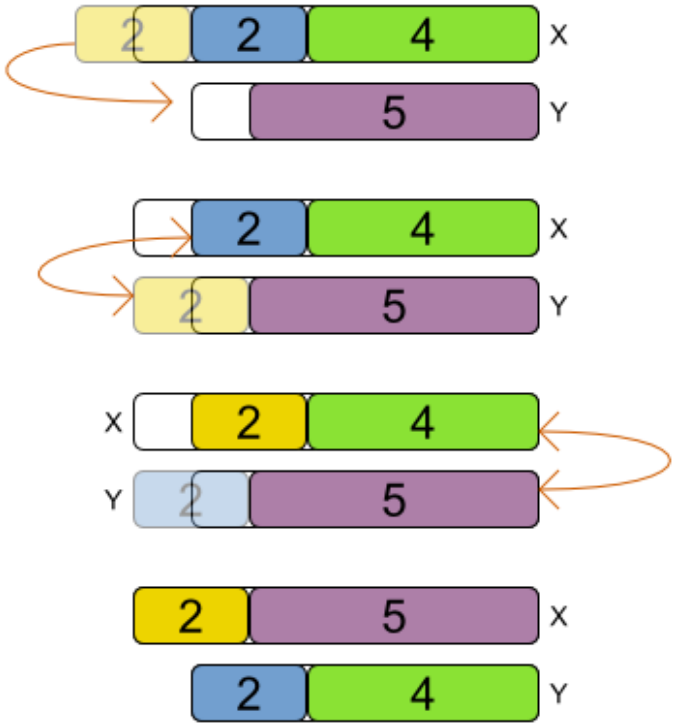
First Fit Decreasing scalability



First Fit Decreasing results



Local Search



Local Search comes after Construction Heuristics

```
<solver>
  ...

  <constructionHeuristic>
    <constructionHeuristicType>FIRST_FIT_DECREASING</>
  </constructionHeuristic>
  <localSearch>
    ...
  <localSearch>
</solver>
```

Local Search needs to be terminated

```
<solver>
  ...

  <termination>
    <maximumMinutesSpend>20</maximumMinutesSpend>
  </termination>
  ...
</solver>
```

Selecting moves

```
<localSearch>
  <selector>
    <selector>

<moveFactoryClass>GenericChangeValueMoveFactory</>
    </selector>
    <selector>
      <moveFactoryClass>GenericSwitchAllValuesMF</>
    </selector>
  </selector>
  ... tabu search, simulated annealing or ...
</localSearch>
```

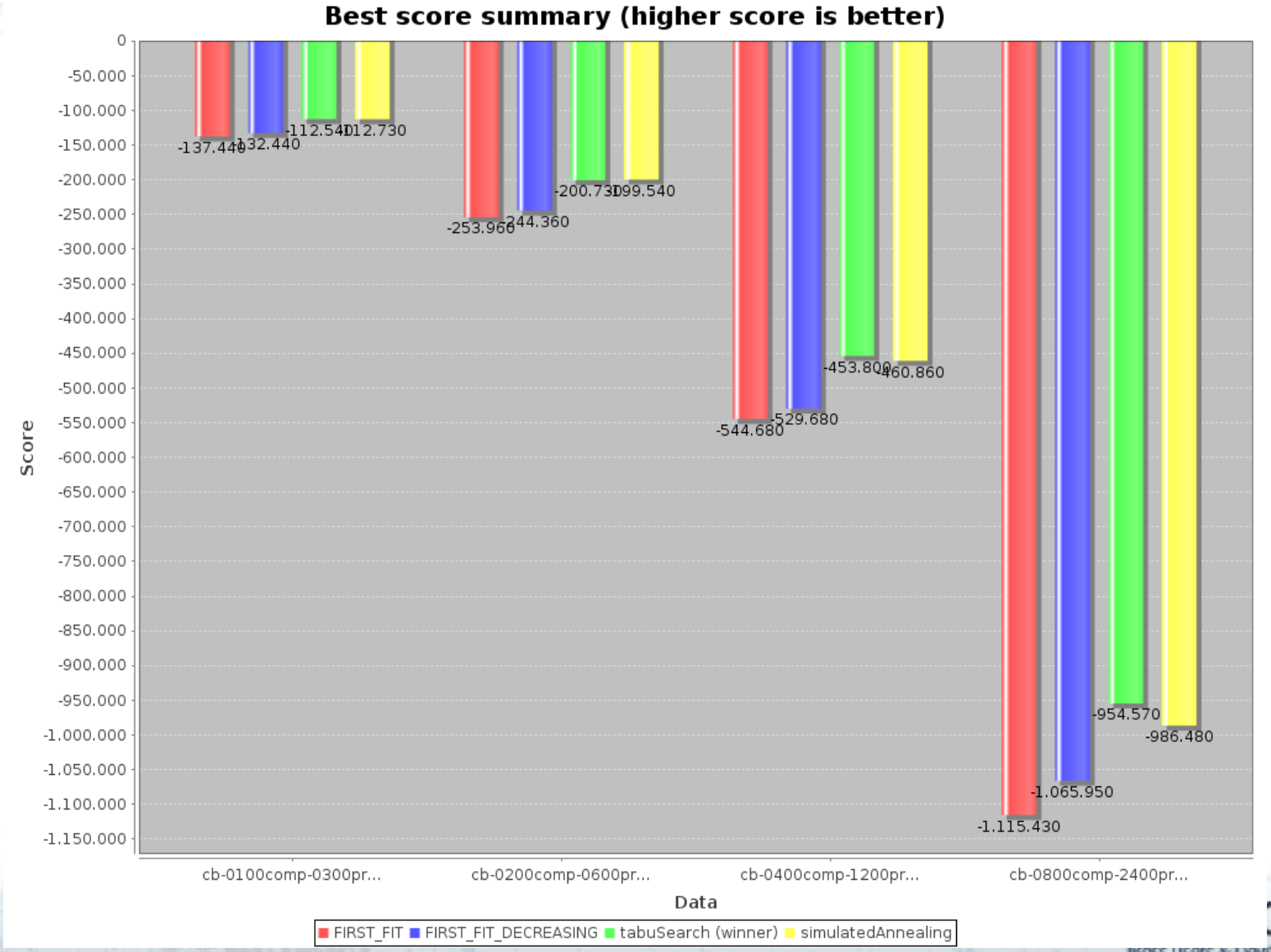
Tabu Search

```
<localSearch>
  <selector>...</selector>
  <acceptor>
    <!-- Untweaked standard values -->
    <solutionTabuSize>1000</solutionTabuSize>
    <propertyTabuSize>7</propertyTabuSize>
  </acceptor>
  <forager>
    <!-- Untweaked standard value -->
    <minimalAcceptedSelection>1000</>
  </forager>
</localSearch>
```

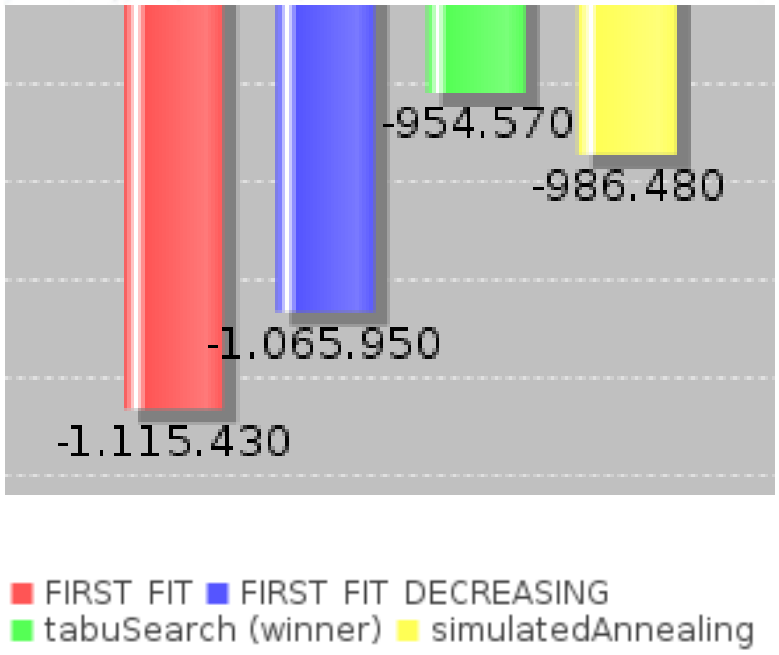
Or Simulated Annealing

```
<localSearch>
  <selector>...</selector>
  <acceptor>
    <!-- Tweaked value -->
    <simulatedAnnealingStartingTemperature>
        0hard/400soft</>
  </acceptor>
  <forager>
    <!-- Untweaked standard value -->
    <minimalAcceptedSelection>4</>
  </forager>
</localSearch>
```

Metaheuristics results



Cost ($) reduction



- Compared to First Fit
  - First Fit Decreasing
    - 49 480 $ = 4 %
  - Tabu Search
    - 160 860 $ = 14 %
  - Simulated annealing
    - 128 950 $ = 11 %
- Few constraints here
  - => low ROI

Organizations rarely optimize planning problems.

http://www.flickr.com/photos/techbirmingham/345897594/

Organizations waste resources.

Real-time planning

Real-time paradox

- First Fit Decreasing 1200 processes
  - 8 seconds
- Time allowed
  - 100ms after last change

Demo

Real-time planning

CloudBalance example

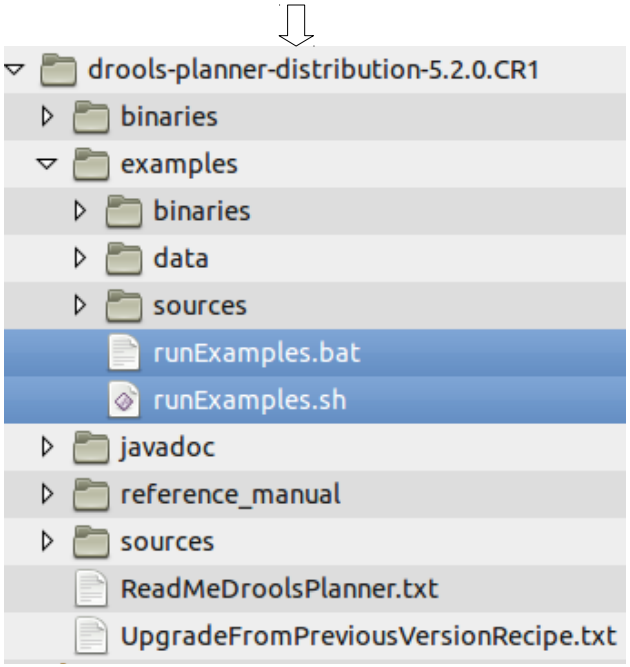
Summary

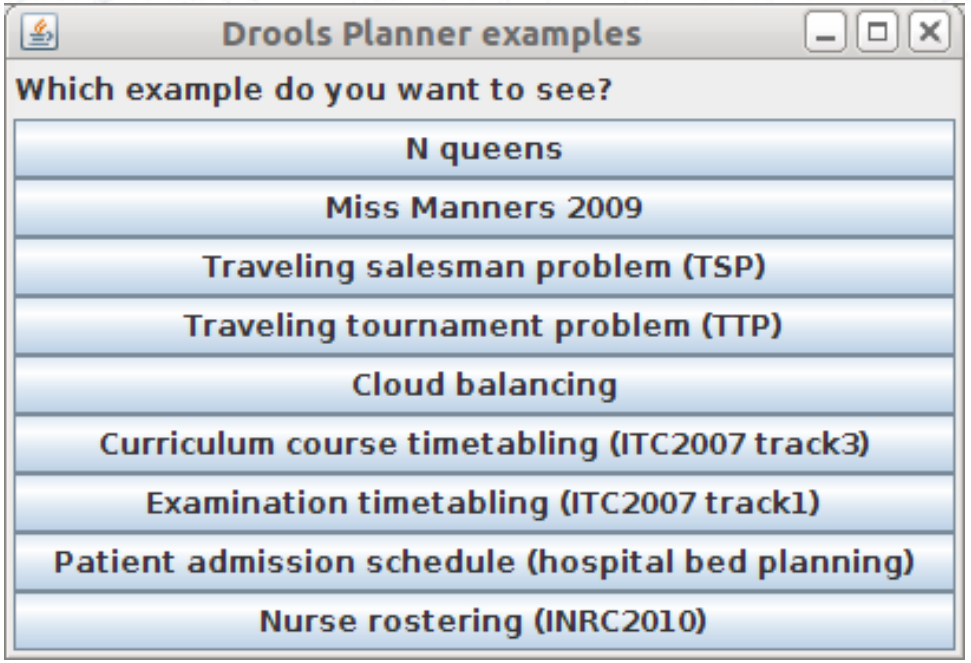
Summary

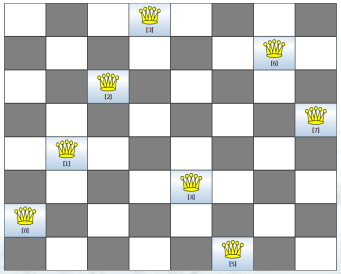
- Drools Planner optimizes planning
- Adding constraints is easy and scalable
- Switching/combining algorithms is easy

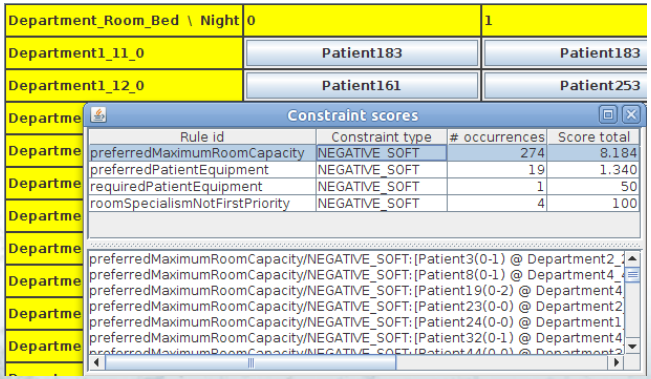
Try an example!

drools-planner-distribution-5.2.0.CR1.zip









E \ SD	01-01	01-02	01-03	01-04	01-05
0(0)	E	E	E	E	N
1(1)	E	E	E	E	N
2(2)				E	E
3(3)	E	E	E	L	D
4(4)	E	L	L	L	D
5(5)	L	L	L	L	

Q&A

- JBoss Drools Planner homepage:
  - http://www.jboss.org/drools/drools-planner
- Reference manual:
  - http://www.jboss.org/drools/documentation
- Download this presentation:
  - http://www.jboss.org/drools/presentations

- Twitter: @geoffreydesmet
- Google+: Geoffrey De Smet