Pushing the rule engine to its limits with Drools Planner

Geoffrey De Smet

Agenda

- Drools Platform overview
- Use cases
  - Bin packaging
    - What is NP complete?
  - Employee shift rostering
    - Hard and soft constraints
  - Patient admission schedule
    - How many possible solutions?
- Algorithms
  - Meta-heuristics
- Benchmarking

Drools Platform

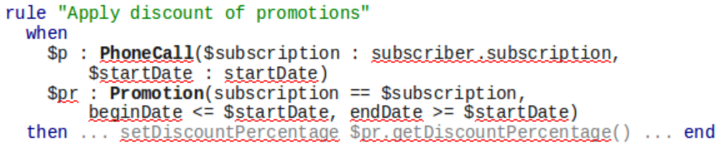
Overview

Business Logic Integration

Business Logic Integration Platform

Drools Expert

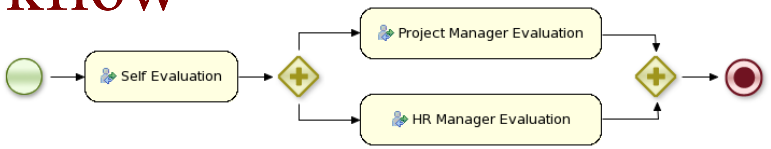
Rule engine



```
rule "Apply discount of promotions"
  when
    $p : PhoneCall($subscription : subscriber.subscription,
         $startDate : startDate)
    $pr : Promotion(subscription == $subscription,
         beginDate <= $startDate, endDate >= $startDate)
  then ... setDiscountPercentage $pr.getDiscountPercentage() ... end
```

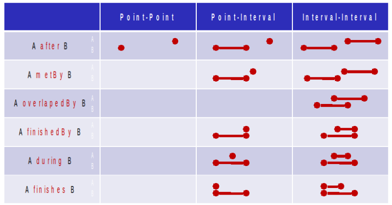
Drools Flow (jBPM 5)

Workflow



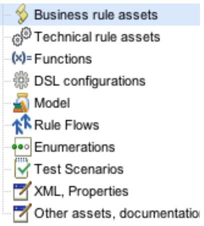
Drools Fusion

Complex event processing (CEP)



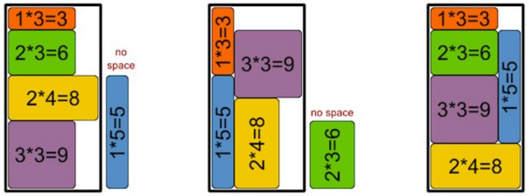
Drools Guvnor

Business Rule Management System (BRMS)



Drools Planner

Automated planning



Use cases

What are planning problems?

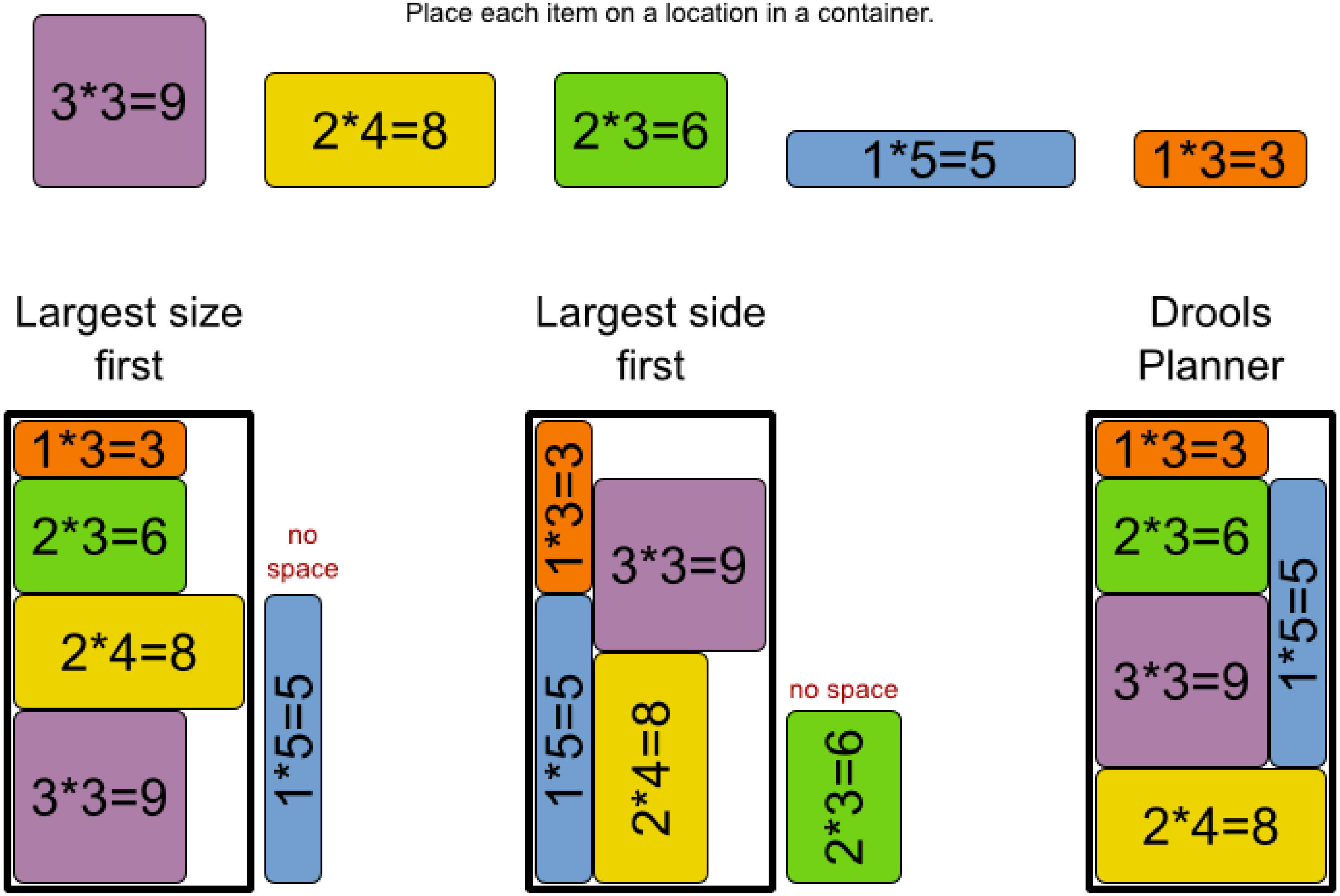
New office furniture... 1 car

Half hour later...

Wasted space

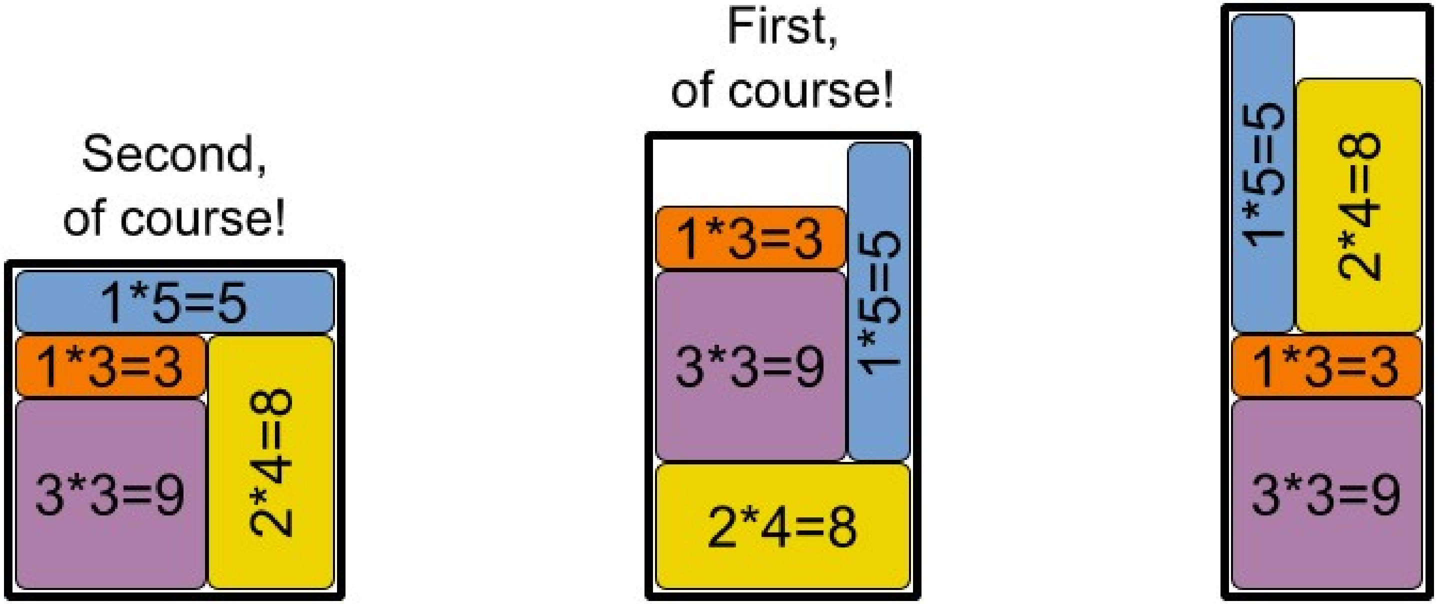
Bin packaging

Place each item on a location in a container.



Bin packaging is NP complete

When do we put 2*4=8 into the container?



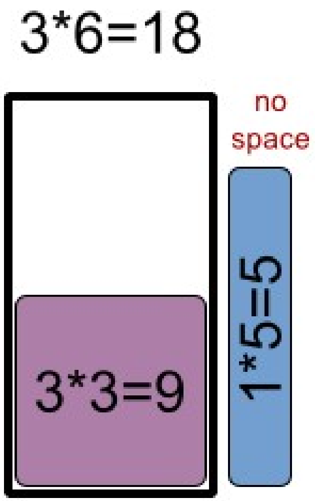
Second,
of course!

First,
of course!

Last,
of course!

A given solution can be verified fast.
There is no efficient way to find a solution

Bin packaging is NP complete



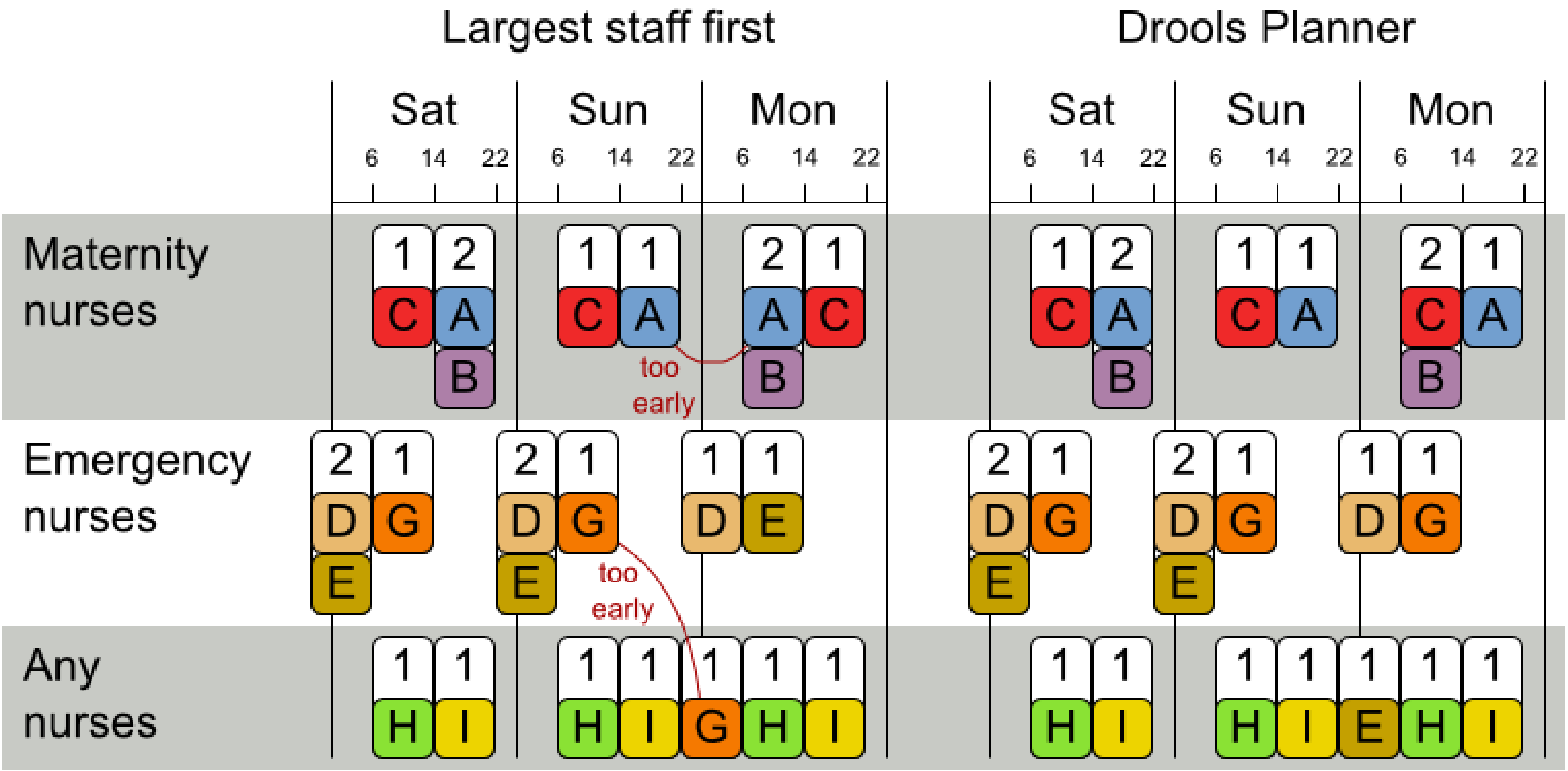
This container of size 18 can not hold these 2 items with a total size of 14.

There is no easy way to verify
if there is even a feasible solution.

Employee shift rostering

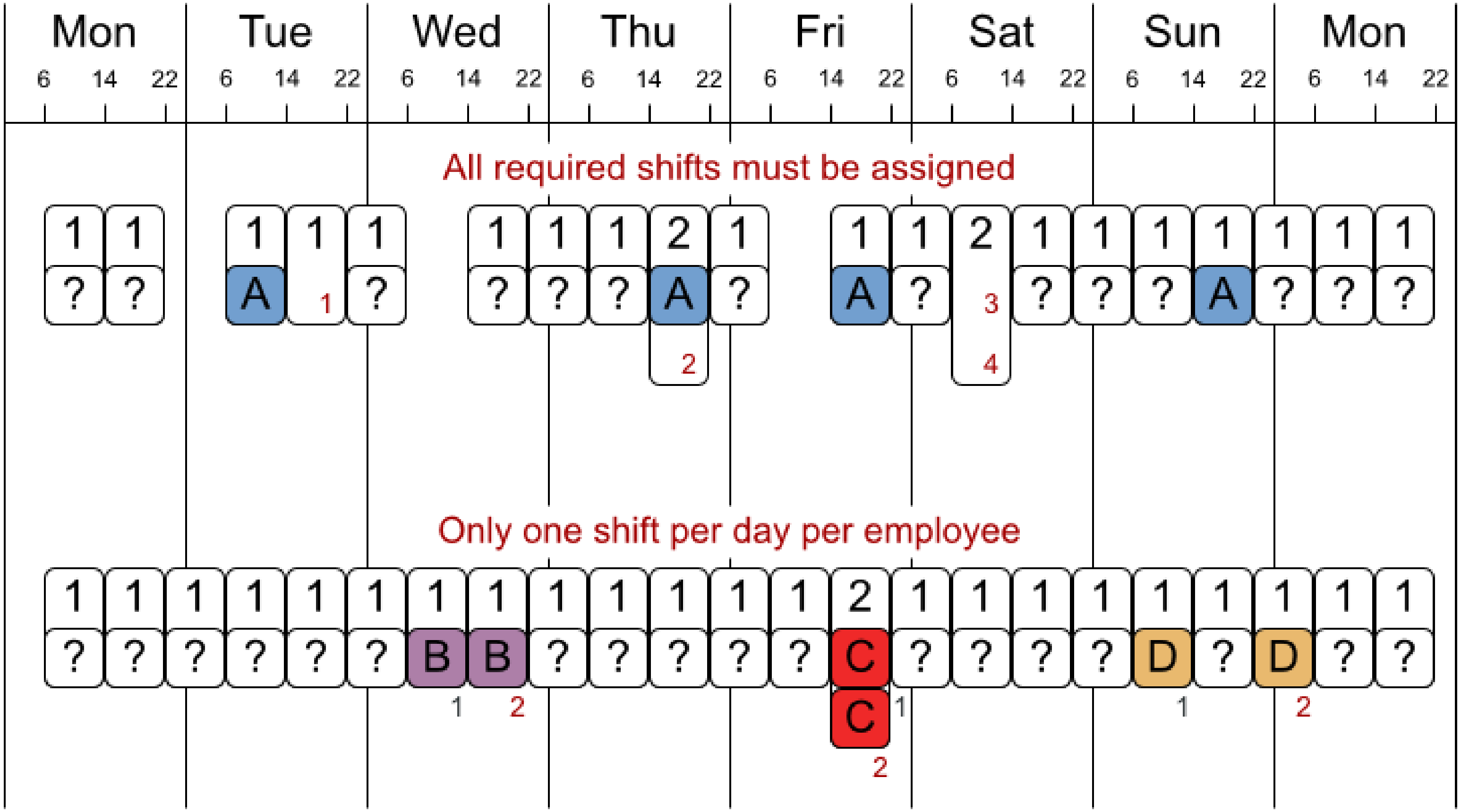
Populate each work shift with a nurse.

Maternity nurses	Emergency nurses	Basic nurses
A Ann B Beth C Cory	D Dan E Elin G Greg	H Hue I Ilse



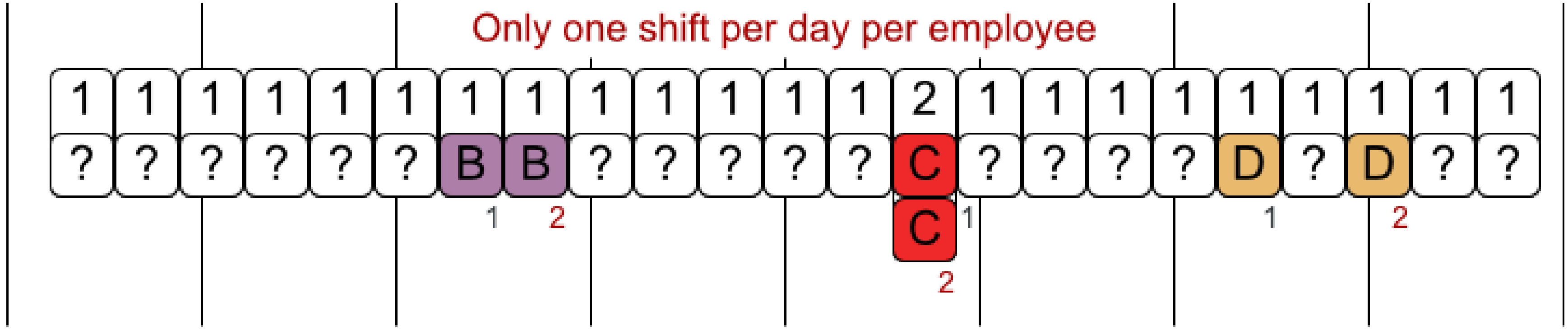
Employee shift rostering

Hard constraints


Mon
Tue
Wed
Thu
Fri
Sat
Sun
Mon
6
14
22
All required shifts must be assigned
1
2
?
A
3
4
Only one shift per day per employee
B
C
D

No hard constraint broken => solution is feasible

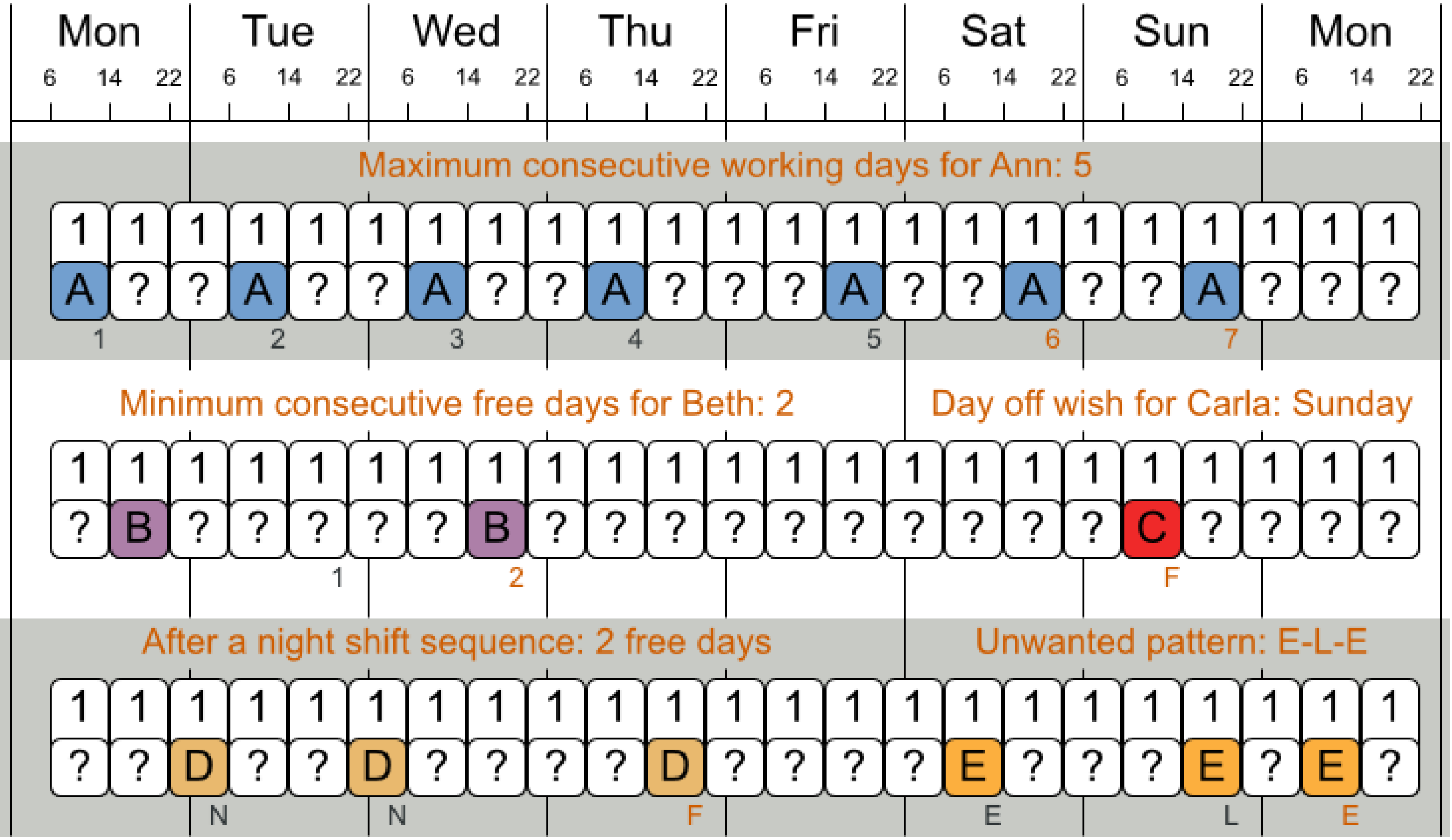
Hard constraint implementation



```
// a nurse can only work one shift per day
rule "oneShiftPerDay"
  when
    $left : EmployeeAssignment(
        $employee : employee,
        $shiftDate : shiftDate,
        $leftId : id
    );
    $right : EmployeeAssignment(
        employee == $employee,
        shiftDate == $shiftDate,
        id > $leftId);
  then
    // Lower the hard score with a weight ...
end
```

Employee shift rostering

Soft constraints



Maximum consecutive working days for Ann: 5

Minimum consecutive free days for Beth: 2

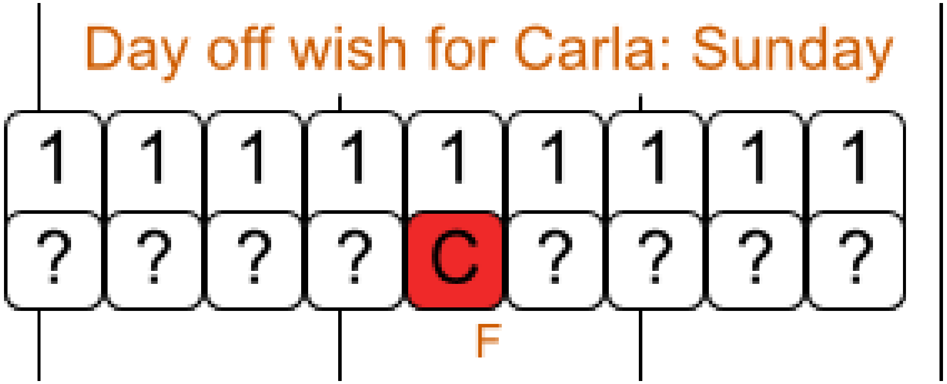
Day off wish for Carla: Sunday

After a night shift sequence: 2 free days

Unwanted pattern: E-L-E

There are many more soft constraints...

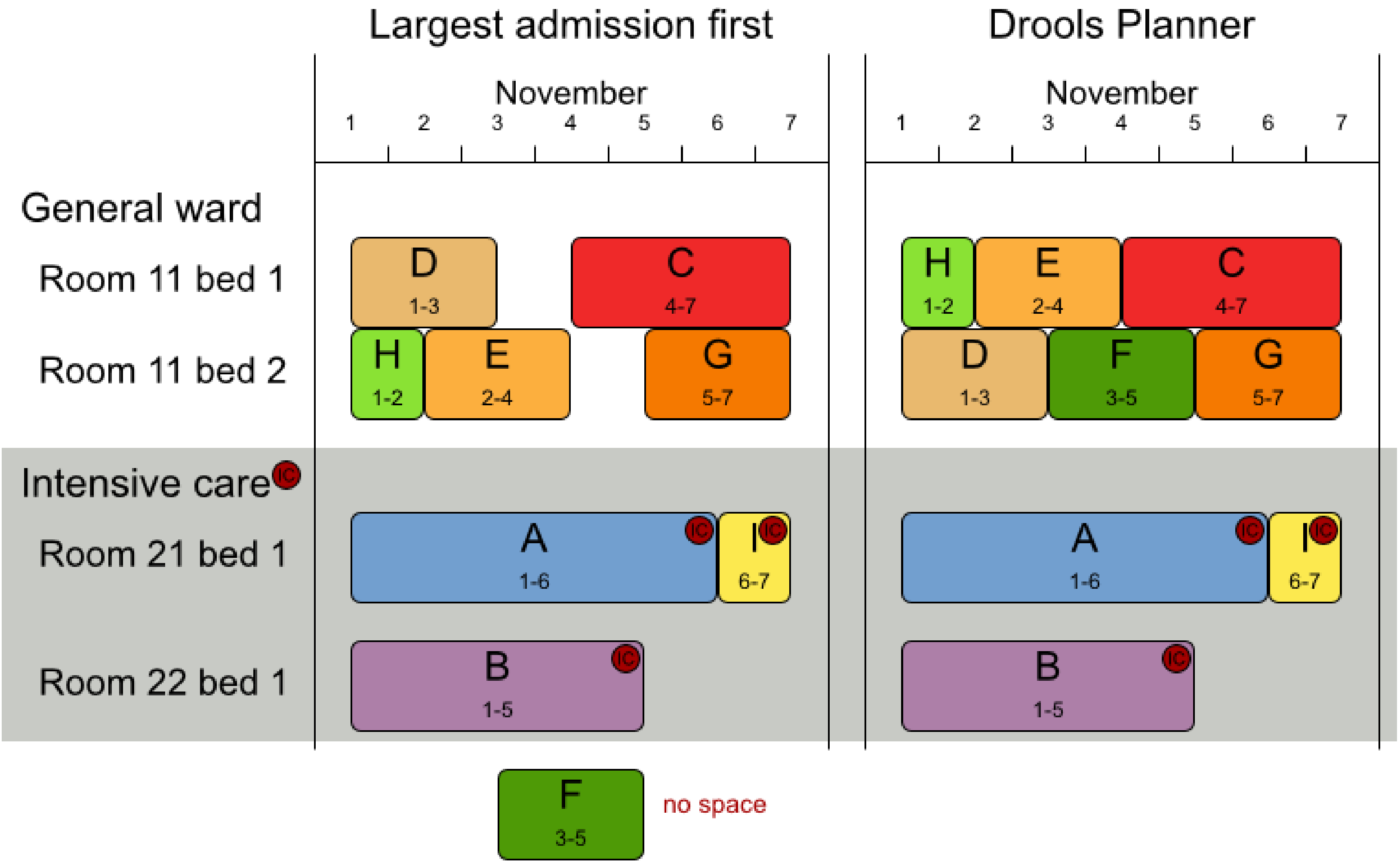
Soft constraint implementation



```
rule "dayOffRequest"
  when
    $dayOffRequest : DayOffRequest(
        $employee : employee,
        $shiftDate : shiftDate,
        $weight : weight
    );
    $employeeAssignment : EmployeeAssignment(
        employee == $employee,
        shiftDate == $shiftDate
    );
    then
      // Lower the soft score with the weight $weight ...
end
```

Patient admission schedule

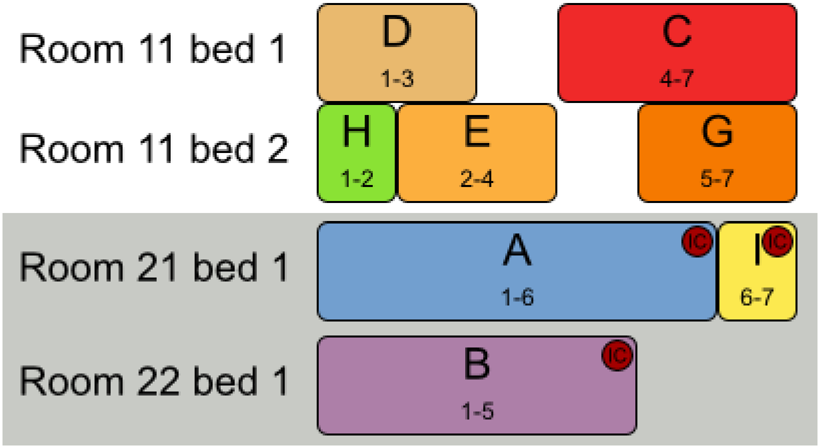
Assign each patient a hospital bed.



Patient admission schedule

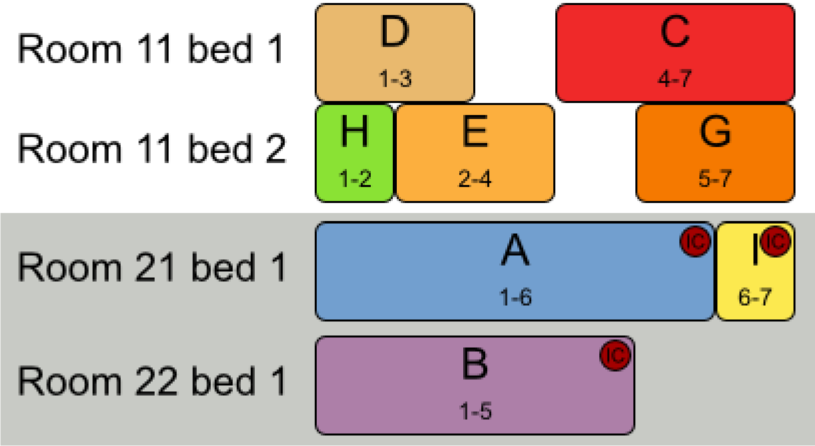
- Hard constraints
  - No 2 patients in same bed in same night
  - Room gender limitation
  - Department minimum or maximum age
  - Patient requires specific room equipment(s)
- Soft constraints
  - Patient prefers maximum room size
  - Department specialization
  - Room specialization
  - Patient prefers specific room equipment(s)

Needle in a haystack



- How many possible solutions?
  - 310 beds
    - in 105 rooms
    - in 4 departments
  - 84 nights
  - 2750 patients (admissions)
- Numbers from a real dataset

Needle in a haystack



How many possible solutions?

310 beds

in 105 rooms

in 4 departments

84 nights

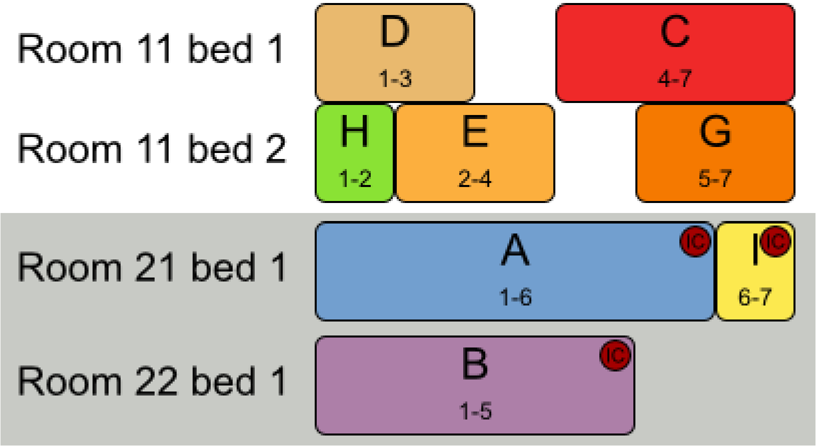
2750 patients (admissions)

> works of art in the Louvre?

35 000 works of art

Source: wikipedia

Needle in a haystack

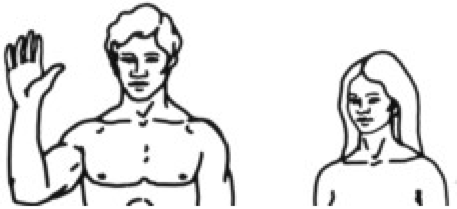


How many possible solutions?

- 310 beds
  - in 105 rooms
  - in 4 departments
- 84 nights
- 2750 patients (admissions)

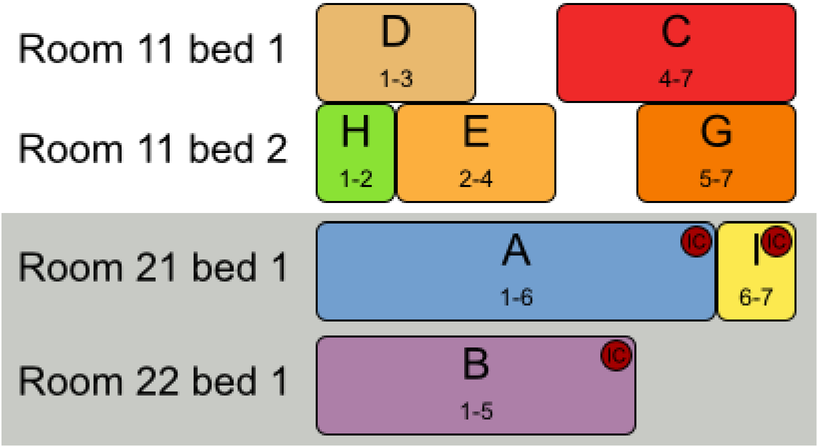
> humans?

- 7 000 000 000 humans



Source: NASA (wikipedia)

Needle in a haystack

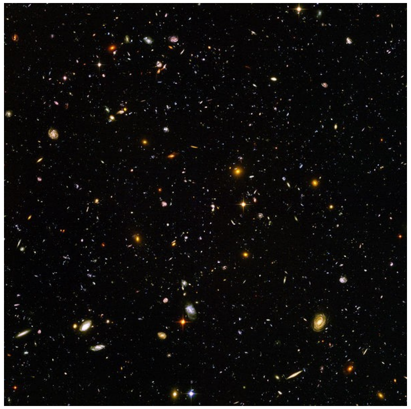


How many possible solutions?

- 310 beds
  - in 105 rooms
  - in 4 departments
- 84 nights
- 2750 patients (admissions)

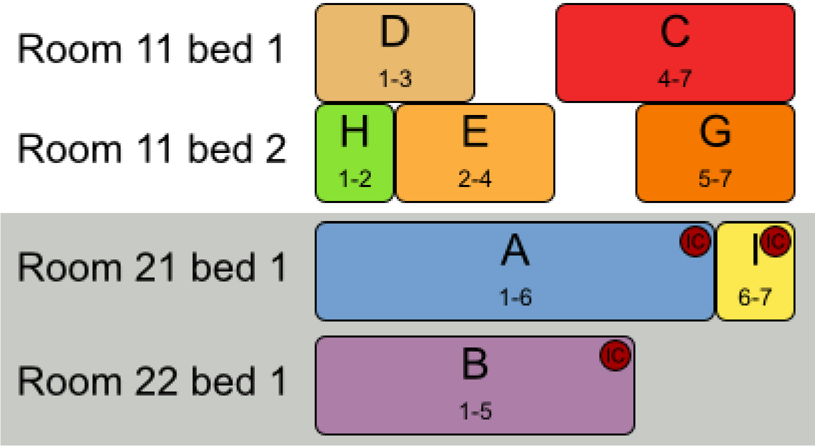
> minimum atoms in the observable universe?

- 10^80



Source: NASA and ESA (wikipedia)

Needle in a haystack



How many possible solutions?

310 beds

in 105 rooms

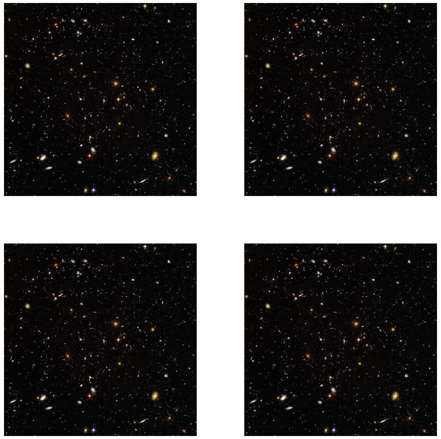
in 4 departments

84 nights

2750 patients (admissions)

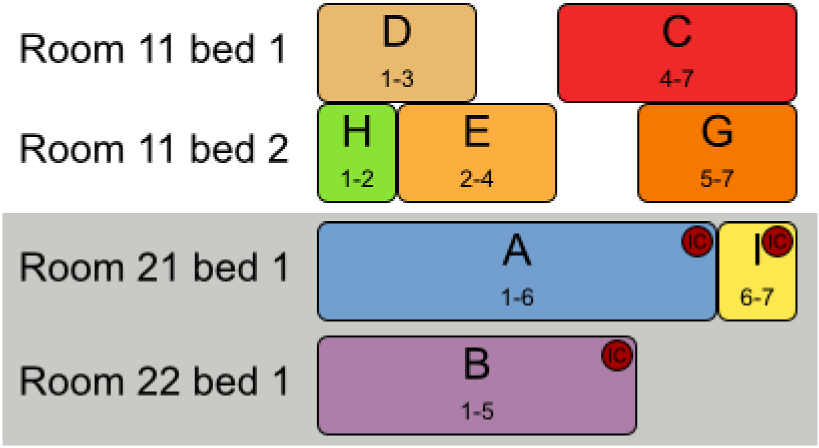
> atoms in the universe if every atom is a universe of atoms?

(10^80)^80 = 10^6400



Source: NASA and ESA (wikipedia)

Needle in a haystack

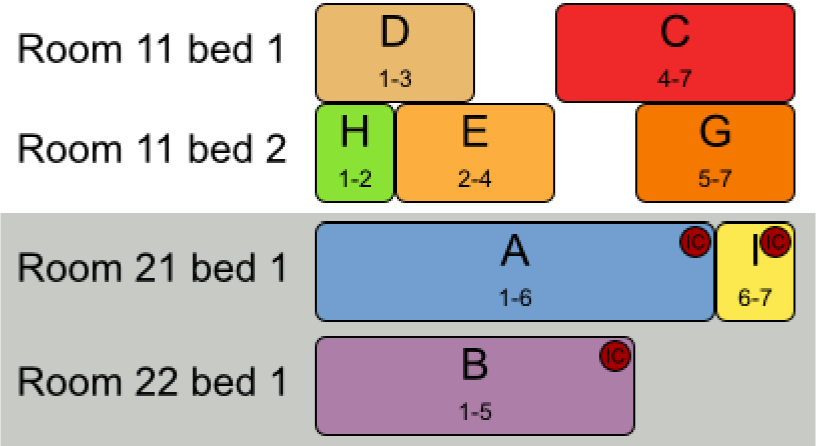


How many possible solutions?

- 310 beds
  - in 105 rooms
  - in 4 departments
- 84 nights
- 2750 patients (admissions)

A little over 10^6851

Do the math



1 patient

310 beds

310 ways to schedule 1 patient

2 patients

310 * 310 = 96 100

3 patients

310 * 310 * 310 = 29 791 000

2750 patients

310 * 310 * ... * 310

310^2750

= a little over 10^6851

A little over 10^6851

```
17565400009613647714189309909847019528176031676612802408947467896773536824694320
25377105894429532838896584732142182052033826938421876324335318655320329177415624
34019697177044291997231955740197313574392215860109009013187756616189881491857442
89903563474940721372704649669447515706252843900344472960295273324597346780837674
88697227533874455473777198677633776549677388281039595554235192833141437843928340
51328923587442242154997922875864744269141114570273352582691671031946927906652177
66902628732489519488236496931246157419949856522543510386164584826962224171420152
25565850232385350464349705325292272792440640617915932702527496869105242581827012
171067055264183750343510879448196106242200027292887379804041858752339124281780095
58605680683578864680145557998942327371019832139824665975180911386722774004539981
34278552385168663637443267160148549642311093017595330858041676796683206809536596
46569395830894437089458443238882278162963824641222099807169369905394994720275907
38400791514217875461942733015467480308665074008529461146577114465977072581447925
88212290682716057000722801705649967418814850790871678616492253646749058716362694
56894529270615321506374546152336416456127910274106084164763806424690851439804640
67453971429400369608313929289399595696359958354101721624055729520838609453039985
59272628937624385694142906376790391997713872443251360270344814604597056658507680
95764769840369232215532708782795742398666571567290512950859701002128560257873450
34666683590797377984104207041334053480226788236770435009014979334517769826461063
28117889455452701285996652060625309878869936718080636701237289582496335799276496
97991233610444564616874109815224930933169104649370892996558804470140748763978122
10684054337706530175130912335567383560436635091489083375525519539844805718811296
85807650768511219249940528633766810704609502399987122465510787717422067936021241
05730014911043812216621387647568988345883813404108921585448372002290085339308167
94663631470201197595487045022087615873490295940409113638894083753062801416644858
```

A little over 10^6851

```
70757942487218045035508810158461046502812782292385633846174494690914238485407798
37976573852840248463244968564240141089763763217269495446550923090738150172870668
68402081731644263484686141346509306107283418762923467113106773326485539514229919
89751468506508069513397904821612697383659788320905691994864296149528670873271380
18006650770249052559419638332728972636228024127885657959189574420249964658137384
98299648678707389648424263725804209284739296524664530660893065815727451390562561
81505240205578741292314133858985615181677968313683876880079649763914095651402272
04777610845144506688836696844897279181666903094579081039689572524839918882203466
75664835223276850061950801785251074912552450542389767056205475823297598574505575
83834141364747838395082871684183199560673441708538602779816347949276957201268066
62737283137076807355893467941027682428304918329951886951690865417997171855081020
26756284570976172802328890960381286165431282000890478132235199027141966946398442
73986565523012106816309974964021700560537928432528631741787455155275823033005847
63710721074772137206636934675415209083984961138990816735390734923436827652228741
07306375553429574524282669680025732278499336914490634182865013110363140489605282
49465982665132492491072491788618403253474529440348798670244615185355092357283764
93638760707623418191667075526942154653033728468983877312232305317179427144435853
36388068489698718841685828476130163980130066330161405037431756112554842734192914
35502775849577615159921009571496639402549077872124227731739936370100132762333353
47390808021571900433548715701062002052376364885669272869945947160341077659253581
65207459958613365778774312937767961242646970494187951860105460569752264242274554
46011031581123438684685838655333776830823720356749120051129394691394743488410658
61354353779545576065515784960221442891508405618131446589507592449826384604047449
56226545521579338675725427458383708893912437023584592443865610814799055455700844
91443709439642235090455604548030317754849613813010298858282615659336373785985294
```

A little over 10^6851

```
58667337266411706925088329158776619790296442059252886388879455197093987039774900
08729398961010317200010000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
```

A little over 10^6851

00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000

- The search space is big!
  - Compare with WWW size
    - 22 020 000 000 pages
- Each possible solution
  - 2750 patients scheduled into 310 beds
  - Still need to calculate the score! **=> Drools Expert**

Algorithms

Operational research is fun.

Brute force? Throw hardware at it?

- Calculate 10^9 scores per ms
  - Impossible today!
  - 31 579 200 000 ms in 1 year
    - < 10^11 ms in 1 year
  - 10^9 * 10^11 scores per year
    - = 10^20 scores per year
- How many years? 10^6851 / 10^20
  - = 10^6831 years
- CPU 1000 times faster
  - It becomes 10^6828 years

Smarter brute force?

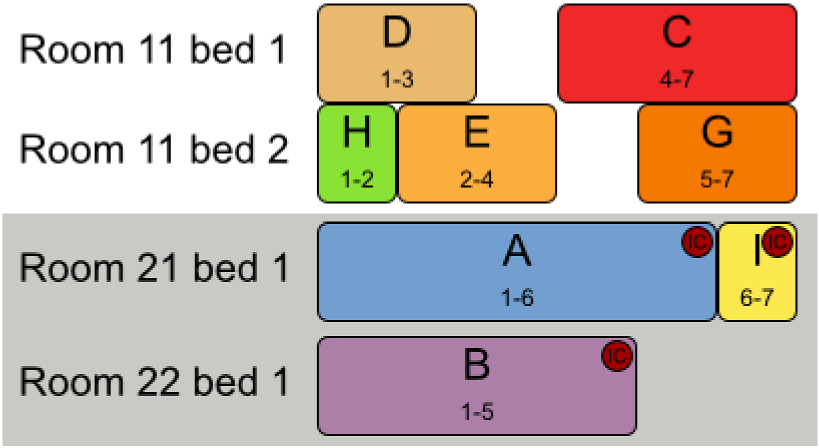
- Eliminate subtrees
  - *Branch and bound*
  - Still too many for loops
  - Still takes forever

```
for (bedOfPatient1 : bedList) {
  patient1.setBed(bedOfPatient1);

  for (bedOfPatient2 : bedList) {
    patient2.setBed(bedOfPatient2);

    if (patient1.shareNightWith(patient2)
         && bedOfPatient1.equals(bedOfPatient2)) {
       continue;
       // bug: best solution might break a hard constraint
    }
    for (bedOfPatient3 : bedList) {
      ...
```

2 patients in the same bed



1 patient

0 *of* 310 (no chance)

2 patients

310 *of* 96 100

= 1 *of* 310

3 patients

620 *of* 29 791 000

= 1 *of* 48 050

2750 patients

310*2750*2749/2 *of* 310^2750

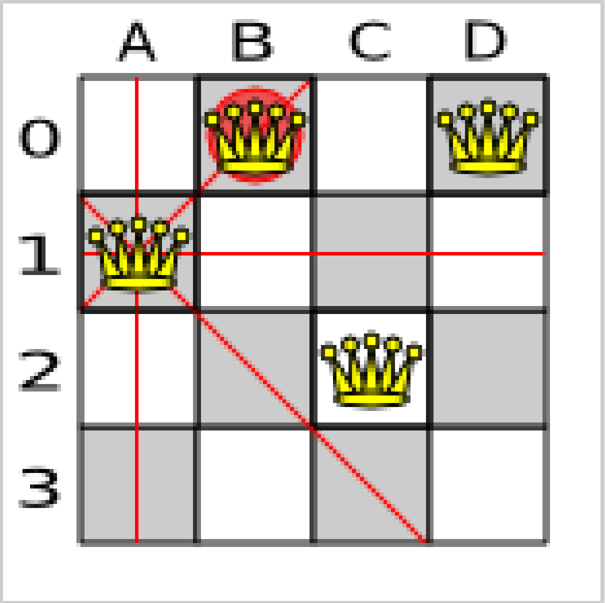
< 1 *of* 310^2740

Imperfect algorithms (mimic a human)

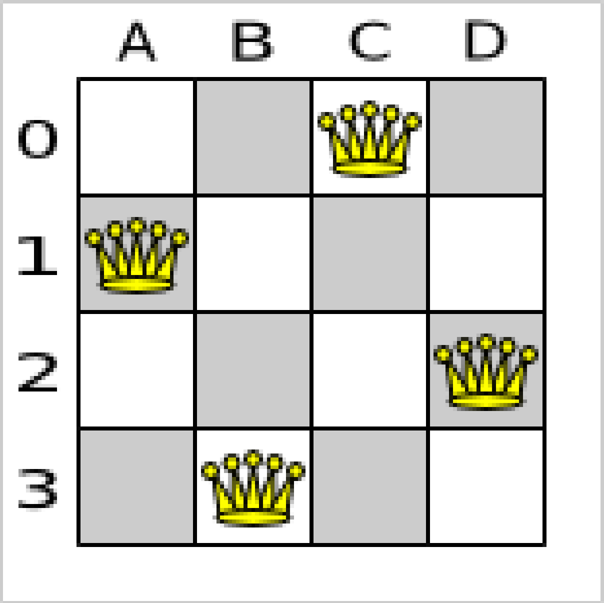
- Deterministic
  - First in, first assigned, never changed
  - Easy to implement
    - Drools Planner score support
  - Fixed time (for example 18 seconds)
- Meta-heuristic
  - Move things around
    - Start from result of deterministic algorithm
  - Drools Planner implementations
  - More time = better score

N Queens: use case

- Place n queens on a n-sized chess board
- No 2 queens can attack each other
  - Score -1 for every 2 queens that can attack each other



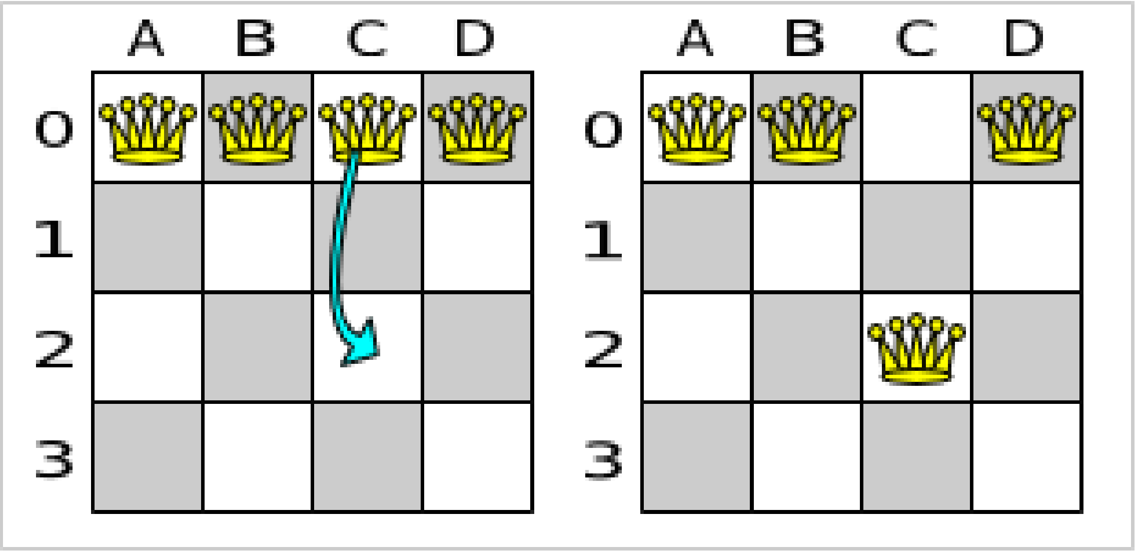
Score = -2



Score = 0

Move things around

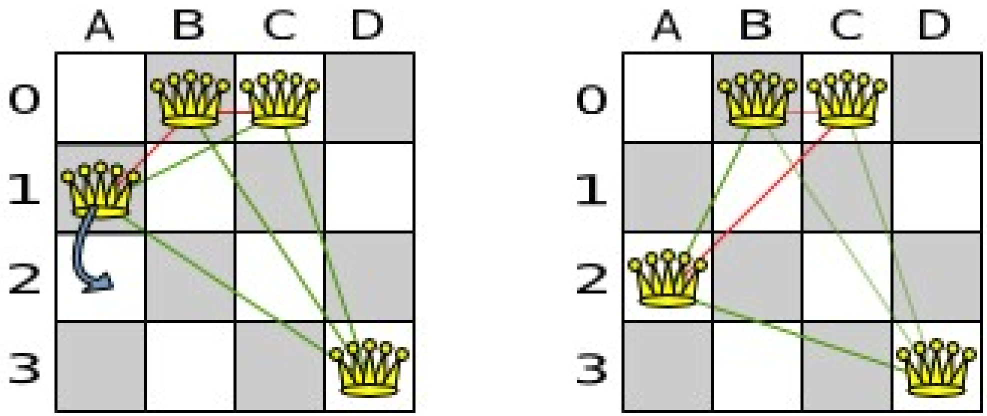
- Move = from solution A to solution B
  - Change the row of 1 queen



Score = -6   Score = -4

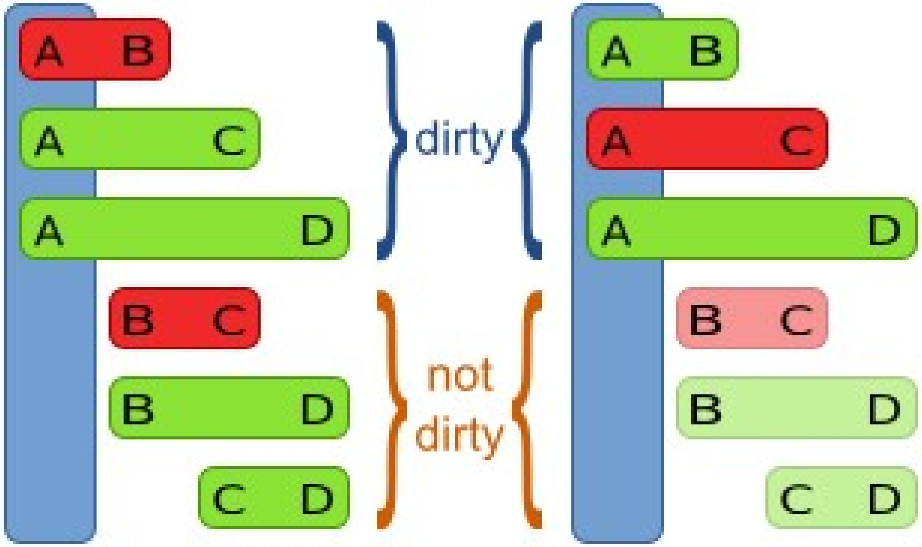
  - Give 2 queens each others rows
  - ...

Thank you statefull rule engine!



Delta based score calculation

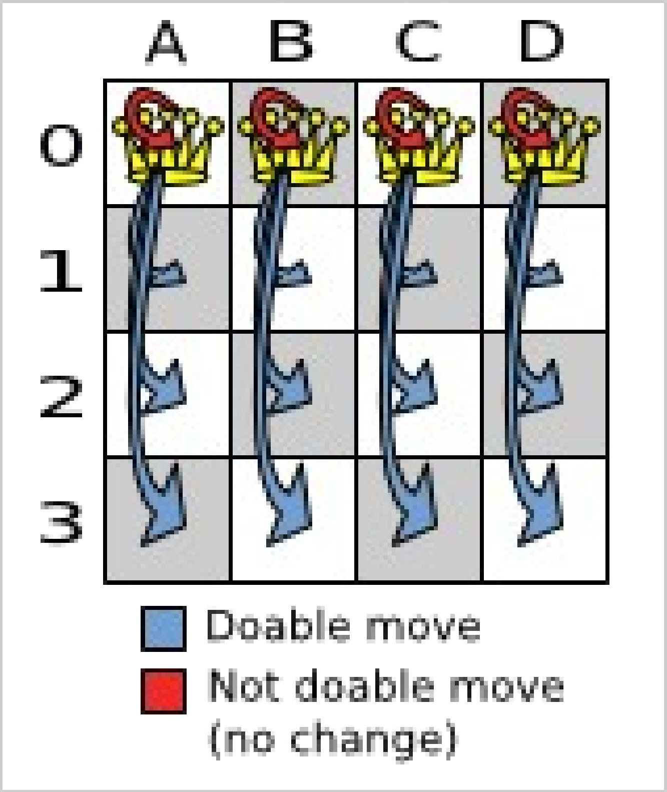
The rule engine (with forward chaining) only recalculates dirty tuples.



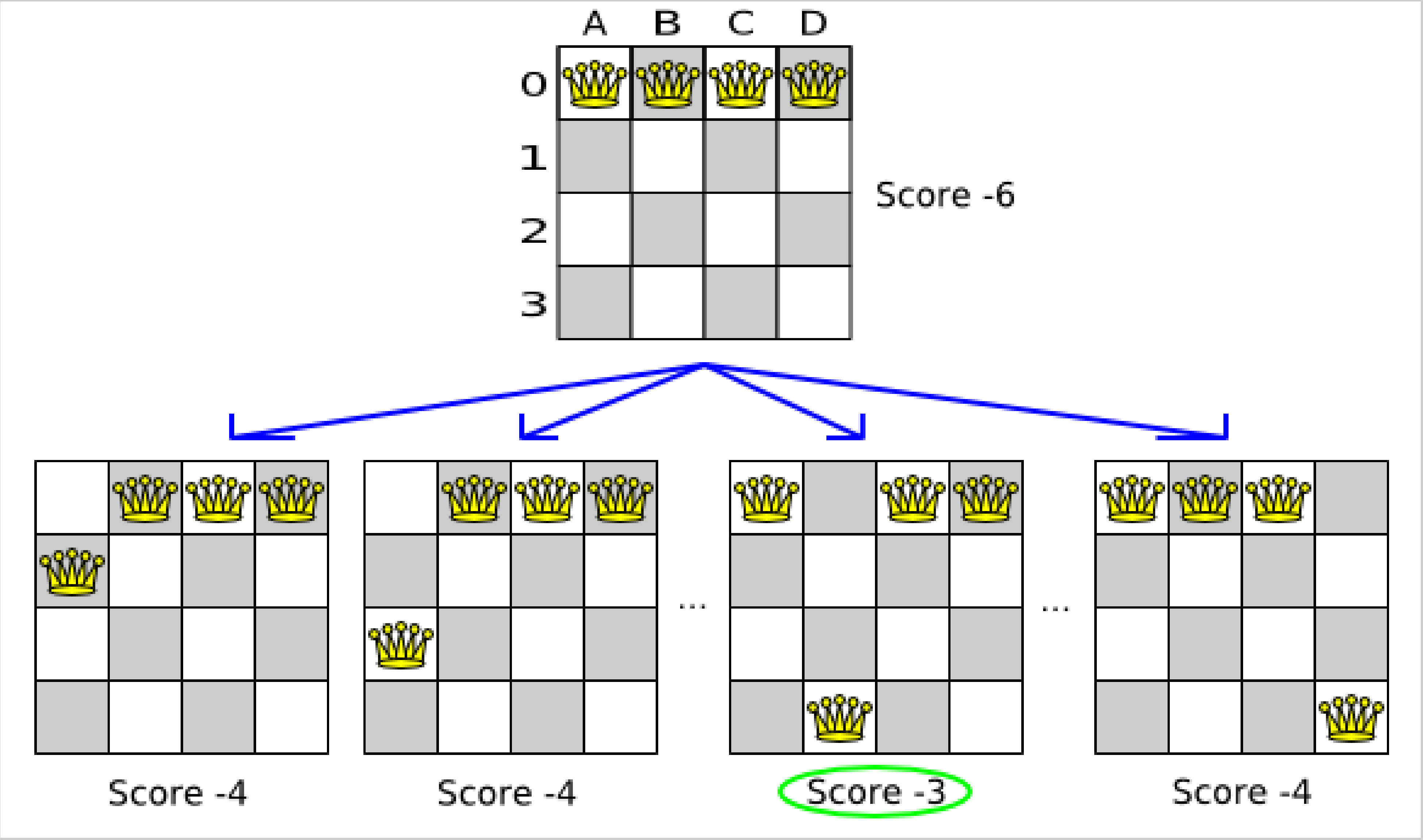
queens	dirty	total	speedup
4	3 of	6	time / 2
8	7 of	28	time / 4
16	15 of	120	time / 8
32	31 of	496	time / 16
64	63 of	2016	time / 32

All moves from one solution

- Number of moves < number of solutions
  - N queens
    - n*n < n^n
  - 4 queens
    - 16 < 256
  - 8 queens
    - 64 < 16777216
  - 64 queens
    - 4096 < 10^116

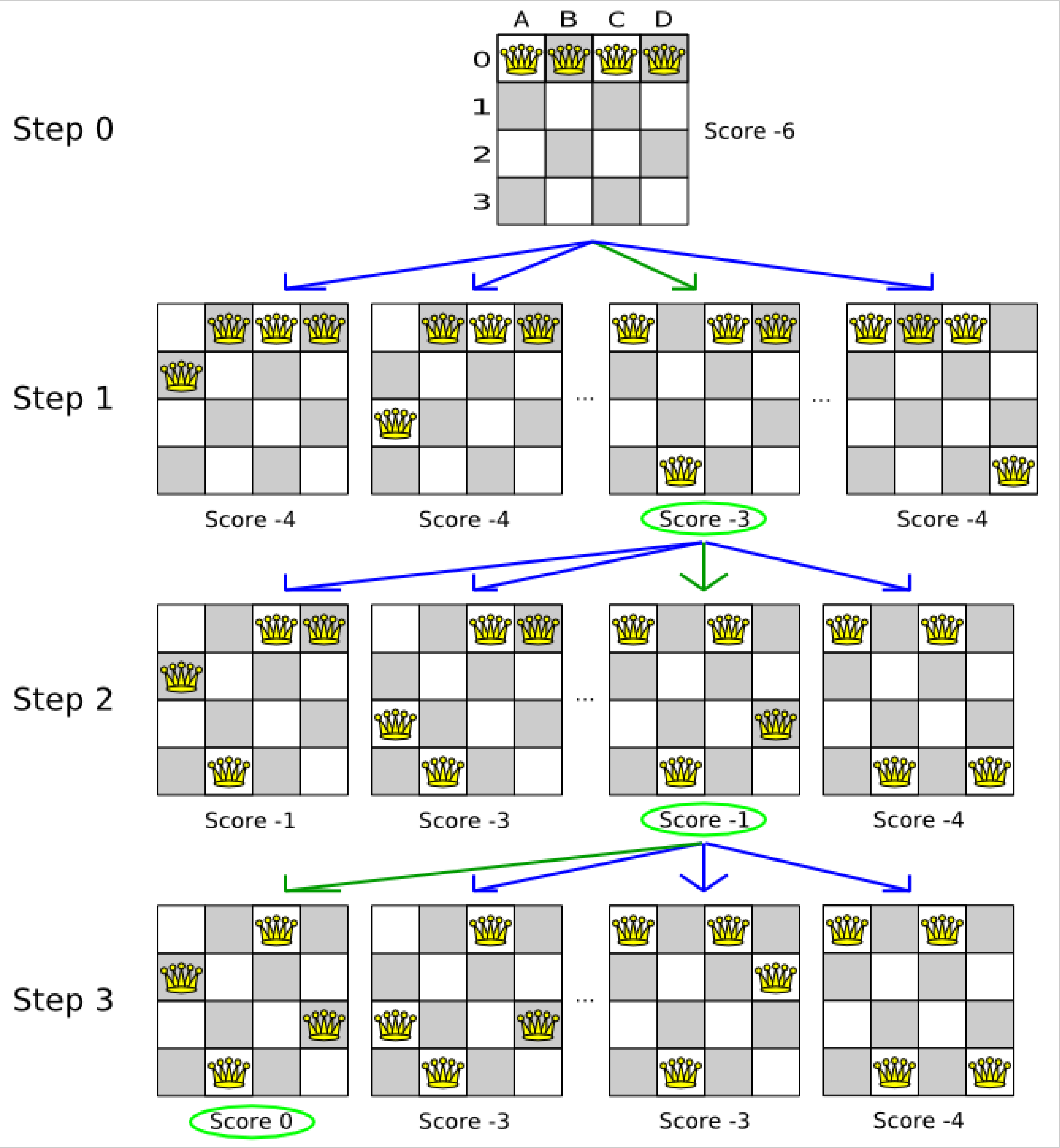


Local search 1/2



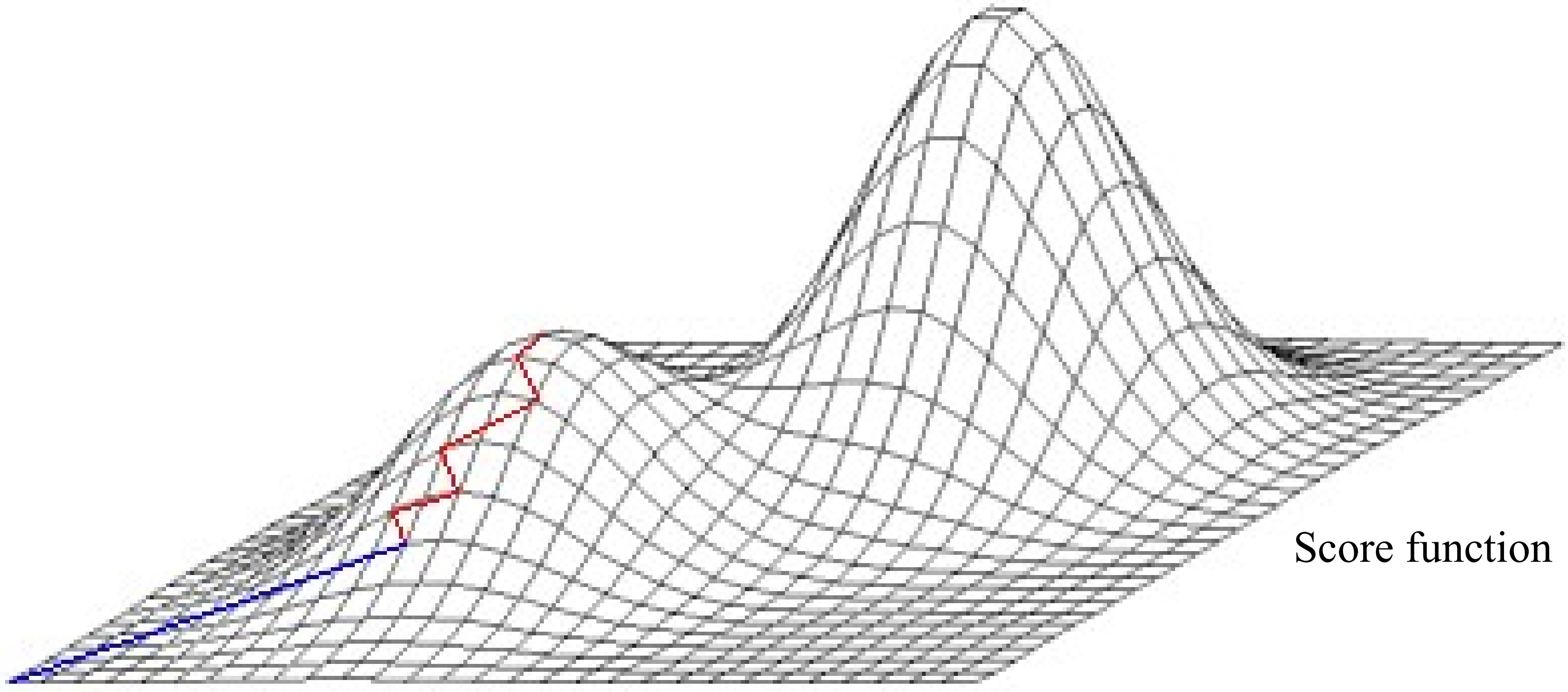
Local search 2/2

- Search path
  - Not a tree



Local optima

- 1) Deterministic StartingSolutionInitializer
- 2) Simple local search
- 3) Stuck in local optimum!



Source: Wikipedia

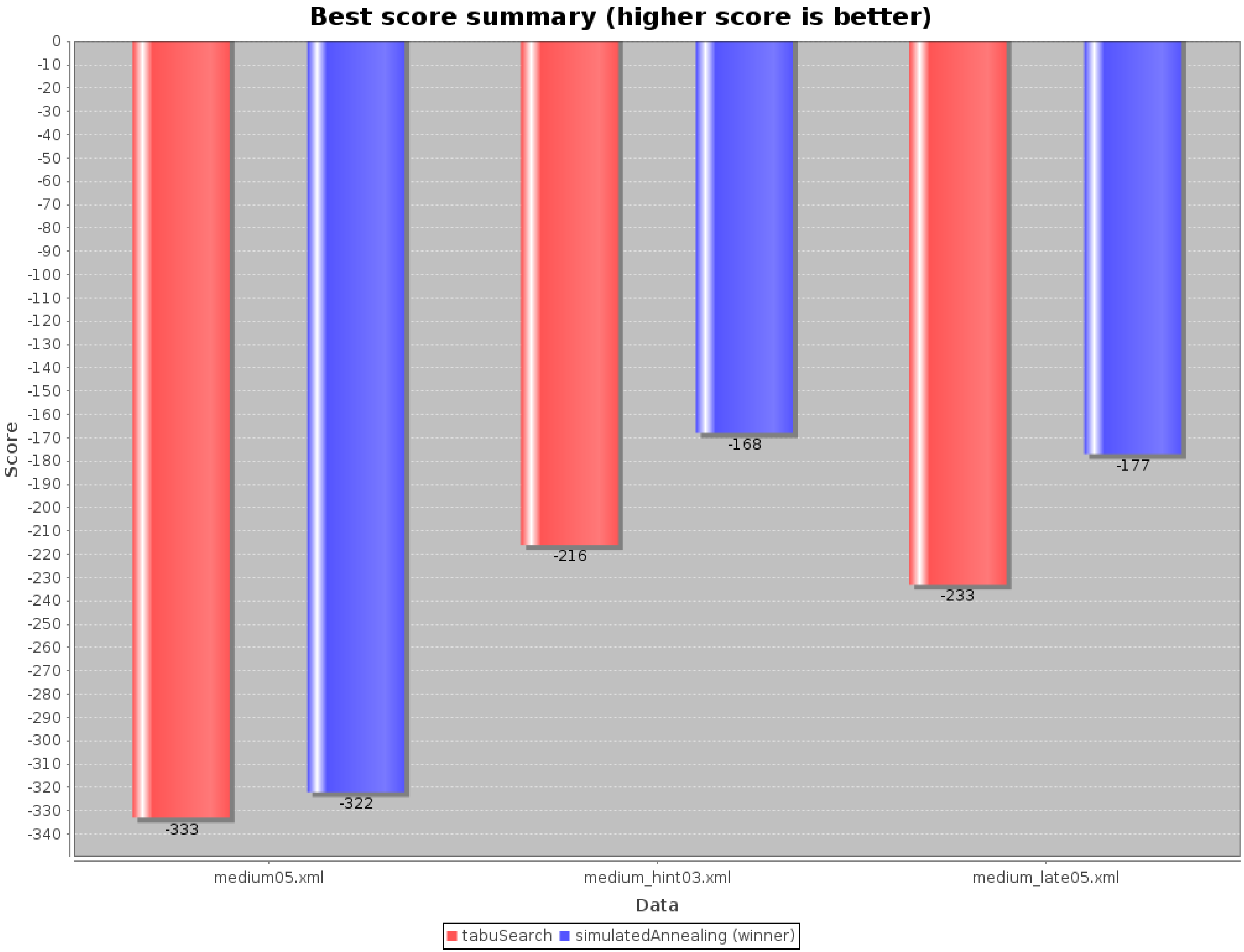
Local search++

- **Tabu Search**
  - Solution tabu (high tabu size)
    - Been there, no need to go there again
  - Move tabu (low tabu size)
    - Done that recently, no need to do that again
  - Property tabu (low tabu size)
    - Changed that recently,
      no need to change that again
- **Simulated annealing**
- Great deluge, late acceptance, …
- Hyper heuristics

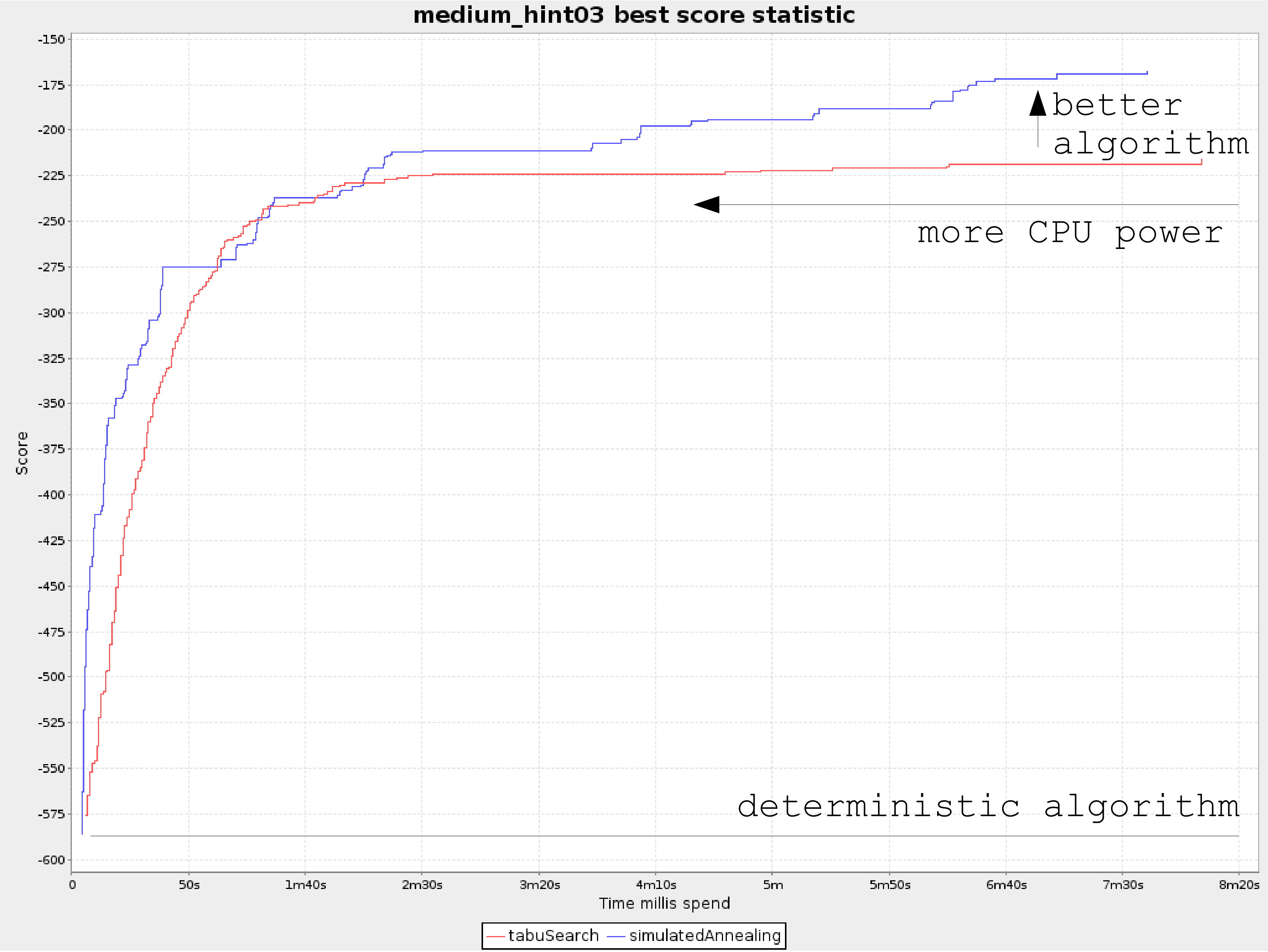
Benchmarker

Measure, don't guess.

Benchmarker utility

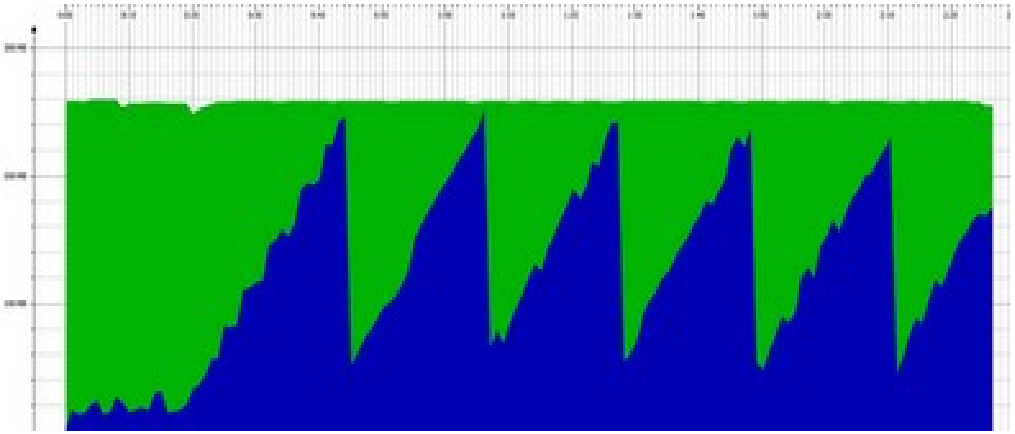


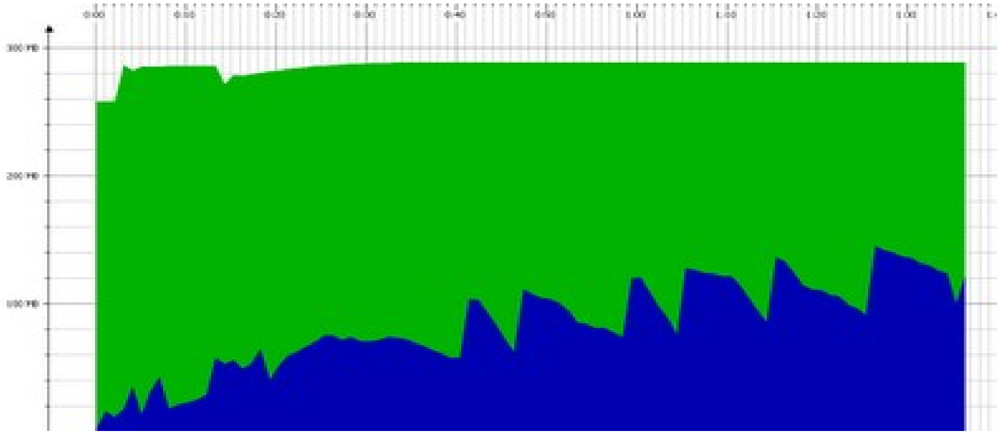
CPU power VS algorithms



Free speed upgrades from the rule engine

- Differential update (AKA *true modify*)
  - Drools 5.0: update = retract (remove) + assert (insert)
  - Drools 5.1: *real* update (released in Q3 2010)
    - Uses less memory and reduces garbage collector stress
    - Improves performance
  - Update is mostly used in statefull environments


Statefull memory drools 5.0
with Drools Planner 5.1


Statefull memory drools 5.1
with Drools Planner 5.1

Summary

Summary

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

Q & A

Questions?

Useful links

- Website
  - http://www.jboss.org/drools/
- Reference manual
  - http://www.jboss.org/drools/documentation.html
- Blog
  - http://blog.athico.com/
- Mailing lists (forum interface through nabble.com)
  - http://www.jboss.org/drools/lists.html