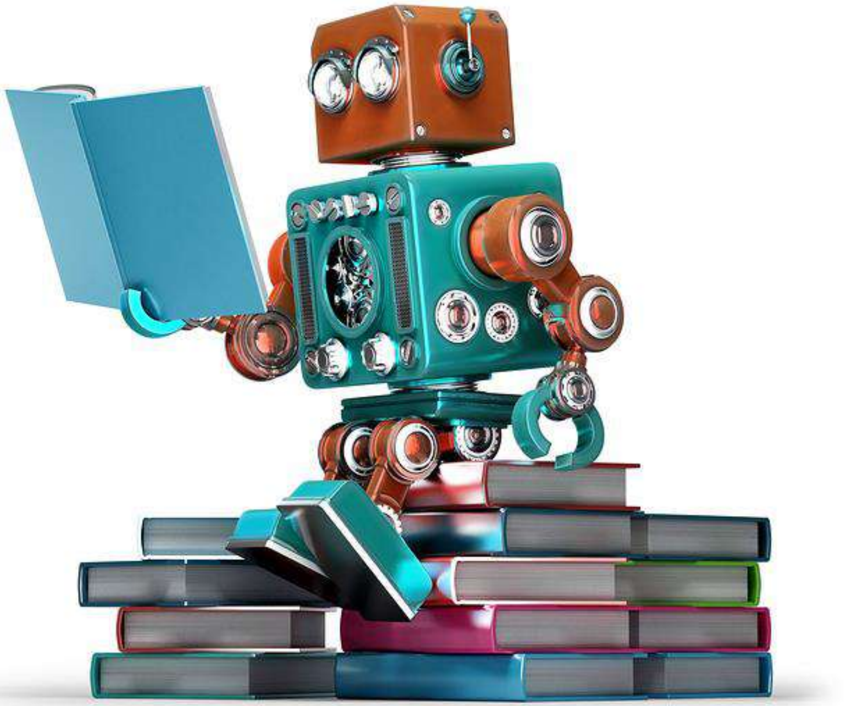
REASONING SYSTEMS
DAY 2



https://robohub.org/wp-content/uploads/2016/11/bigstock-Retro-Robot-Reading-A-Book-Is-110707406.jpg

DAY 2 AGENDA

2.1 Informed Search Techniques (part 2/2)

2.2 Search Based Intelligent Systems

2.3 Course **Assessment** 1

2.4 Search Reasoning **Workshop**

DAY 2 TIMETABLE

No	Time	Topic	By Whom	Where
1	9 am	2.1 Informed Search Techniques (part 2/2)	GU Zhan (Sam)	Class
2	10.10 am	Morning Break		
3	10.30 am	2.2 Search Based Intelligent Systems	GU Zhan (Sam)	Class
4	12.10 pm	Lunch Break		
5	1.30 pm	2.3 Course Assessment	All	Class
6	3.10 pm	Afternoon Break		
7	3.30 pm	2.4 Search Reasoning Workshop	All	Class
8	4.50 pm	Summary and Review	All	Class
9	5 pm	End		

2.1

INFORMED SEARCH TECHNIQUES

(PART 2/2)

2.1 INFORMED SEARCH TECHNIQUES (2/2)

- Use Heuristics
- Hill Climbing Search (HC)
- A Star Search (A*)
- Tabu Search (TS)
- Simulated Annealing (SA)
- Informed Search Use Case

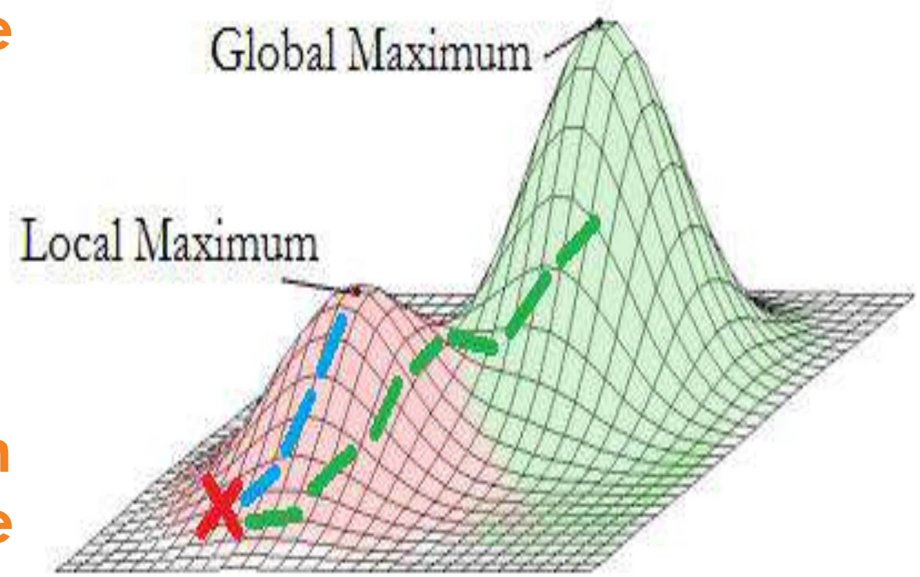
https://modernmarketingtoday.com/wp-content/uploads/2013/02/search-marketing.jpg

2.1 INFORMED SEARCH TECHNIQUES (2/2)

Sub-optimal Result

Greedy search algorithms suffer from the local minimum / maximum problem.

- Hill climbing grabs a visible best neighbour state without thinking too farther ahead; and without "looking back" either.
- When search reaches the vicinity of a local maximum (sub-optimal goal found), it will then stop searching: getting stuck at local optimum.
- A possible strategies to escape a local optimum is to allow non-improving "downhill" but feasible moves. However, non-improving moves will sometimes lead to infinite search cycling (repeated search states) unless provision is added to prevent repeating solutions.



https://sites.google.com/site/practicavectorial/3-parcial-1/contenido-opcional

2.1 INFORMED SEARCH TECHNIQUES (2/2)

Avoid Repeated States

- **For many problems, repeated state are unavoidable, such as route-finding problems.**
  - The search trees for these problems become infinite.
- **There is a fundamental tradeoff between space and time**
  - Algorithm that forgets its history is doomed to repeatedly visit sub-optimal search states (candidate solution): **Time problem**
  - Algorithm that remembers every search states (candidate solution) that it has visited: **Space problem**

2.1 INFORMED SEARCH TECHNIQUES (2/2)

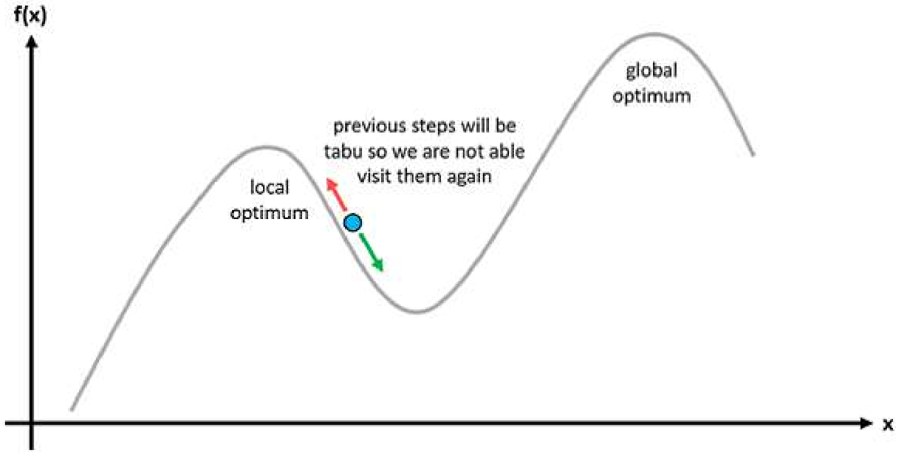
Avoid Repeated States

- **Tree-search algorithms can be modified to include extra search process variables / data structure**
  - **Closed List** : Stores every expanded node
  - **Open List** : Stores the fringe of unexpended nodes
- **If the current node matches a node on the Closed List, it is discarded instead of being expanded**
- **Improving searches are algorithms that begin at a feasible solution, and then seek for even-improving evaluation/objective function value.**
  - Tabu Search
  - Simulated Annealing
  - Genetic Algorithms

2.1 INFORMED SEARCH TECHNIQUES (2/2)

Tabu Search (TS)

- **Tabu search tackles cycling by temporarily forbidding moves that would return to a solution recently visited.**
- **It is an enhanced variant of Hill Climbing.**
- **How it works**
  - Having a **Tabu List** record forbidden moves, and each iteration chooses a non-tabu feasible move.
  - After each step, a collection of moves that includes any returning immediately to the previous point is added to the tabu list, e.g. no such move is allowed for **a few iterations** though eventually all are removed from the **Tabu List** and again available.

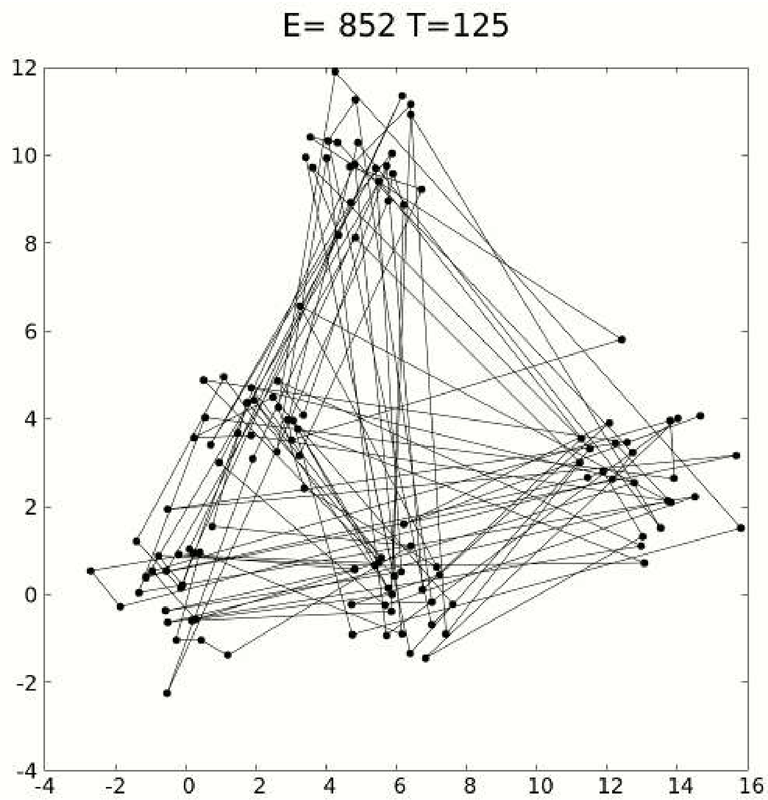


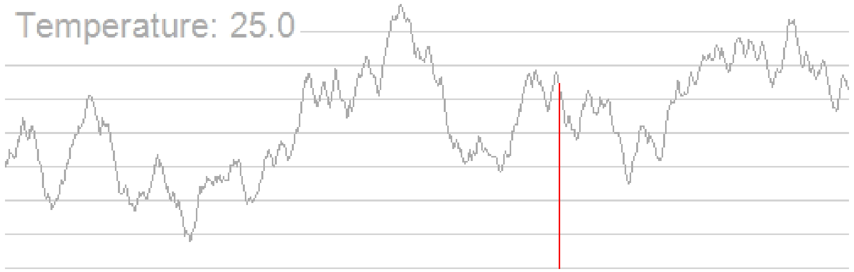
http://www.globalsoftwaresupport.com/tabu-search/

2.1 INFORMED SEARCH TECHNIQUES (2/2)

Simulated Annealing (SA)

- **Hill-climbing algorithm never makes "downhill" moves, so can get stuck on a local optimum.**
  - However, a purely random walk is "**complete**", meaning global optimum is **ensured**.
  - But randomly exploring states for global optimal solution is very inefficient.
- **Simulated Annealing search**
  - Combines hill climbing with a random walk.
  - Instead of picking the best move, it picks a **random move/node/state**. If the move improves the situation, it is always accepted, otherwise the algorithm accepts the worse move with probability lesser than a threshold (**Temperature**, e.g. from100% hot to 0% cool).
  - It has been applied widely to factory scheduling and other large-scale optimization tasks.





https://en.wikipedia.org/wiki/Simulated_annealing

2.1 INFORMED SEARCH TECHNIQUES (2/2)

Summary of Search Strategies

- **Uninformed (Brute Force / Blind) Search**
  - Evaluate all possible solutions in a systematic way until a valid solution is found
  - No guarantee of optimality of the first solution - to find a good solution, search for more valid solutions & pick best
  - Impractical for highly combinatorial complex problems
  - Algorithms: Breadth first, Depth first, Backtracking, Uniform cost search, etc.
- **Informed (Heuristic) Search**
  - Make use of heuristics and human expertise to focus the search in the most promising directions
  - The first solution found is (hopefully) moderately optimal
  - Algorithms: Hill climbing, A*, Tabu, Simulated annealing, Backtracking, Constraint propagation, Genetic algorithms, etc.

2.1 INFORMED SEARCH TECHNIQUES (2/2)

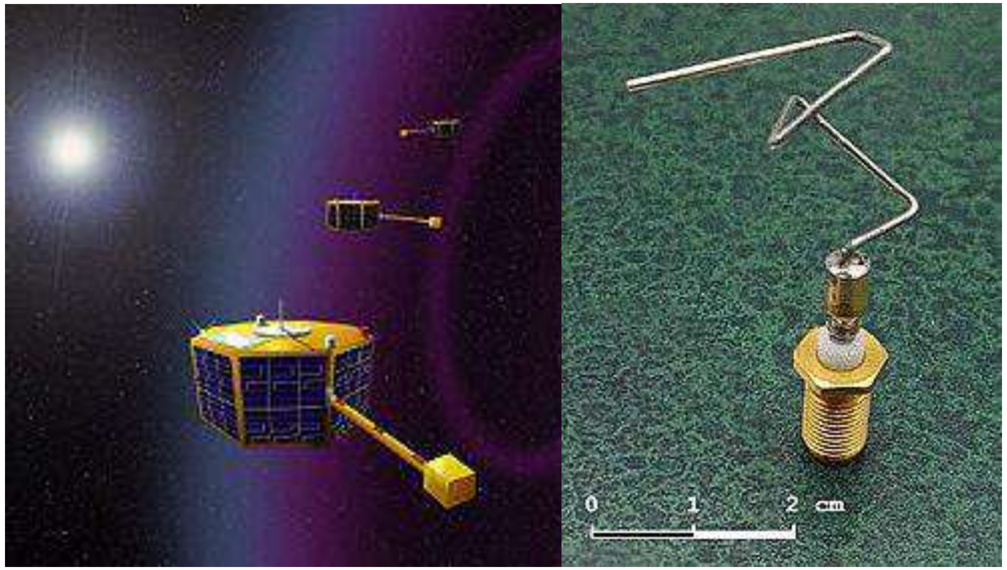
Summary of Search Strategies

- **Uninformed Search**
  - "Blind" but systematic
  - Need complete problem space description
- **Informed (Heuristic) Search**
  - Less complete but more efficient
  - Need extra information / knowledge for search
- **Informed (Heuristic) Search + Randomness**
  - In-between heuristic & uniformed
  - Need extra search parameters, e.g. Simulated Annealing probability threshold
    Tabu list size

2.2
SEARCH BASED INTELLIGENT SYSTEMS

2.2 SEARCH BASED INTELLIGENT SYSTEMS

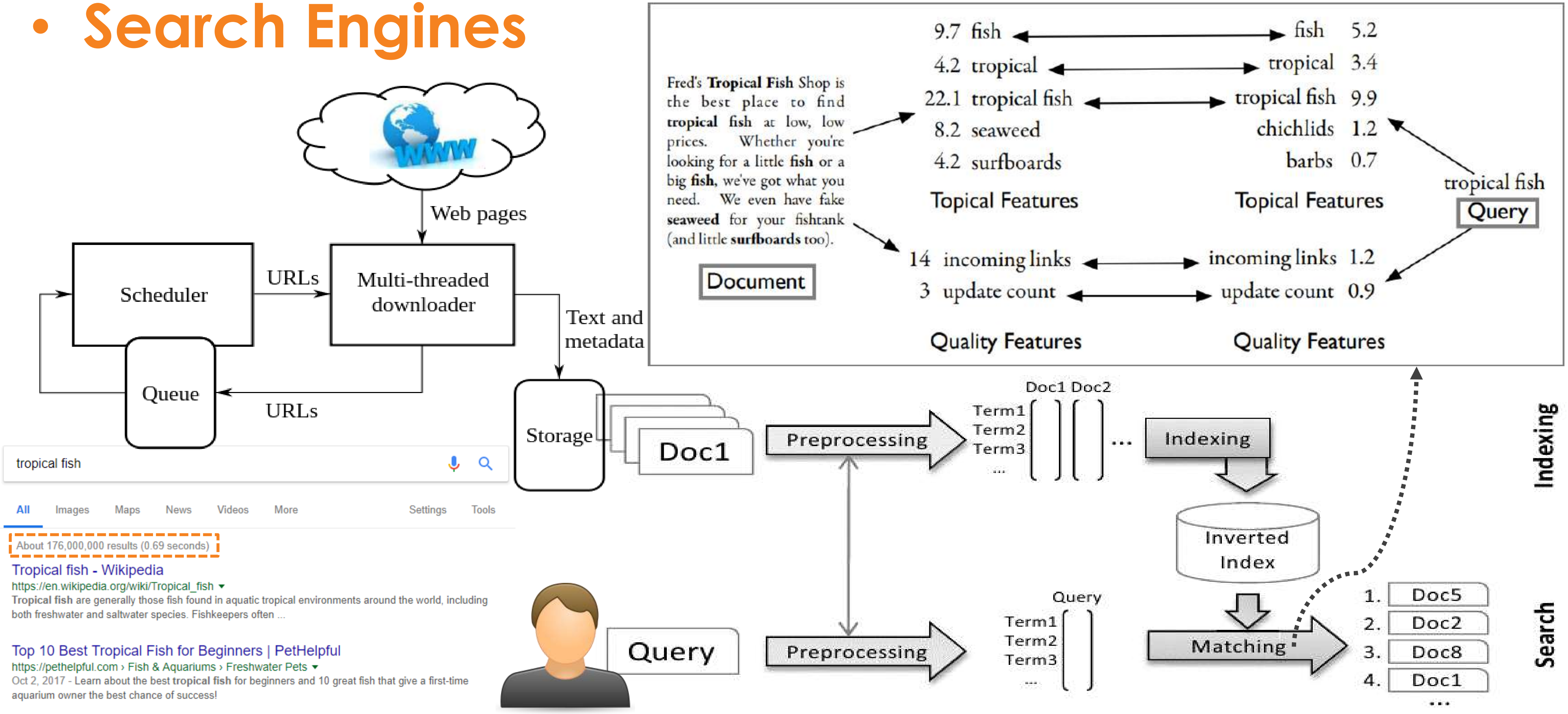
- **Search methods are useful for business resource optimization where the aim is to find an optimal solution among feasible solutions, according to an objective / evaluation / score function.**
- **Many business scenarios:**
  - Airport gate assignment
  - Integrated-circuit design
  - Factory-floor layout
  - Vehicle routing
  - Antenna design
  - Choose a project group…



The 2006 NASA ST5 spacecraft antenna. This complicated shape was found by an evolutionary computer design program to create the **unusual/best** radiation pattern.

2.2 SEARCH BASED INTELLIGENT SYSTEMS

- ## Search Engines



http://ciir.cs.umass.edu/downloads/SEIRiP.pdf
https://www.researchgate.net/figure/1-Information-Retrieval-model_fig1_286513038
https://news.dk/2013/5/2/9/4sbv4df1.jpg?quality=75&progressive=true&width=480&height=308&mode=crop-up&cropUpZoom=true

2.2 SEARCH BASED INTELLIGENT SYSTEMS

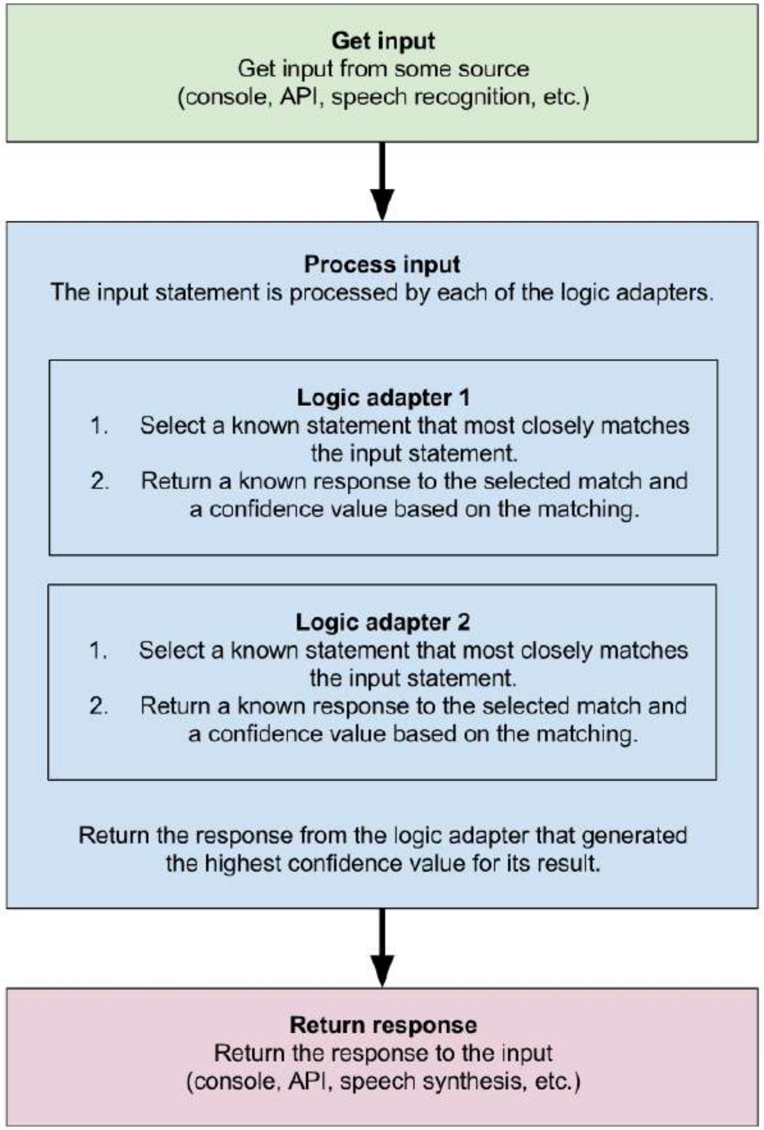
About ChatterBot

ChatterBot is a Python library that makes it easy to generate automated responses to a user's input. ChatterBot uses a selection of machine learning algorithms to produce different types of responses. This makes it easy for developers to create chat bots and automate conversations with users. For more details about the ideas and concepts behind ChatterBot see the :ref:`process flow diagram <process_flow_diagram>`.

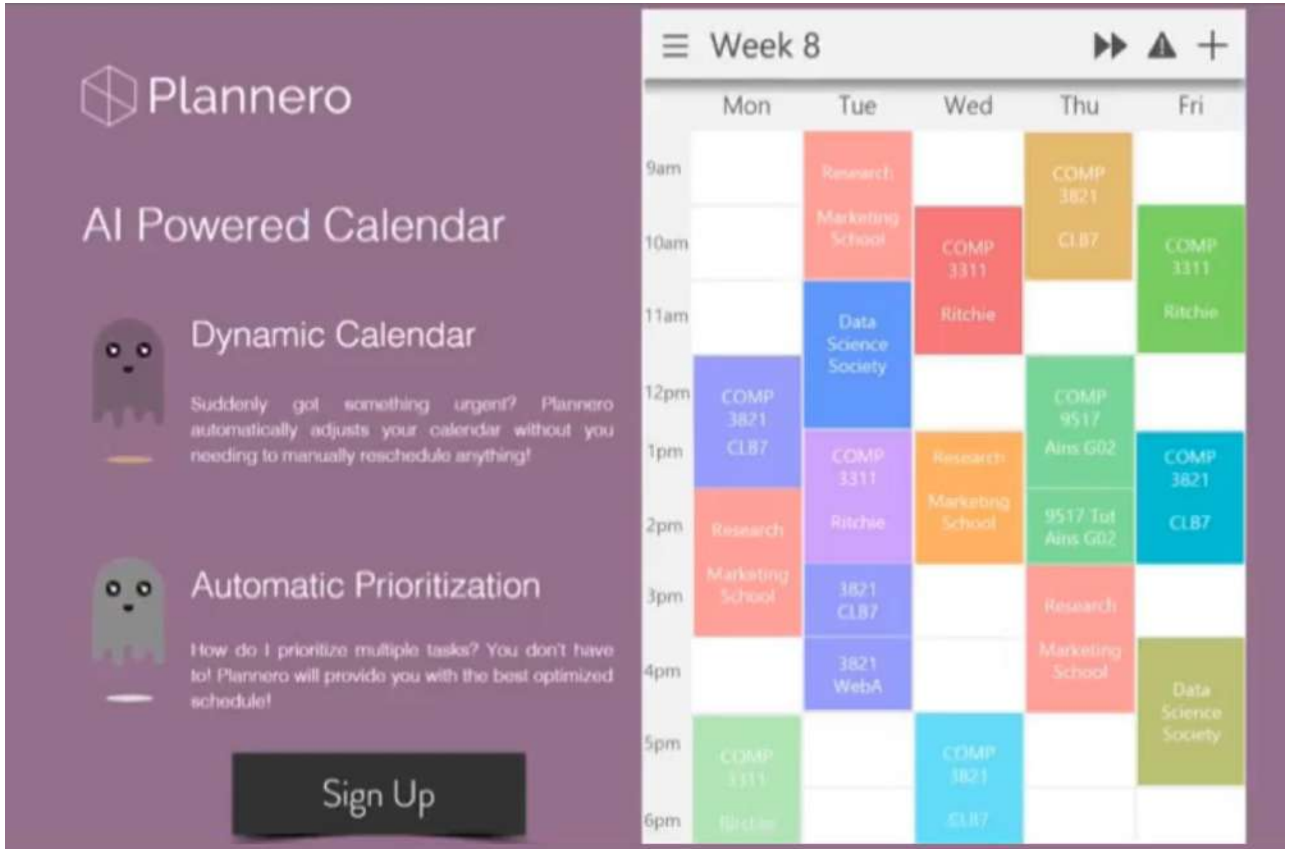
An example of typical input would be something like this:

```
user: Good morning! How are you doing?
bot:  I am doing very well, thank you for asking.
user: You're welcome.
bot:  Do you like hats?
```

https://github.com/gunthercox/ChatterBot



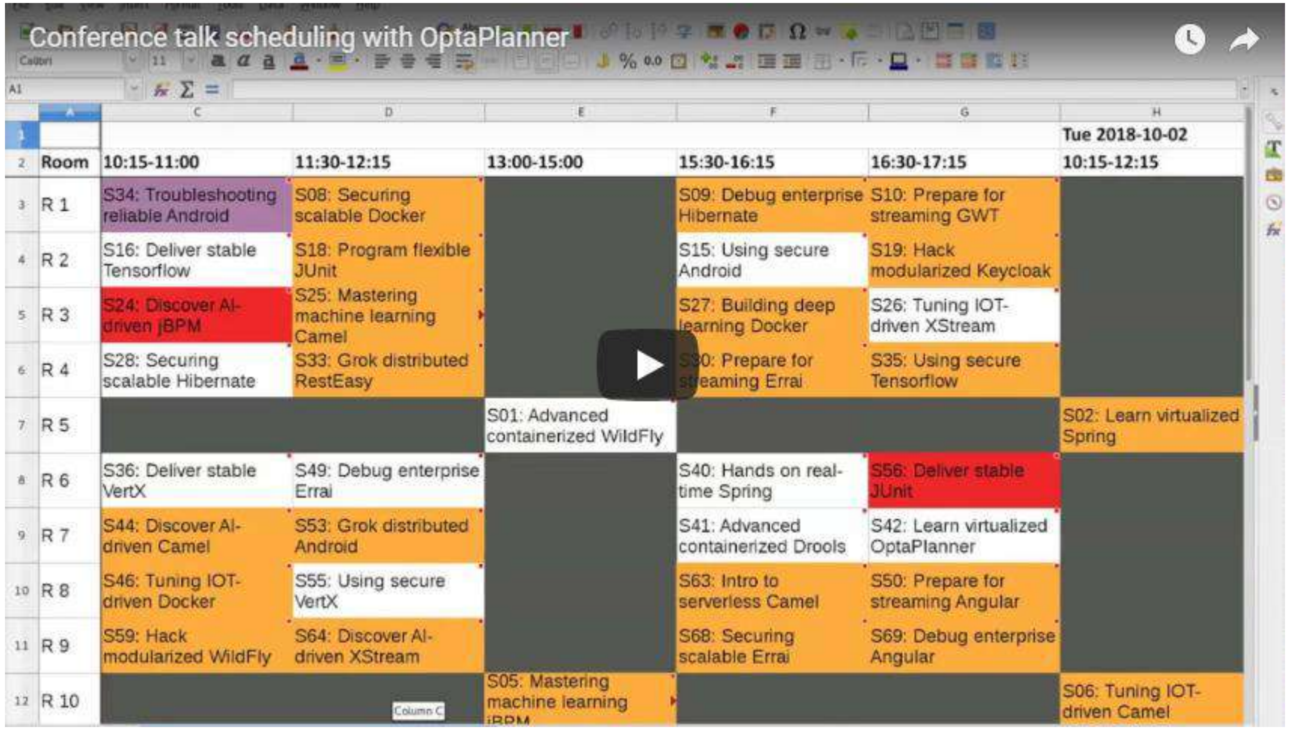
- Planning: Task & calendar management



http://www.plannero.com.au/

2.2 SEARCH BASED INTELLIGENT SYSTEMS

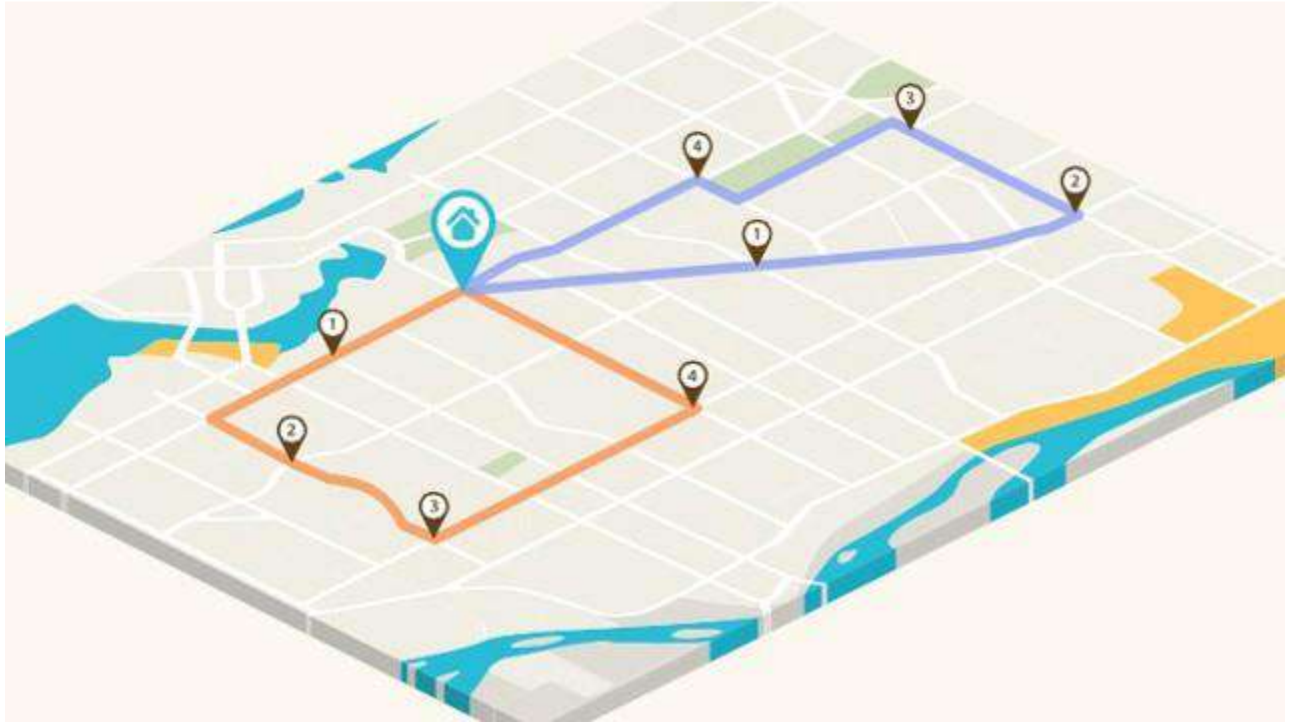
- **Planning: Task & calendar management** (open source)



Room	10:15-11:00	11:30-12:15	13:00-15:00	15:30-16:15	16:30-17:15	Tue 2018-10-02 10:15-12:15
R 1	S34: Troubleshooting reliable Android	S08: Securing scalable Docker		S09: Debug enterprise Hibernate	S10: Prepare for streaming GWT	
R 2	S16: Deliver stable Tensorflow	S18: Program flexible JUnit		S15: Using secure Android	S19: Hack modularized Keycloak	
R 3	S24: Discover AI-driven jBPM	S25: Mastering machine learning Camel		S27: Building deep learning Docker	S26: Tuning IOT-driven XStream	
R 4	S28: Securing scalable Hibernate	S33: Grok distributed RestEasy		S30: Prepare for streaming Errai	S35: Using secure Tensorflow	
R 5			S01: Advanced containerized WildFly			S02: Learn virtualized Spring
R 6	S36: Deliver stable VertX	S49: Debug enterprise Errai		S40: Hands on real-time Spring	S56: Deliver stable JUnit	
R 7	S44: Discover AI-driven Camel	S53: Grok distributed Android		S41: Advanced containerized Drools	S42: Learn virtualized OptaPlanner	
R 8	S46: Tuning IOT-driven Docker	S55: Using secure VertX		S63: Intro to serverless Camel	S50: Prepare for streaming Angular	
R 9	S59: Hack modularized WildFly	S64: Discover AI-driven XStream		S68: Securing scalable Errai	S69: Debug enterprise Angular	
R 10			S05: Mastering machine learning jBPM			S06: Tuning IOT-driven Camel

https://www.youtube.com/watch?v=R0JizNdxEjU&t

- **Vehicle Scheduling: Delivery routing**

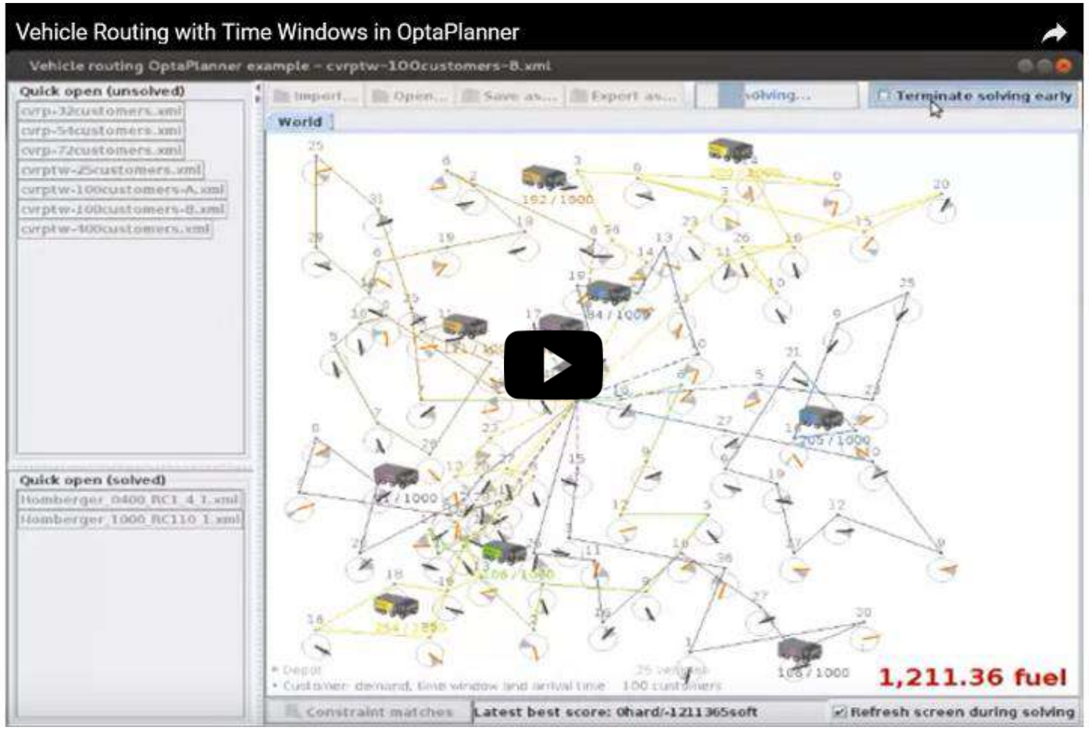


https://routific.com/

Routific Solutions

2.2 SEARCH BASED INTELLIGENT SYSTEMS

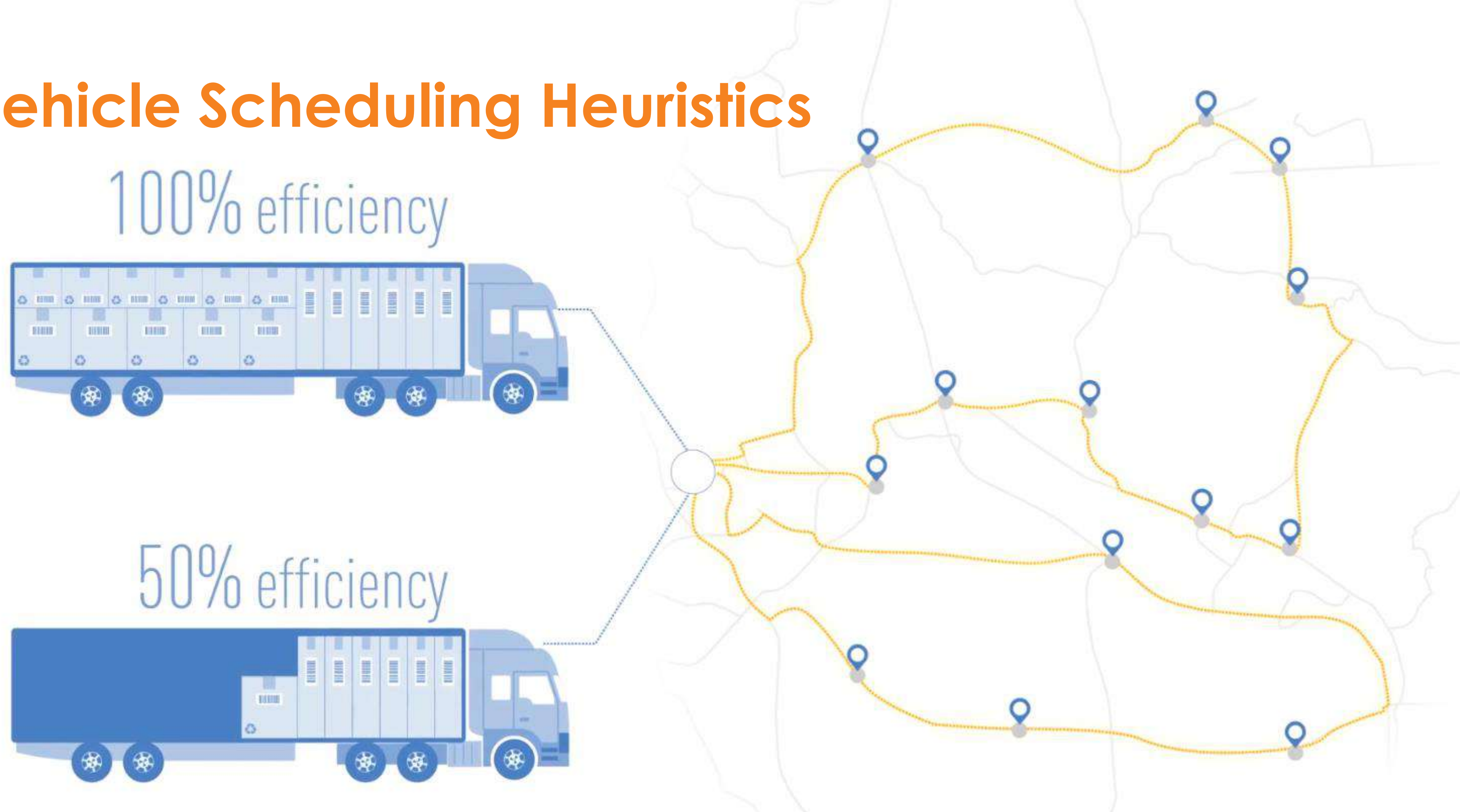
- **Vehicle Scheduling: Delivery routing** (open source)



https://www.youtube.com/watch?v=BxO3UFmtAPg

2.2 SEARCH BASED INTELLIGENT SYSTEMS

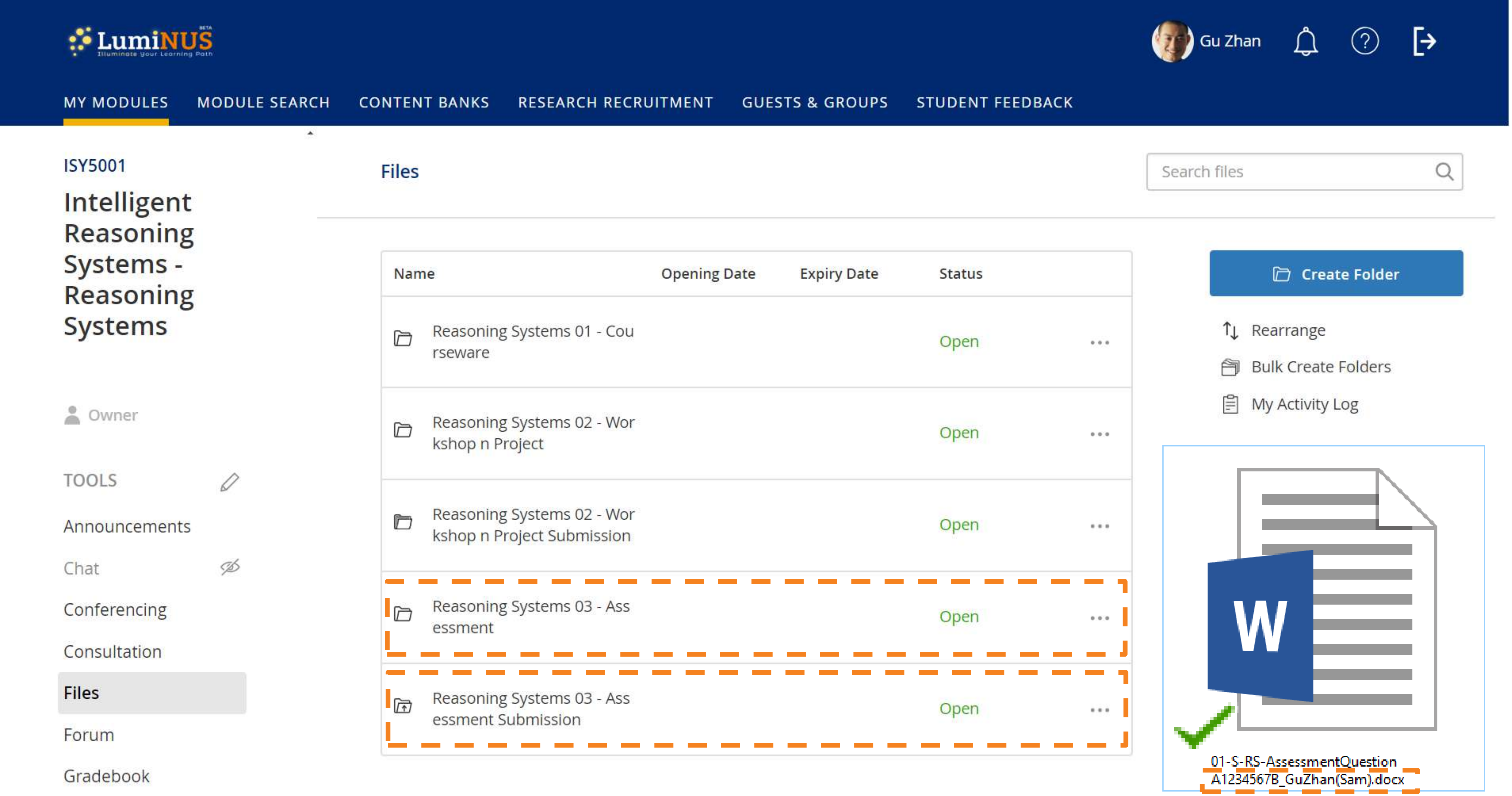
- **Vehicle Scheduling Heuristics**



https://www.youtube.com/watch?v=2O77mpvJU1A

2.3

COURSE ASSESSMENT



Upload word, pdf or zip file to LumiNUS (one single file per participant)

2.4 WORKSHOP

SEARCH REASONING

2.4 WORKSHOP SEARCH REASONING

- **Cloud Balance Solver Deep Dive**
  - Cloud Balance Solver [Java IDE]
  - Cloud Balance Solver [KIE Workbench]
- **Cloud Balance Solver Enhancement**
  - GPU requirements; Data centre physical locations; Network latency, etc.

2.4 WORKSHOP SEARCH REASONING

KIE OptaPlanner Deep Dive – Cloud Computer Balancing

- **Business Scenario / Problem Description**

2.4 WORKSHOP SEARCH REASONING

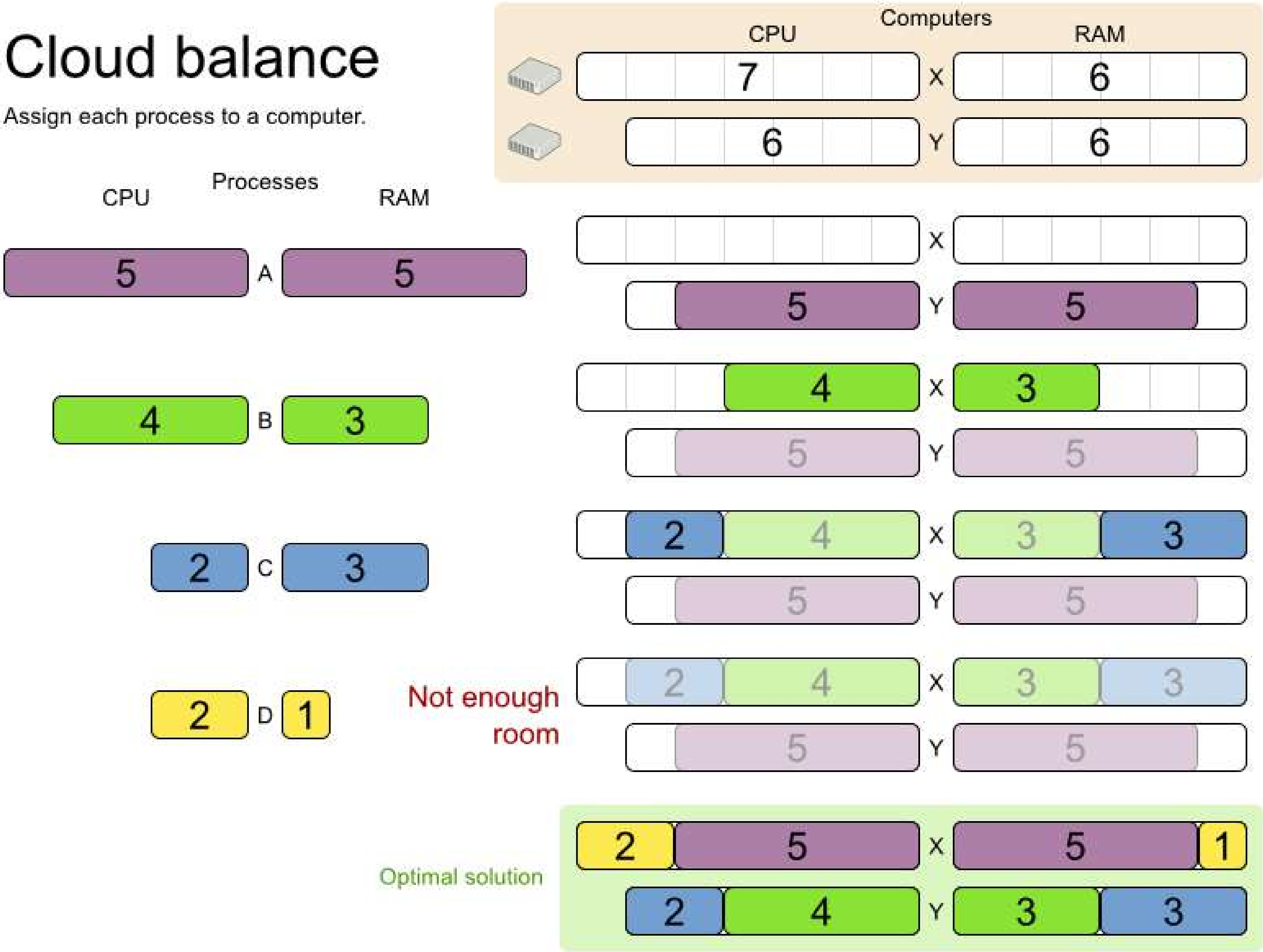
KIE OptaPlanner Deep Dive – Cloud Computer Balancing

- Business Scenario / Problem Description
- A cloud service provider owns a number of cloud computers and needs to run a number of customers' processes on those computers. Assign each process to a computer.

Link https://docs.optaplanner.org/latest/optaplanner-docs/html_single/index.html#cloudBalancingTutorial

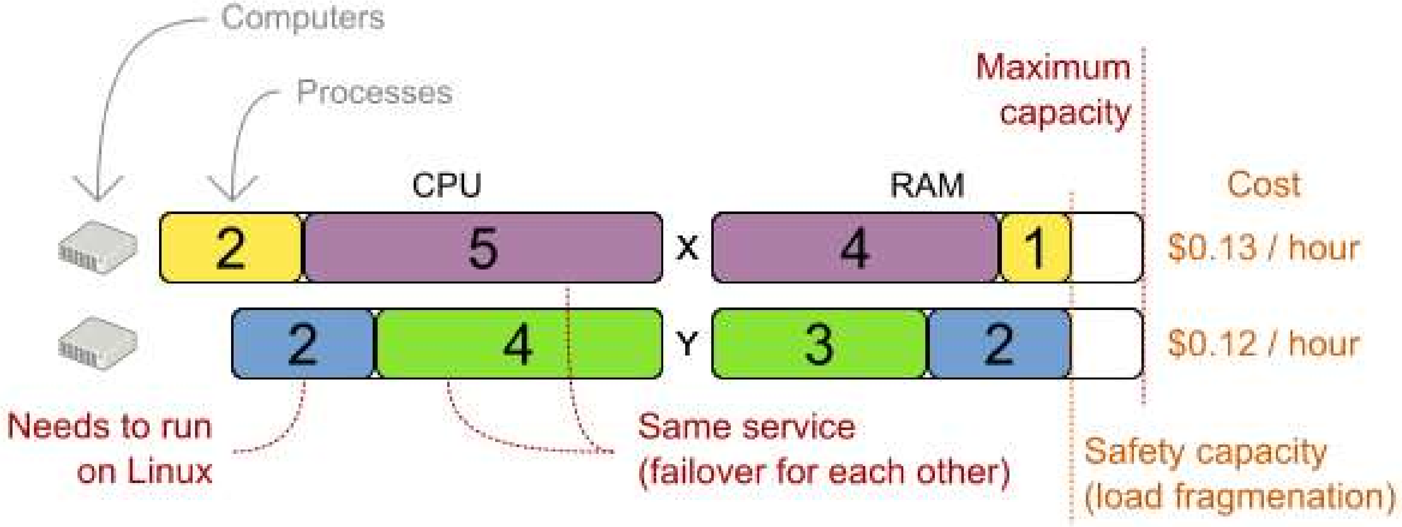
Cloud balance

Assign each process to a computer.



Cloud optimization

Assign processes to machines more efficiently.



Users

oVirt

CloudBalancing benchmark	Average	Min/Max	# datasets	Biggest dataset
Cloud hosting cost	-18%	-16% -21%	5	1600 computers 4800 processes
OptaPlanner versus traditional algorithm with domain knowledge		5 mins Simulated Annealing vs First Fit Decreasing		

MachineReassignment benchmark	Average	Min/Max	# datasets	Biggest dataset
Hardware congestion	-63%	-25% -97%	20	50k machines 5k processes
OptaPlanner versus arbitrary feasible assignments		5 mins Tabu Search vs First Feasible Fit		

Don't believe us? Run our open benchmarks yourself: http://www.optaplanner.org/code/benchmarks.html

2.4 WORKSHOP SEARCH REASONING

KIE OptaPlanner Deep Dive – Cloud Computer Balancing

The following hard constraints must be fulfilled:

- Every computer must be able to handle the minimum hardware requirements of the sum of its processes:
  - **CPU capacity**: The CPU power of a computer must be at least the sum of the CPU power required by the processes assigned to that computer.
  - **Memory capacity**: The RAM memory of a computer must be at least the sum of the RAM memory required by the processes assigned to that computer.
  - **Network capacity**: The network bandwidth of a computer must be at least the sum of the network bandwidth required by the processes assigned to that computer.

The following soft constraints should be optimized:

- Each computer that has one or more processes assigned, incurs a maintenance cost (which is fixed per computer).
  - **Cost**: Minimize the total maintenance cost.

KIE OptaPlanner Deep Dive – Cloud Computer Balancing

- **Domain Modelling / Constraint Satisfaction**

2.4 WORKSHOP SEARCH REASONING

KIE OptaPlanner Deep Dive – Cloud Computer Balancing

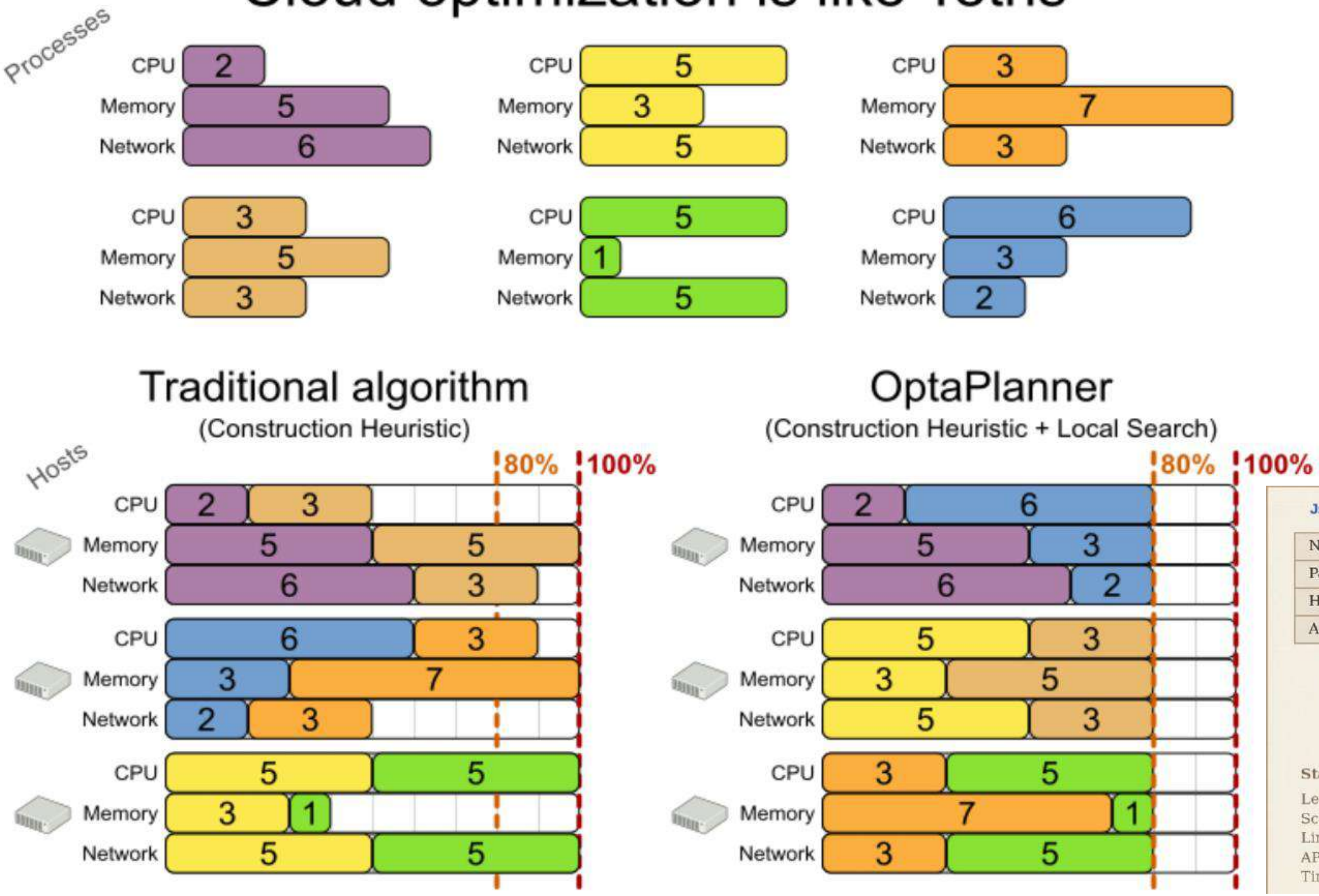
- **Domain Modelling / Constraint Satisfaction**
  - To create a domain model, define all the objects that represent the input data for the problem. In this simple example, the objects are **processes** and **computers**.
  - A separate object (Solution Class) in the domain model must represent a full data set of problem, which contains the input data as well as a solution. In this example, this object holds a list of **computers** and a list of **processes**. Each process is assigned to a computer; the **distribution of processes between computers** is the **solution**.

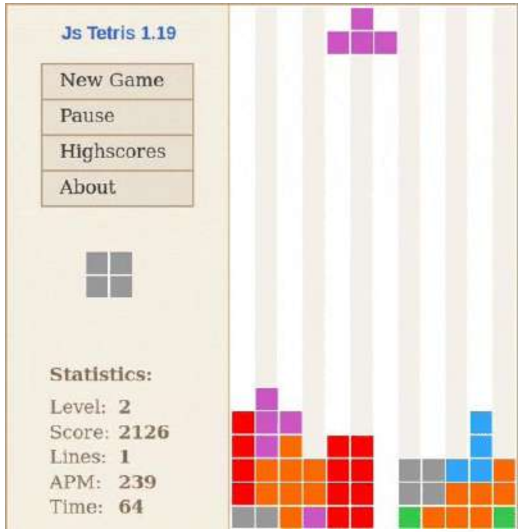
2.4 WORKSHOP SEARCH REASONING

KIE OptaPlanner Deep Dive – Cloud Computer Balancing

1. **Draw a class diagram of your domain model.**
2. **Normalize it to remove duplicate data.**
3. **Write down some sample instances for each class.**
   - **Computer**: represents a computer with certain hardware and maintenance costs.
   - **Process**: represents a process with a demand. Needs to be assigned to a Computer by Planner.
   - **CloudBalance**: represents a solution. Contains every Computer and Process for a certain data set.
   - For an object representing the full data set and solution, a sample instance holding the **score** must be present. Planner can calculate and compare the scores for different solutions; the solution with the highest score is the optimal solution. Therefore, the sample instance for CloudBalance is score.
4. **Determine which relationships (or fields) change during planning.**
   - **Planning entity**: The class (or classes) that Planner can change during solving. In this example, it is the class Process, because Planner can assign processes to computers.
   - **Problem fact**: A class representing input data that Planner can not change.
   - Planning variable: The property (or properties) of a planning entity class that changes during solving. In this example, it is the property computer on the class Process.
   - **Planning solution**: The class that represents a solution to the problem. This class must represent the full data set and contain all planning entities. In this example that is the class CloudBalance.

Cloud optimization is like Tetris





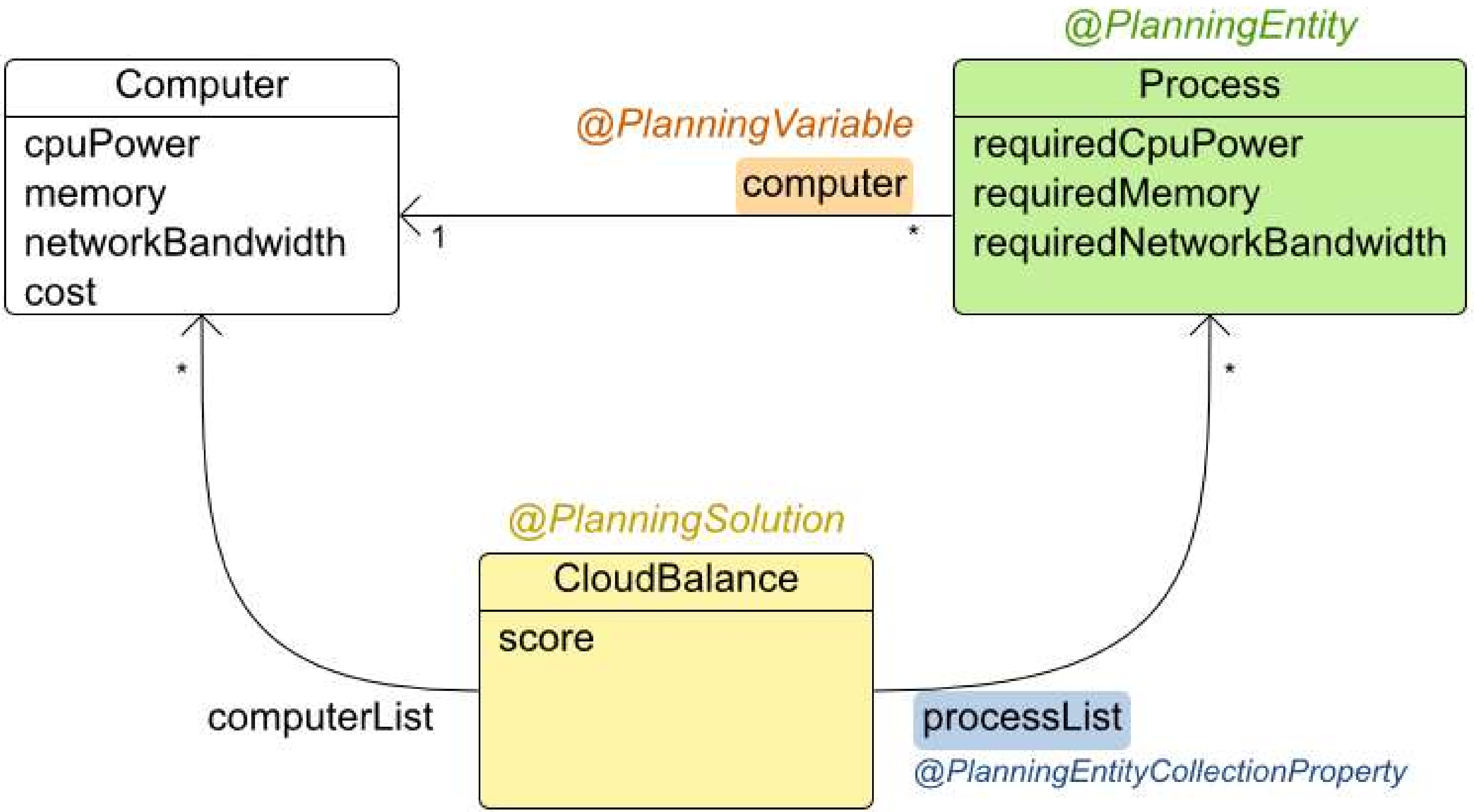
Terminology

- Problem fact
- Planning entity class: Process
- Planning entity: Process-7
- Planning variable: Process.getComputer()
- Planning value: Computer-3
- Planning value range: from Computer-0 to Computer-8

What changes during planning?

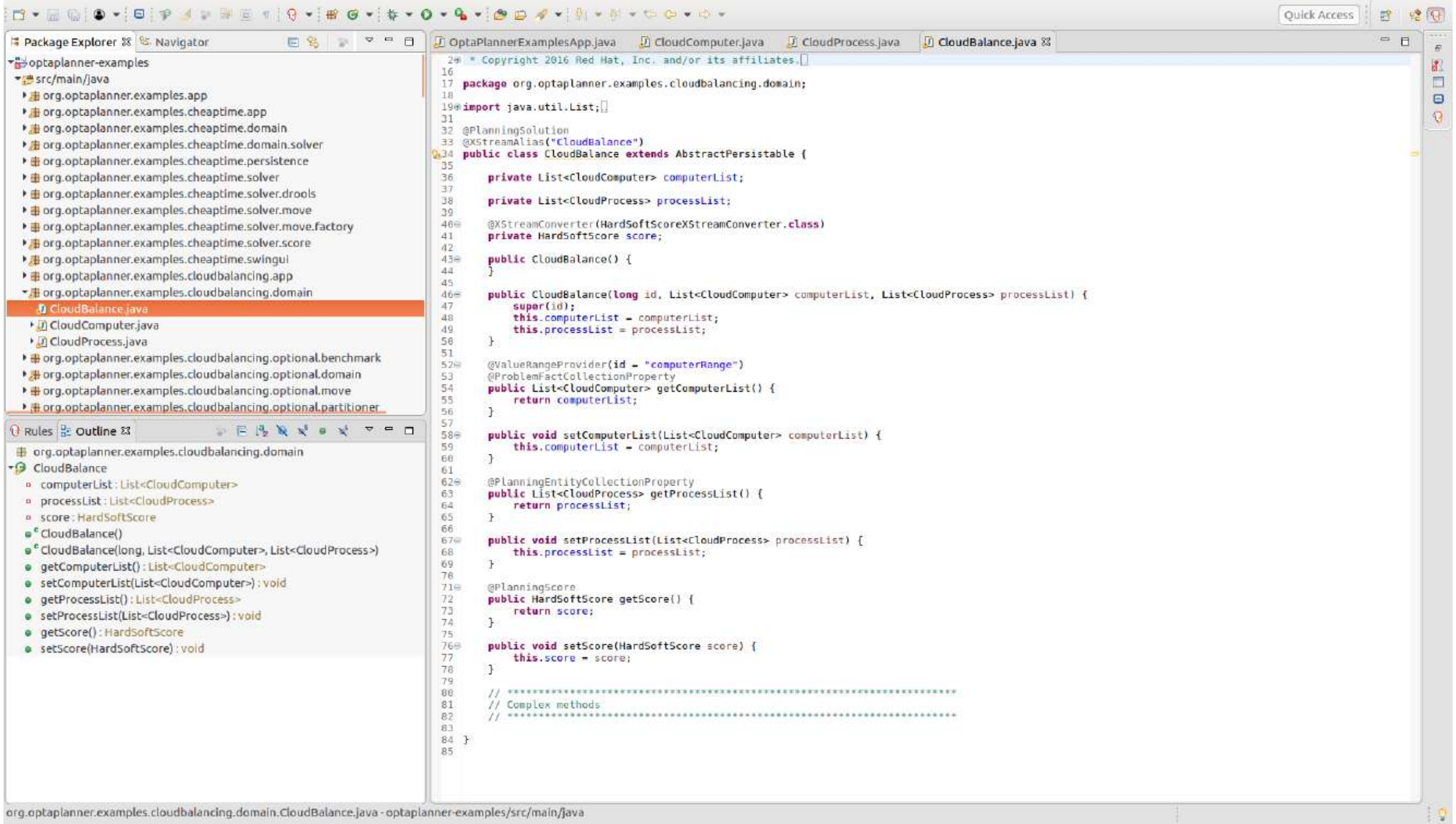
- Which class is a planning entity?
- Which property is a planning variable?

Cloud balance class diagram



KIE OptaPlanner Deep Dive – Cloud Computer Balancing

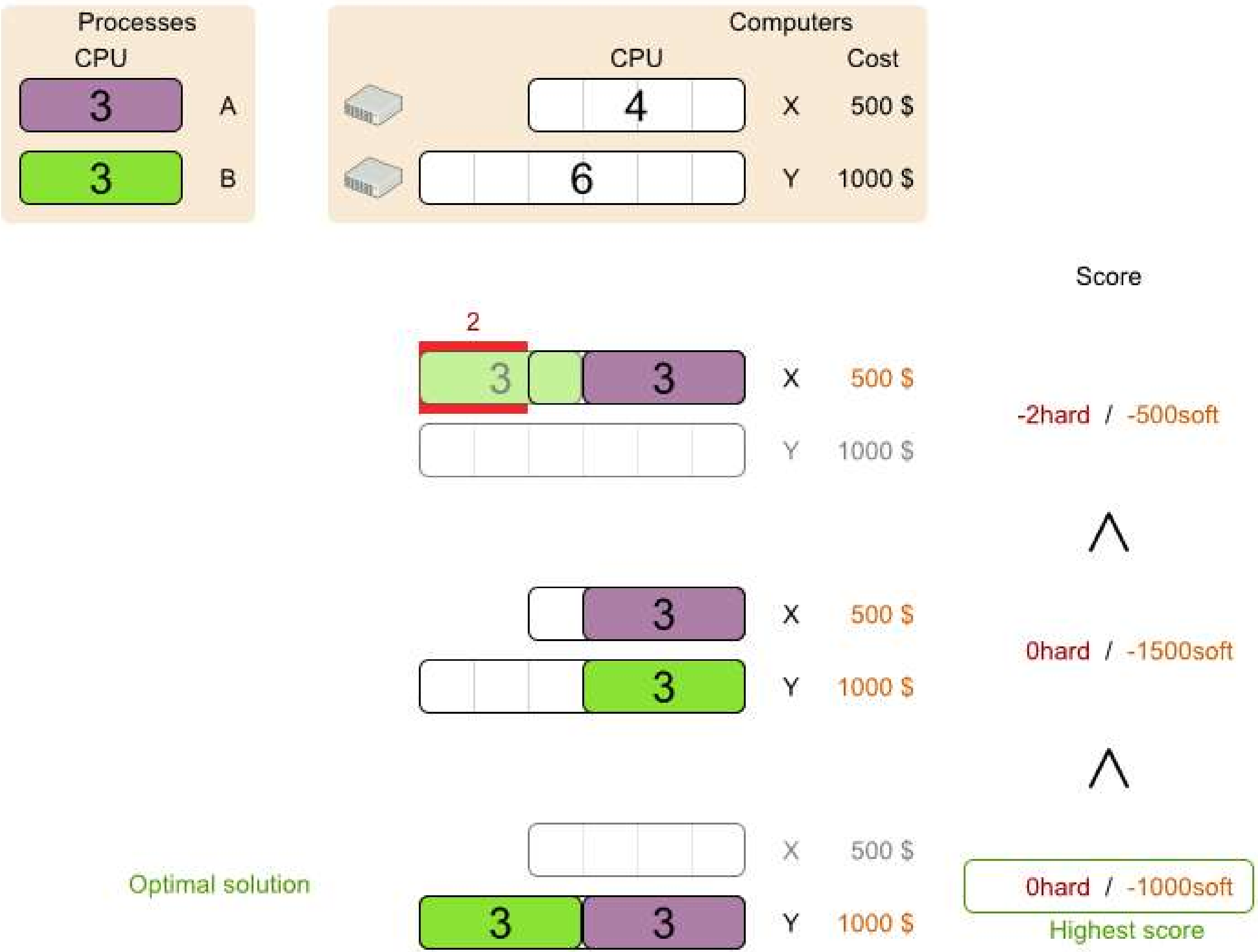
• Demo: Class Diagram Objects



2.4 WORKSHOP SEARCH REASONING

KIE OptaPlanner Deep Dive – Cloud Computer Balancing

- **Score Design**
  - To handle constraints
  - To compare solutions


Processes
CPU
3
A
3
B
Computers
CPU
Cost
4
X
500 $
6
Y
1000 $
Score
2
3
3
X
500 $
Y
1000 $
-2hard / -500soft
3
X
500 $
3
Y
1000 $
0hard / -1500soft
X
500 $
Optimal solution
3
3
Y
1000 $
0hard / -1000soft
Highest score

2.5.1. Easy Java Score Configuration

One way to define a score function is to implement the interface `EasyScoreCalculator` in plain Java.

```
<scoreDirectorFactory>
  <easyScoreCalculatorClass>org.optaplanner.examples.cloudbalancing.optional.score.CloudBalancingl
</scoreDirectorFactory>
```

Just implement the `calculateScore(Solution)` method to return a `HardSoftScore` instance.

Example 6. CloudBalancingEasyScoreCalculator.java

```
public class CloudBalancingEasyScoreCalculator implements EasyScoreCalculator<CloudBalance> {

    /**
     * A very simple implementation. The double loop can easily be removed by using Maps as sho
     * {@link CloudBalancingMapBasedEasyScoreCalculator#calculateScore(CloudBalance)}.
     */
    public HardSoftScore calculateScore(CloudBalance cloudBalance) {
        int hardScore = 0;
        int softScore = 0;
        for (CloudComputer computer : cloudBalance.getComputerList()) {
            int cpuPowerUsage = 0;
            int memoryUsage = 0;
            int networkBandwidthUsage = 0;
            boolean used = false;

            // Calculate usage
            for (CloudProcess process : cloudBalance.getProcessList()) {
                if (computer.equals(process.getComputer())) {
                    cpuPowerUsage += process.getRequiredCpuPower();
                    memoryUsage += process.getRequiredMemory();
```

2.5.2. Drools Score Configuration

Drools score calculation uses incremental calculation, where every score constraint is written as one or more score rules.

Using the Drools rule engine for score calculation, allows you to integrate with other Drools technologies, such as decision tables (XLS or web based), the KIE Workbench, ...

Prerequisite To use the Drools rule engine as a score function, simply add a `scoreDrl` resource in the classpath:

```
<scoreDirectorFactory>
  <scoreDrl>org/optaplanner/examples/cloudbalancing/solver/cloudBalancingScoreRules.drl</scoreDrl:
</scoreDirectorFactory>
```

1. We want to make sure that all computers have enough CPU, RAM and network bandwidth to support all their processes, so we make these hard constraints:

Example 7. cloudBalancingScoreRules.drl - Hard Constraints

```
rule "requiredCpuPowerTotal"
    when
        $computer : CloudComputer($cpuPower : cpuPower)
        accumulate(
            CloudProcess(
                computer == $computer,
                $requiredCpuPower : requiredCpuPower);
            $requiredCpuPowerTotal : sum($requiredCpuPower);
            $requiredCpuPowerTotal > $cpuPower
        )
    then
        scoreHolder.addHardConstraintMatch(kcontext, $cpuPower - $requiredCpuPowerTotal);
end
```

Drools score calculation

- Constraints in Drools Rule Language (DRL)
  - Declarative (like SQL, regular expression)
- Integration opportunities
  - Drools Workbench
  - Decision tables

Drools score calculation: facts

- Facts in DRL loaded from
  - `@ProblemFact(Collection)Property`
  - `@PlanningEntity(Collection)Property`

DRL hard constraint: CPU power

```
rule "requiredCpuPowerTotal"
  when
    // there is a computer
    $s : Computer($cpu : cpuPower)
    // with too little cpu for its processes
    accumulate(
      Process(computer == $s, $requiredCpu : requiredCpuPower);
      $total : sum($requiredCpu);
      $total > $cpu
    )
  then
    // lower hard score by the excessive CPU usage
    scoreHolder.addHardConstraintMatch(kcontext,
        $cpu - $total);
end
```

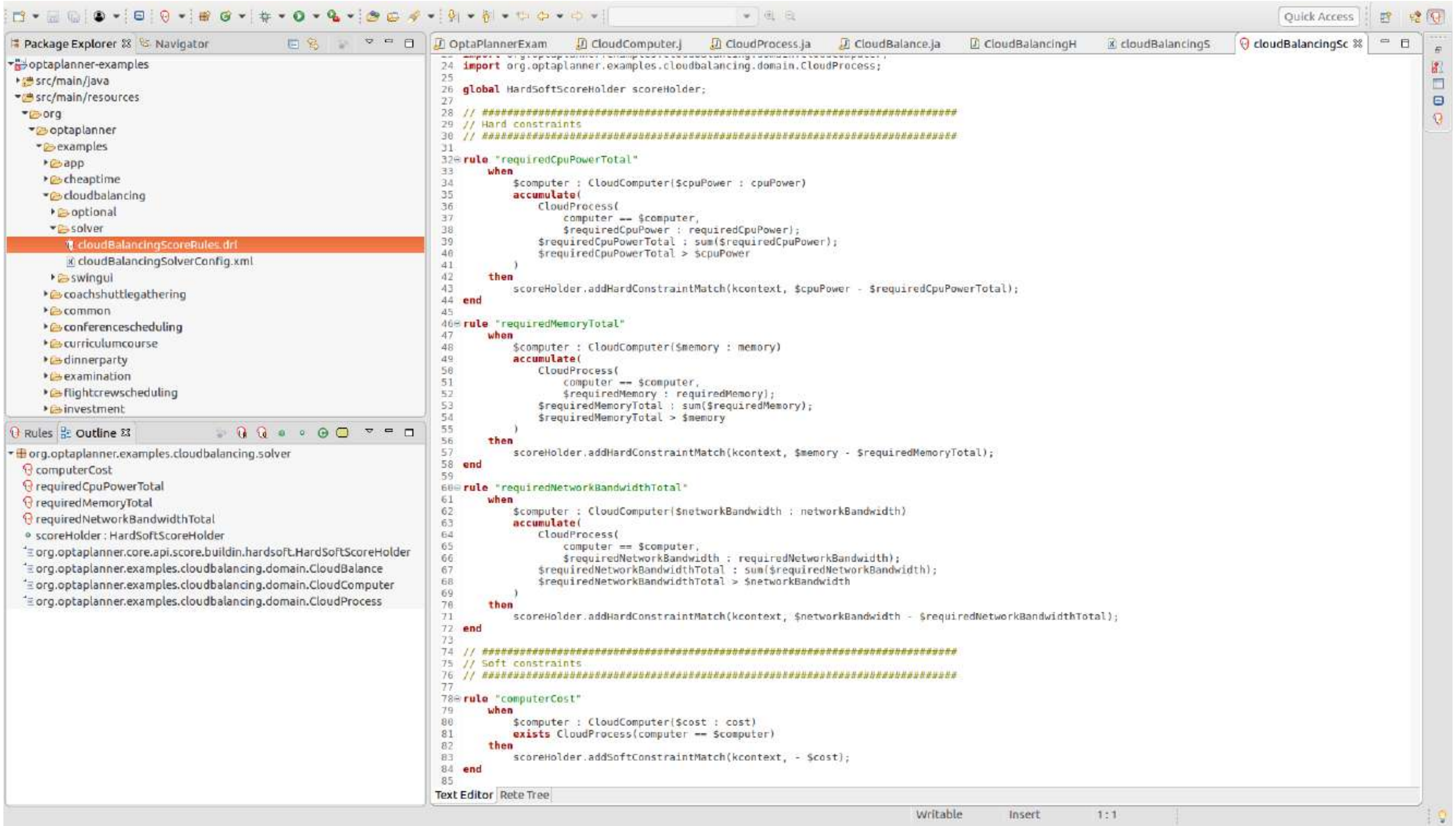
DRL soft constraint: computer cost

```
rule "computerCost"
  when
    // there is a computer
    $s : Computer($c : cost)
    // there is a processes on that computer
    exists Process(computer == $s)
  then
    // lower soft score by the maintenance cost
    scoreHolder.addSoftConstraintMatch(kcontext, - $c);
end
```

2.4 WORKSHOP SEARCH REASONING

KIE OptaPlanner Deep Dive – Cloud Computer Balancing

- **Demo: Score Design**



2.4 WORKSHOP SEARCH REASONING

KIE OptaPlanner Deep Dive – Cloud Computer Balancing

- **Solver in Java / Eclipse**

2.4 WORKSHOP SEARCH REASONING

KIE OptaPlanner Deep Dive – Cloud Computer Balancing

Example 4. CloudBalancingHelloWorld.java

```
public class CloudBalancingHelloWorld {

    public static void main(String[] args) {
        // Build the Solver
        SolverFactory<CloudBalance> solverFactory = SolverFactory.createFromXmlResource(
                "org/optaplanner/examples/cloudbalancing/solver/cloudBalancingSolverConfig.xml"
        Solver<CloudBalance> solver = solverFactory.buildSolver();

        // Load a problem with 400 computers and 1200 processes
        CloudBalance unsolvedCloudBalance = new CloudBalancingGenerator().createCloudBalance(40

        // Solve the problem
        CloudBalance solvedCloudBalance = solver.solve(unsolvedCloudBalance);

        // Display the result
        System.out.println("\nSolved cloudBalance with 400 computers and 1200 processes:\n"
                + toDisplayString(solvedCloudBalance));
    }

    ...
}
```

2.4 WORKSHOP SEARCH REASONING

KIE OptaPlanner Deep Dive – Cloud Computer Balancing

The solver configuration file determines how the solving process works; it is considered a part of the code. The file is named `cloudBalancingSolverConfig.xml`.

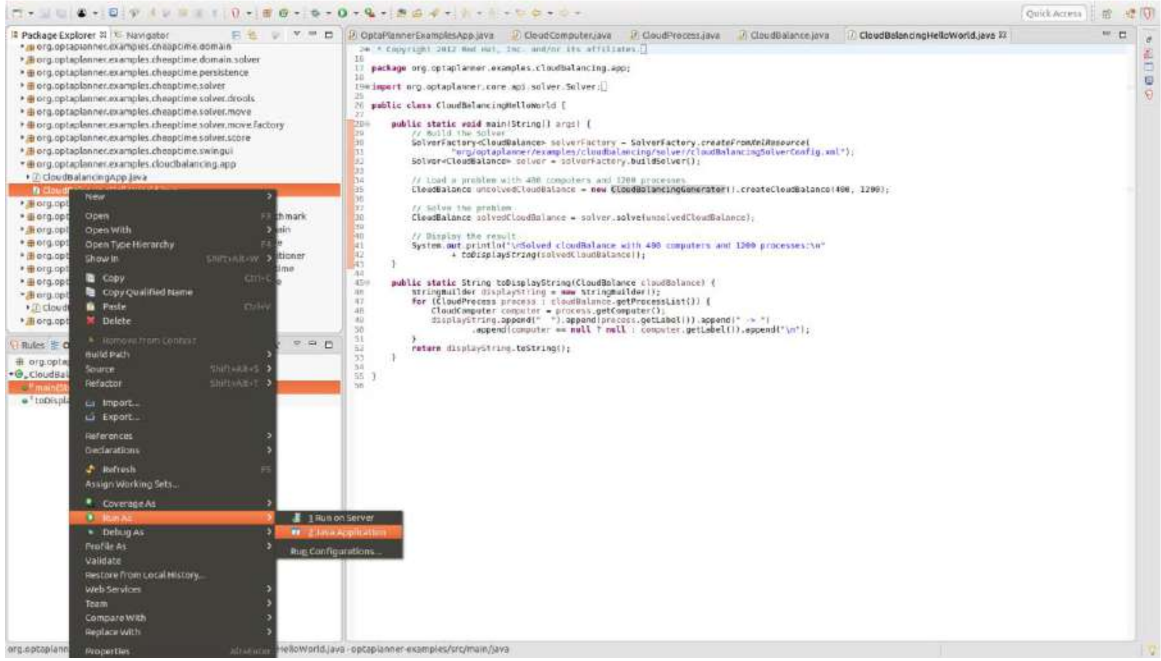
Example 5. cloudBalancingSolverConfig.xml

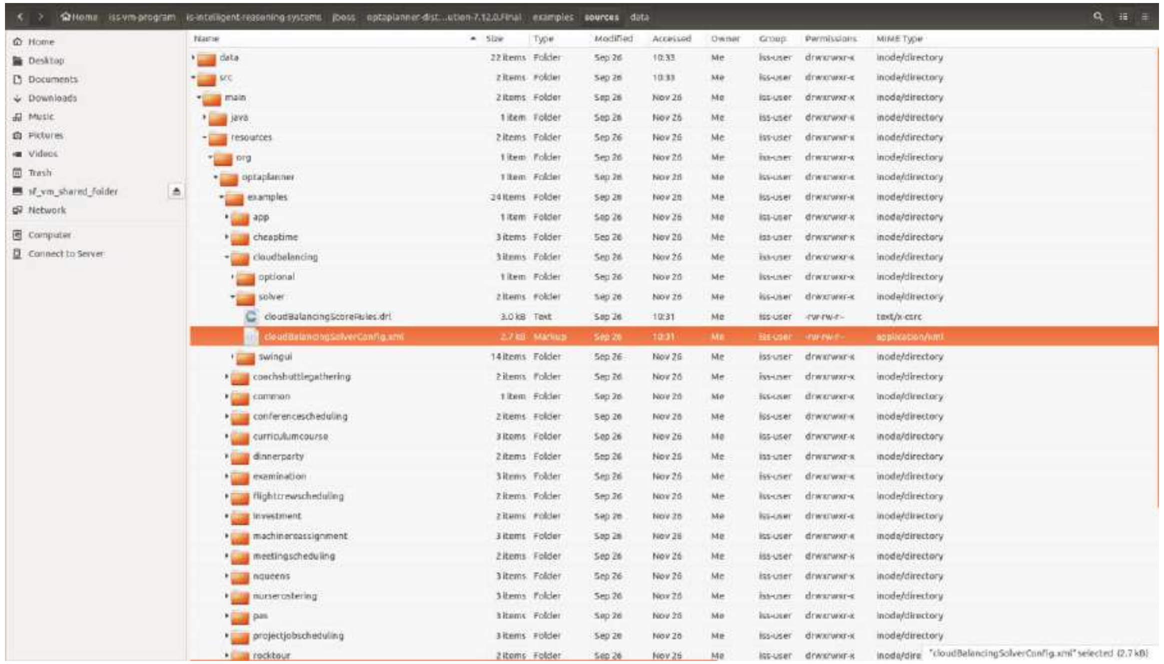
```
<?xml version="1.0" encoding="UTF-8"?>
<solver>
  <!-- Domain model configuration -->
  <scanAnnotatedClasses/>

  <!-- Score configuration -->
  <scoreDirectorFactory>
    <easyScoreCalculatorClass>org.optaplanner.examples.cloudbalancing.optional.score.CloudBalan
    <!--<scoreDrl>org/optaplanner/examples/cloudbalancing/solver/cloudBalancingScoreRules.drl</
  </scoreDirectorFactory>

  <!-- Optimization algorithms configuration -->
  <termination>
    <secondsSpentLimit>30</secondsSpentLimit>
  </termination>
</solver>
```

2.4 WORKSHOP SEARCH REASONING

KIE OptaPlanner Deep Dive – Cloud Computer Balancing

- Demo: Solver in Java / Eclipse





2.4 WORKSHOP SEARCH REASONING

KIE OptaPlanner Deep Dive – Cloud Computer Balancing

- **Solver in KIE Workbench & Server**

2.4 WORKSHOP SEARCH REASONING

KIE OptaPlanner Deep Dive – Cloud Computer Balancing

- **Solver in KIE Workbench & Server**

OptaPlanner Workbench and Execution Server User Guide

OptaPlanner

1. OptaPlanner Engine

See the OptaPlanner User Guide.

Link https://docs.jboss.org/optaplanner/release/latestFinal/optaplanner-docs/html_single/

OptaPlanner Workbench

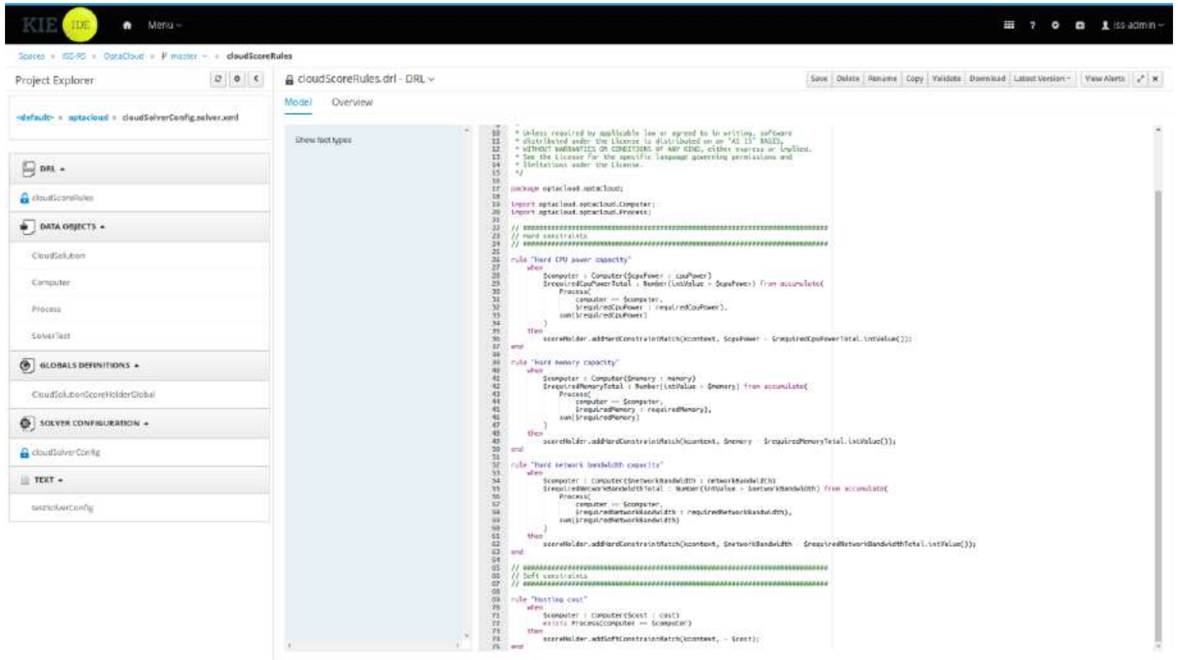
Link https://docs.jboss.org/optaplanner/release/latestFinal/optaplanner-wb-es-docs/html_single/

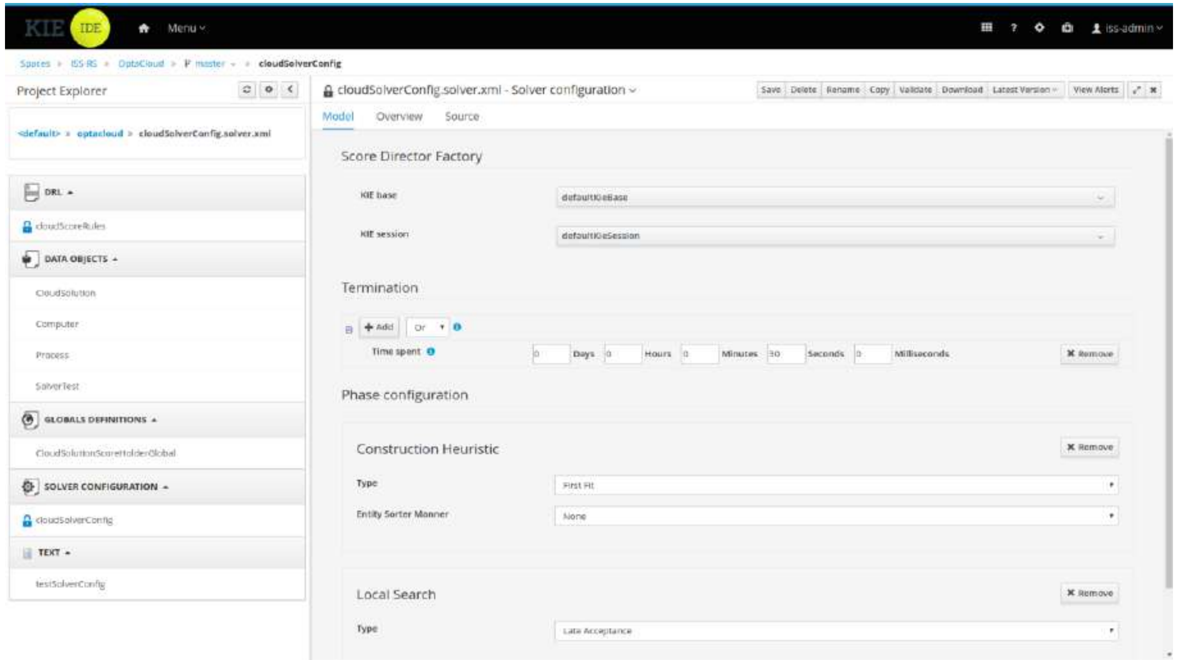
- User type: application user
- Username: planner
- Password: Planner123_
- Groups: kie-server,admin

2.4 WORKSHOP SEARCH REASONING

KIE OptaPlanner Deep Dive – Cloud Computer Balancing

- Demo: Solver in KIE Workbench & Server

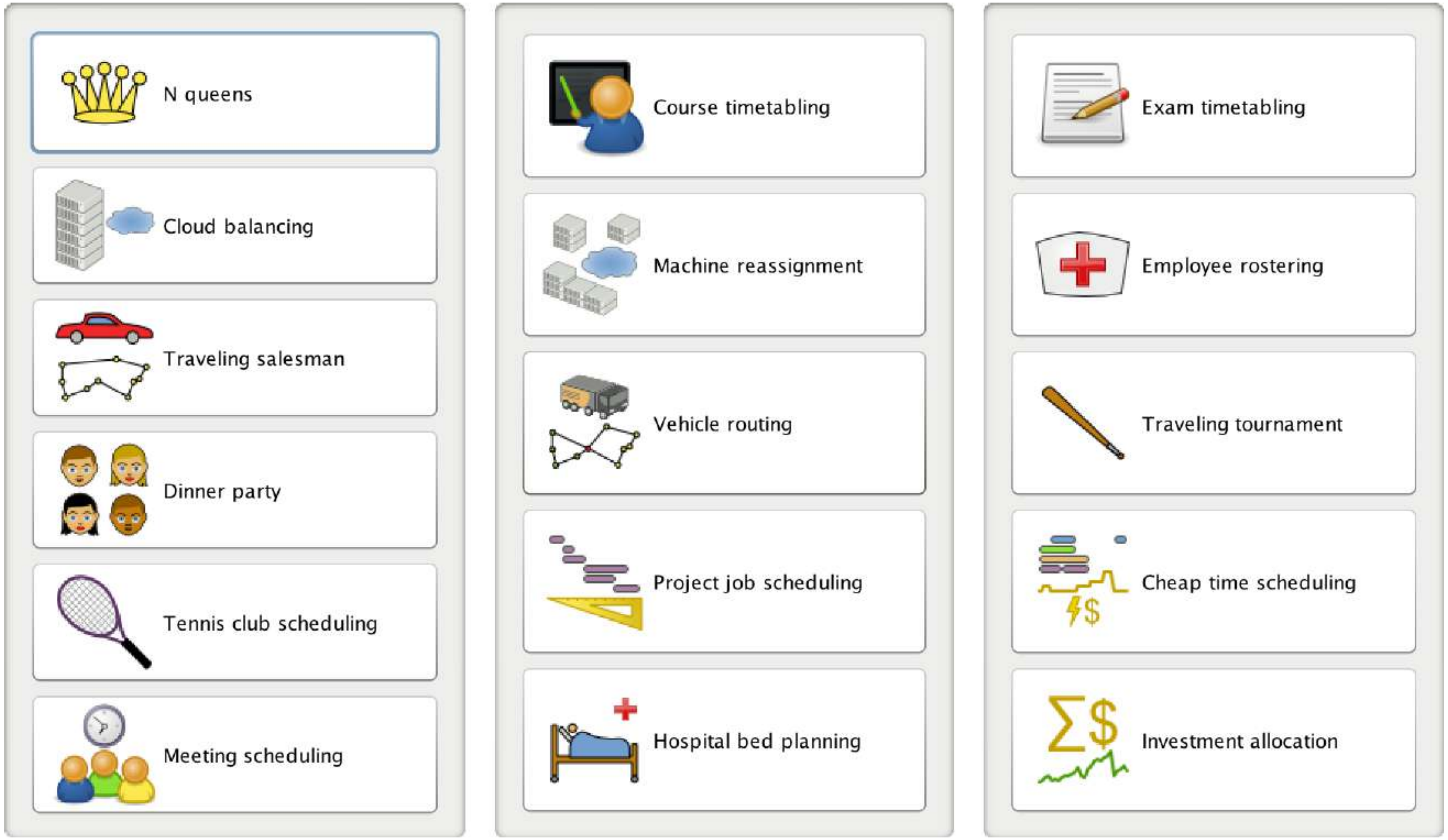




2.4 WORKSHOP SEARCH REASONING

KIE OptaPlanner Development – Individual Work

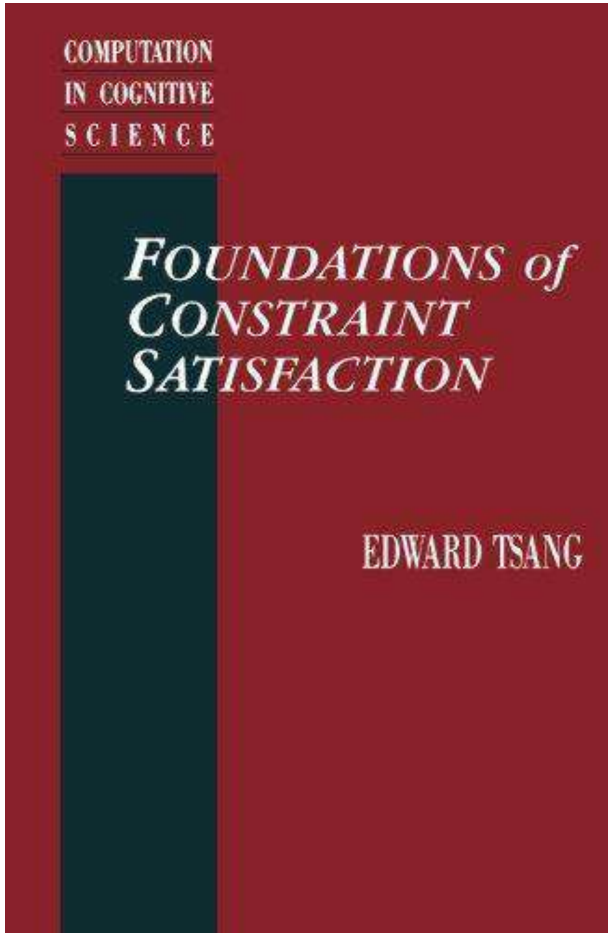
- Choose relevant use cases; Analyse and adapt.



Link http://www.optaplanner.org/learn/slides/optaplanner-presentation/index.html#/1

Link http://www.optaplanner.org/learn/slides/optaplanner-presentation/training.html#/1

DAY 2 REFERENCE



1. **OptaPlanner Deep Dive Training**

   http://www.optaplanner.org/learn/slides/optaplanner-presentation/training.html#/1

2. OptaPlanner User Guide: Workbench & Execution Server

   https://docs.jboss.org/optaplanner/release/latestFinal/optaplanner-wb-es-docs/html_single/

3. OptaPlanner User Guide: Engine

   https://docs.jboss.org/optaplanner/release/latestFinal/optaplanner-docs/html_single/

4. Chapter 2. Getting started with solvers in Workbench: An employee rostering example

   https://access.redhat.com/documentation/en-us/red_hat_process_automation_manager/7.2/html-single/getting_started_with_red_hat_business_optimizer/

5. Chapter 3. Getting started with Java solvers: A cloud balancing example

   https://access.redhat.com/documentation/en-us/red_hat_process_automation_manager/7.2/html-single/getting_started_with_red_hat_business_optimizer/

DAY 2 SUMMARY

2.1 Informed Search Techniques (2/2)

- Tabu Search (TS)
- Simulated Annealing (SA)
- Difference Between Informed & Uninformed Search

2.2 Search Based Intelligent Systems

- Search Engines
- Logic & Information Retrieval Chat-bot
- Calendar Planner
- Vehicle Routing Optimizer

2.4 Search Reasoning Workshop

END OF LECTURE NOTES

APPENDICES

KIE OptaPlanner Development example – Curriculum Course Scheduling

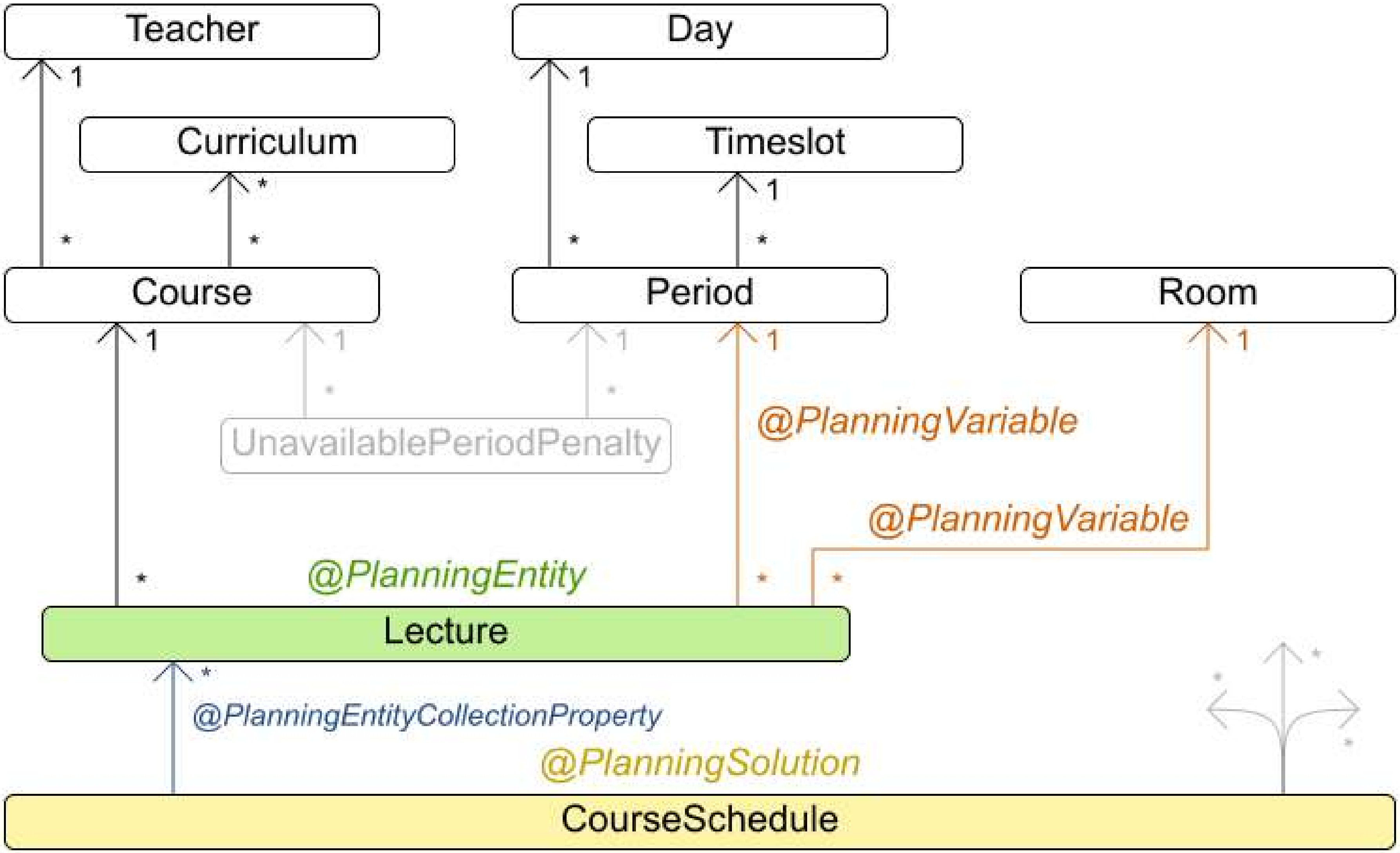
2.4 WORKSHOP SEARCH REASONING

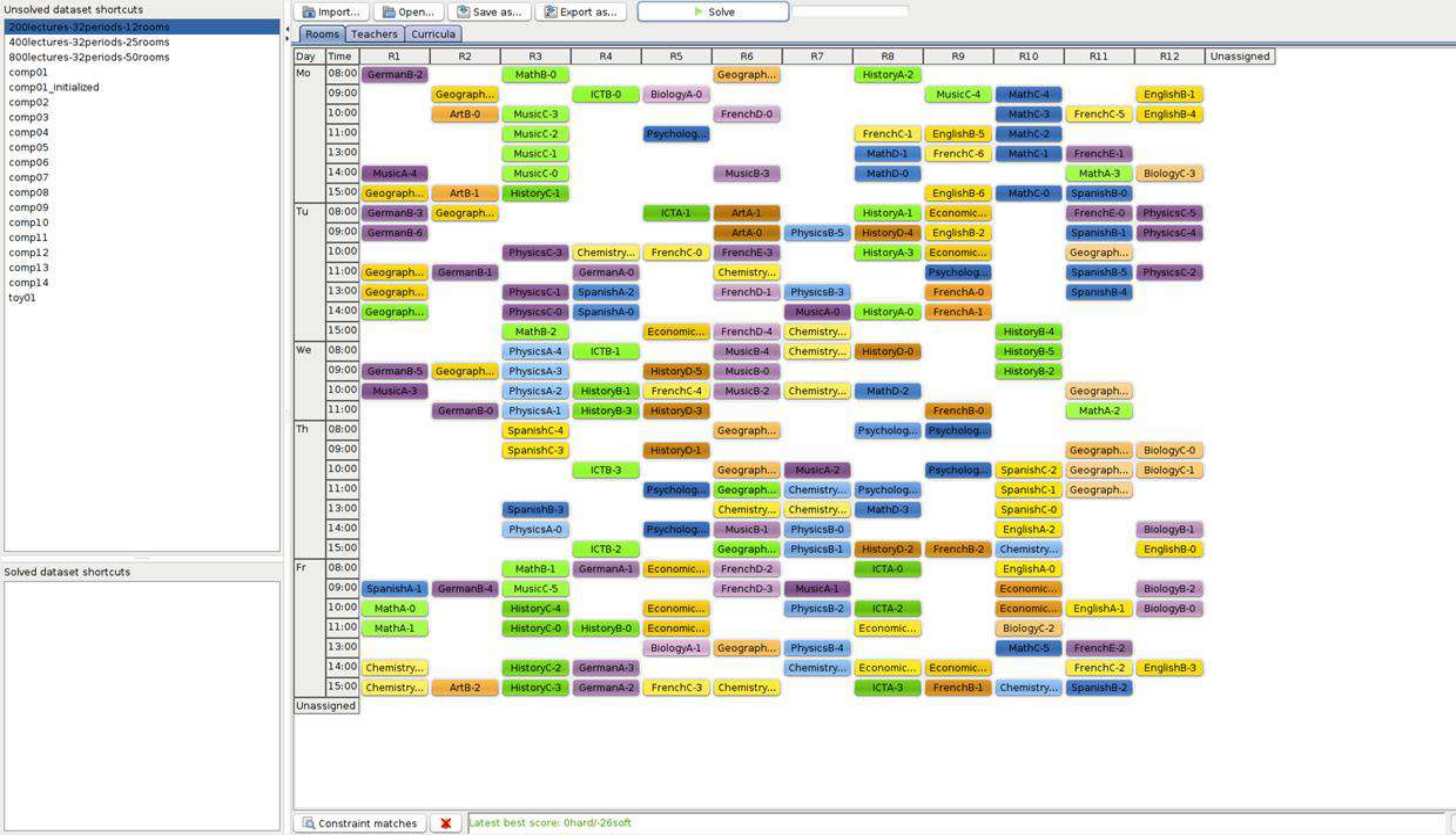
KIE OptaPlanner Development example – Curriculum Course Scheduling

- **Business Scenario / Problem Description**
- **Schedule each lecture into a timeslot and into a room.**
- **Hard constraints:**
  - Teacher conflict: A teacher must not have two lectures in the same period.
  - Curriculum conflict: A curriculum must not have two lectures in the same period.
  - Room occupancy: two lectures must not be in the same room in the same period.
  - Unavailable period (specified per dataset): A specific lecture must not be assigned to a specific period.
- **Soft constraints:**
  - Room capacity: A room's capacity should not be less than the number of students in its lecture.
  - Minimum working days: Lectures of the same course should be spread out into a minimum number of days.
  - Curriculum compactness: Lectures belonging to the same curriculum should be adjacent to each other (so in consecutive periods).
  - Room stability: Lectures of the same course should be assigned to the same room.
- **The problem is defined by [the International Timetabling Competition 2007 track 3](http://www.cs.qub.ac.uk/itc2007/curriculmcourse/course_curriculm_index.htm).**

  http://www.cs.qub.ac.uk/itc2007/curriculmcourse/course_curriculm_index.htm

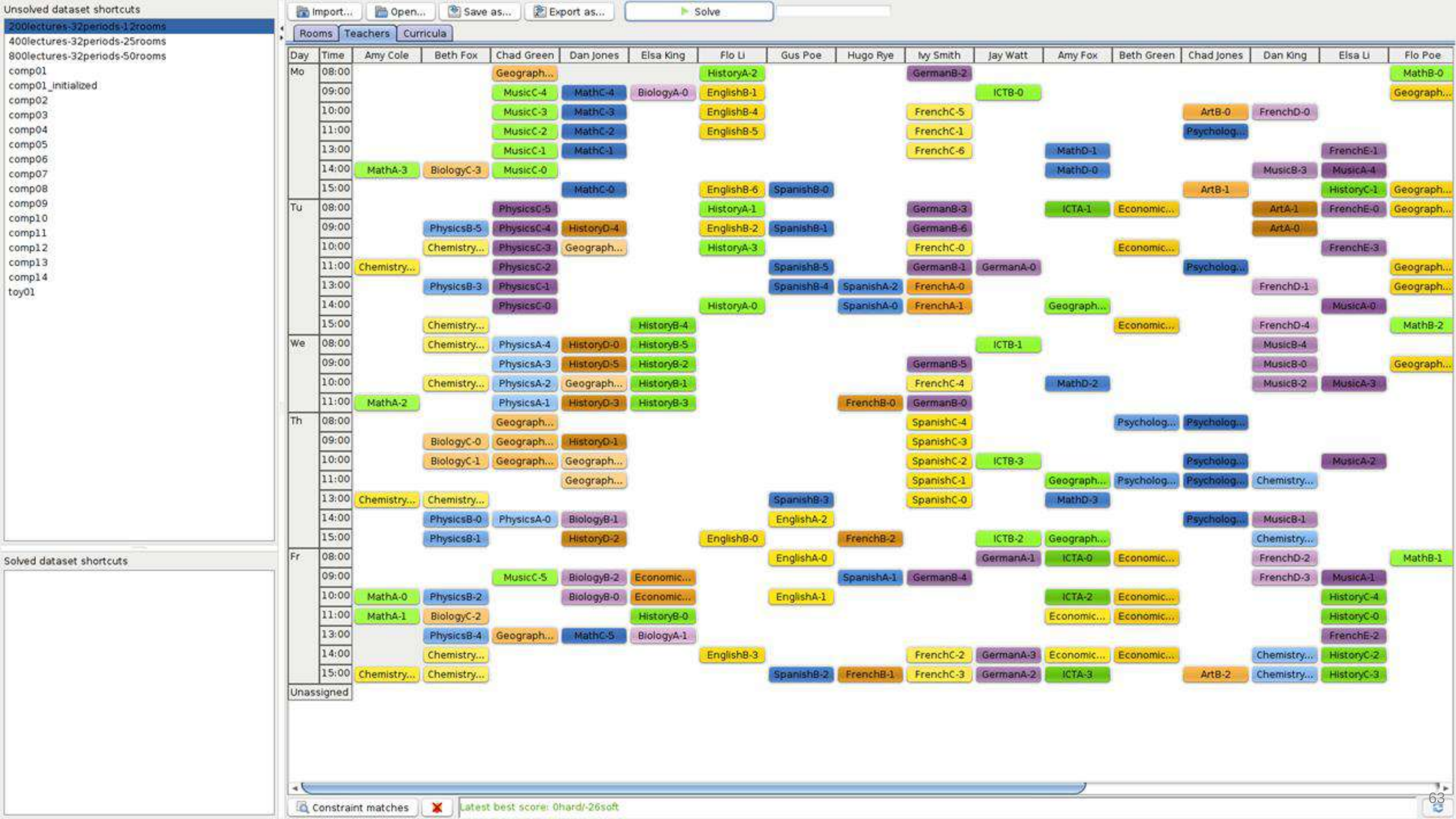
Curriculum course class diagram




Unsolved dataset shortcuts
200lectures-32periods-12rooms
400lectures-32periods-25rooms
800lectures-32periods-50rooms
comp01
comp01_initialized
comp02
comp03
comp04
comp05
comp06
comp07
comp08
comp09
comp10
comp11
comp12
comp13
comp14
toy01
Solved dataset shortcuts
Import...
Open...
Save as...
Export as...
Solve
Rooms
Teachers
Curricula

Day	Time	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	R11	R12	Unassigned
Mo	08:00	GermanB-2		MathB-0			Geograph...		HistoryA-2					
	09:00		Geograph...		ICTB-0	BiologyA-0				MusicC-4	MathC-4		EnglishB-1	
	10:00		ArtB-0	MusicC-3			FrenchD-0				MathC-3	FrenchC-5	EnglishB-4	
	11:00			MusicC-2		Psycholog...			FrenchC-1	EnglishB-5	MathC-2			
	13:00			MusicC-1					MathD-1	FrenchC-6	MathC-1	FrenchE-1		
	14:00	MusicA-4		MusicC-0			MusicB-3		MathD-0			MathA-3	BiologyC-3	
	15:00	Geograph...	ArtB-1	HistoryC-1						EnglishB-6	MathC-0	SpanishB-0		
Tu	08:00	GermanB-3	Geograph...			ICTA-1	ArtA-1		HistoryA-1	Economic...		FrenchE-0	PhysicsC-5	
	09:00	GermanB-6					ArtA-0	PhysicsB-5	HistoryD-4	EnglishB-2		SpanishB-1	PhysicsC-4	
	10:00			PhysicsC-3	Chemistry...	FrenchC-0	FrenchE-3		HistoryA-3	Economic...		Geograph...		
	11:00	Geograph...	GermanB-1		GermanA-0		Chemistry...			Psycholog...		SpanishB-5	PhysicsC-2	
	13:00	Geograph...		PhysicsC-1	SpanishA-2		FrenchD-1	PhysicsB-3		FrenchA-0		SpanishB-4		
	14:00	Geograph...		PhysicsC-0	SpanishA-0			MusicA-0	HistoryA-0	FrenchA-1				
	15:00			MathB-2		Economic...	FrenchD-4	Chemistry...			HistoryB-4			
We	08:00			PhysicsA-4	ICTB-1		MusicB-4	Chemistry...	HistoryD-0		HistoryB-5			
	09:00	GermanB-5	Geograph...	PhysicsA-3		HistoryD-5	MusicB-0				HistoryB-2			
	10:00	MusicA-3		PhysicsA-2	HistoryB-1	FrenchC-4	MusicB-2	Chemistry...	MathD-2			Geograph...		
	11:00		GermanB-0	PhysicsA-1	HistoryB-3	HistoryD-3				FrenchB-0		MathA-2		
Th	08:00			SpanishC-4			Geograph...		Psycholog...	Psycholog...				
	09:00			SpanishC-3		HistoryD-1						Geograph...	BiologyC-0	
	10:00				ICTB-3		Geograph...	MusicA-2		Psycholog...	SpanishC-2	Geograph...	BiologyC-1	
	11:00					Psycholog...	Geograph...	Chemistry...	Psycholog...		SpanishC-1	Geograph...		
	13:00			SpanishB-3			Chemistry...	Chemistry...	MathD-3		SpanishC-0			
	14:00			PhysicsA-0		Psycholog...	MusicB-1	PhysicsB-0			EnglishA-2		BiologyB-1	
	15:00				ICTB-2		Geograph...	PhysicsB-1	HistoryD-2	FrenchB-2	Chemistry...		EnglishB-0	
Fr	08:00			MathB-1	GermanA-1	Economic...	FrenchD-2		ICTA-0		EnglishA-0			
	09:00	SpanishA-1	GermanB-4	MusicC-5			FrenchD-3	MusicA-1			Economic...		BiologyB-2	
	10:00	MathA-0		HistoryC-4		Economic...		PhysicsB-2	ICTA-2		Economic...	EnglishA-1	BiologyB-0	
	11:00	MathA-1		HistoryC-0	HistoryB-0	Economic...			Economic...		BiologyC-2			
	13:00					BiologyA-1	Geograph...	PhysicsB-4			MathC-5	FrenchE-2		
	14:00	Chemistry...		HistoryC-2	GermanA-3			Chemistry...	Economic...	Economic...		FrenchC-2	EnglishB-3	
	15:00	Chemistry...	ArtB-2	HistoryC-3	GermanA-2	FrenchC-3	Chemistry...		ICTA-3	FrenchB-1	Chemistry...	SpanishB-2		
Unassigned														

Constraint matches
Latest best score: 0hard/-26soft


Unsolved dataset shortcuts
200lectures-32periods-12rooms
400lectures-32periods-25rooms
800lectures-32periods-50rooms
comp01
comp01_initialized
comp02
comp03
comp04
comp05
comp06
comp07
comp08
comp09
comp10
comp11
comp12
comp13
comp14
toy01
Solved dataset shortcuts
Import...
Open...
Save as...
Export as...
Solve
Rooms
Teachers
Curricula

Day	Time	Amy Cole	Beth Fox	Chad Green	Dan Jones	Elsa King	Flo Li	Gus Poe	Hugo Rye	Ivy Smith	Jay Watt	Amy Fox	Beth Green	Chad Jones	Dan King	Elsa Li	Flo Poe
Mo	08:00			Geograph...			HistoryA-2			GermanB-2							MathB-0
	09:00			MusicC-4	MathC-4	BiologyA-0	EnglishB-1				ICTB-0						Geograph...
	10:00			MusicC-3	MathC-3		EnglishB-4			FrenchC-5				ArtB-0	FrenchD-0		
	11:00			MusicC-2	MathC-2		EnglishB-5			FrenchC-1				Psycholog...			
	13:00			MusicC-1	MathC-1					FrenchC-6		MathD-1				FrenchE-1	
	14:00	MathA-3	BiologyC-3	MusicC-0								MathD-0			MusicB-3	MusicA-4	
	15:00				MathC-0		EnglishB-6	SpanishB-0						ArtB-1		HistoryC-1	Geograph...
Tu	08:00			PhysicsC-5			HistoryA-1			GermanB-3		ICTA-1	Economic...		ArtA-1	FrenchE-0	Geograph...
	09:00		PhysicsB-5	PhysicsC-4	HistoryD-4		EnglishB-2	SpanishB-1		GermanB-6					ArtA-0		
	10:00		Chemistry...	PhysicsC-3	Geograph...		HistoryA-3			FrenchC-0			Economic...			FrenchE-3	
	11:00	Chemistry...		PhysicsC-2				SpanishB-5		GermanB-1	GermanA-0			Psycholog...			Geograph...
	13:00		PhysicsB-3	PhysicsC-1				SpanishB-4	SpanishA-2	FrenchA-0					FrenchD-1		Geograph...
	14:00			PhysicsC-0			HistoryA-0		SpanishA-0	FrenchA-1		Geograph...				MusicA-0	
	15:00		Chemistry...			HistoryB-4							Economic...		FrenchD-4		MathB-2
We	08:00		Chemistry...	PhysicsA-4	HistoryD-0	HistoryB-5					ICTB-1				MusicB-4		
	09:00			PhysicsA-3	HistoryD-5	HistoryB-2				GermanB-5					MusicB-0		Geograph...
	10:00		Chemistry...	PhysicsA-2	Geograph...	HistoryB-1				FrenchC-4		MathD-2			MusicB-2	MusicA-3	
	11:00	MathA-2		PhysicsA-1	HistoryD-3	HistoryB-3			FrenchB-0	GermanB-0							
Th	08:00			Geograph...						SpanishC-4			Psycholog...	Psycholog...			
	09:00		BiologyC-0	Geograph...	HistoryD-1					SpanishC-3							
	10:00		BiologyC-1	Geograph...	Geograph...					SpanishC-2	ICTB-3			Psycholog...		MusicA-2	
	11:00				Geograph...					SpanishC-1		Geograph...	Psycholog...	Psycholog...	Chemistry...		
	13:00	Chemistry...	Chemistry...					SpanishB-3		SpanishC-0		MathD-3					
	14:00		PhysicsB-0	PhysicsA-0	BiologyB-1			EnglishA-2						Psycholog...	MusicB-1		
	15:00		PhysicsB-1		HistoryD-2		EnglishB-0		FrenchB-2		ICTB-2	Geograph...			Chemistry...		
Fr	08:00							EnglishA-0			GermanA-1	ICTA-0	Economic...		FrenchD-2		MathB-1
	09:00			MusicC-5	BiologyB-2	Economic...			SpanishA-1	GermanB-4					FrenchD-3	MusicA-1	
	10:00	MathA-0	PhysicsB-2		BiologyB-0	Economic...		EnglishA-1				ICTA-2	Economic...			HistoryC-4	
	11:00	MathA-1	BiologyC-2			HistoryB-0						Economic...	Economic...			HistoryC-0	
	13:00		PhysicsB-4	Geograph...	MathC-5	BiologyA-1										FrenchE-2	
	14:00		Chemistry...				EnglishB-3			FrenchC-2	GermanA-3	Economic...	Economic...		Chemistry...	HistoryC-2	
	15:00	Chemistry...	Chemistry...					SpanishB-2	FrenchB-1	FrenchC-3	GermanA-2	ICTA-3		ArtB-2	Chemistry...	HistoryC-3	
Unassigned																	

Constraint matches
Latest best score: 0hard/-26soft

NICF- Reasoning Systems (SF)

Overview

Reference No	CRS-Q-0036478-ICT
Part of	-
Duration	5 days
Course Time	9:00am - 5:00pm
Enquiry	Please contact Ms. Jaymee TAN at tel: 65161206 or email isstfhj@nus.edu.sg for more details.

How can you capitalise on the use of Artificial Intelligence – Reasoning Systems to drive innovation and efficiency in your organisation? This 5-day course enables participants to understand the relevant knowledge needed to architect and/or lead teams executing intelligent system projects to reason, simulate, and optimise complex business problems. For example, how to predict future workload and staffing requirements by area, designation, skill and/or role; and how to dynamically redeploy workforce based on unplanned events (sick leave, workload and customer orders fluctuations) in real-time.

Through a mix of lectures reinforced by case examples, participants will acquire comprehensive knowledge of artificial intelligence (AI) techniques, including search, scheduling, optimisation, constraint satisfaction, evolutionary computation, and data mining. Participants will also get hands-on learning to integrate hybrid reasoning systems

This course is for business managers, data specialists, consultants, IT professionals and business professionals interested in learning how reasoning systems with AI optimization techniques can be applied into an organization to drive innovation, efficiency and identify competitive advantages.

This course is a part of the Artificial Intelligence and Graduate Certificate in Intelligent Reasoning Systems, which is a part of the Stackable Graduate Certificate Programme in Artificial Intelligent Systems (Masters Degree) offered by NUS-ISS.

Upcoming Classes

Class 1   12 Nov 2019 to 18 Nov 2019 (Full Time)

Duration: 5 days

When:

Nov: 12, 13, 14, 15, 18

Time:

9:00am - 5:00pm

NICF- Problem Solving using Pattern Recognition (SF)

Overview

Reference No	CRS-Q-0035256-ICT
Part of	-
Duration	5 days
Course Time	9:00am - 5:00pm
Enquiry	Please contact Ms. Jaymee TAN at tel: 65161206 or email isstfhj@nus.edu.sg for more details.

Pattern recognition is one of the most important areas of Artificial Intelligence. It is a branch of machine learning that focuses on the recognition of patterns and regularities in data. Pattern recognition systems can be trained from labelled training data through supervised learning and or unlabelled data through unsupervised learning.

Pattern recognition has been widely used to solve many real-world problems such as image processing, speech recognition, data mining, business analytics, etc. There are many pattern recognition techniques available to perform different tasks such as regression, classification, clustering, etc. using various statistical and machine learning algorithms.

This course will be useful for participants to acquire pattern recognition knowledge. It will help participants analyse data more effectively by deriving useful hidden patterns in the data. Participants will also learn how to select and apply the most suitable pattern recognition techniques to solve the given problems and develop pattern recognition systems.

This course is part of the Artificial Intelligence and Graduate Certificate in Pattern Recognition Systems Series offered by NUS-ISS

Upcoming Classes

Class 1   04 Nov 2019 to 21 Nov 2019 (Full Time)

Duration: 5 days

When:

Nov: 04, 05, 14, 15, 21

Time:

9.00am - 5.00pm

KIE OptaPlanner Deep Dive - Advanced Topics

2.4 WORKSHOP SEARCH REASONING

KIE OptaPlanner Deep Dive

- **Advanced Topics**

Compatibility

OptaPlanner works on any Java Virtual Machine

Standard Java

- Linux
- Mac
- Windows
- ...

Enterprise Java

- WildFly
- OPENSHIFT
- jetty://
- ...

JVM languages

- Java
- Scala
- Groovy
- ...

KIE functionality overview

What are the KIE projects?

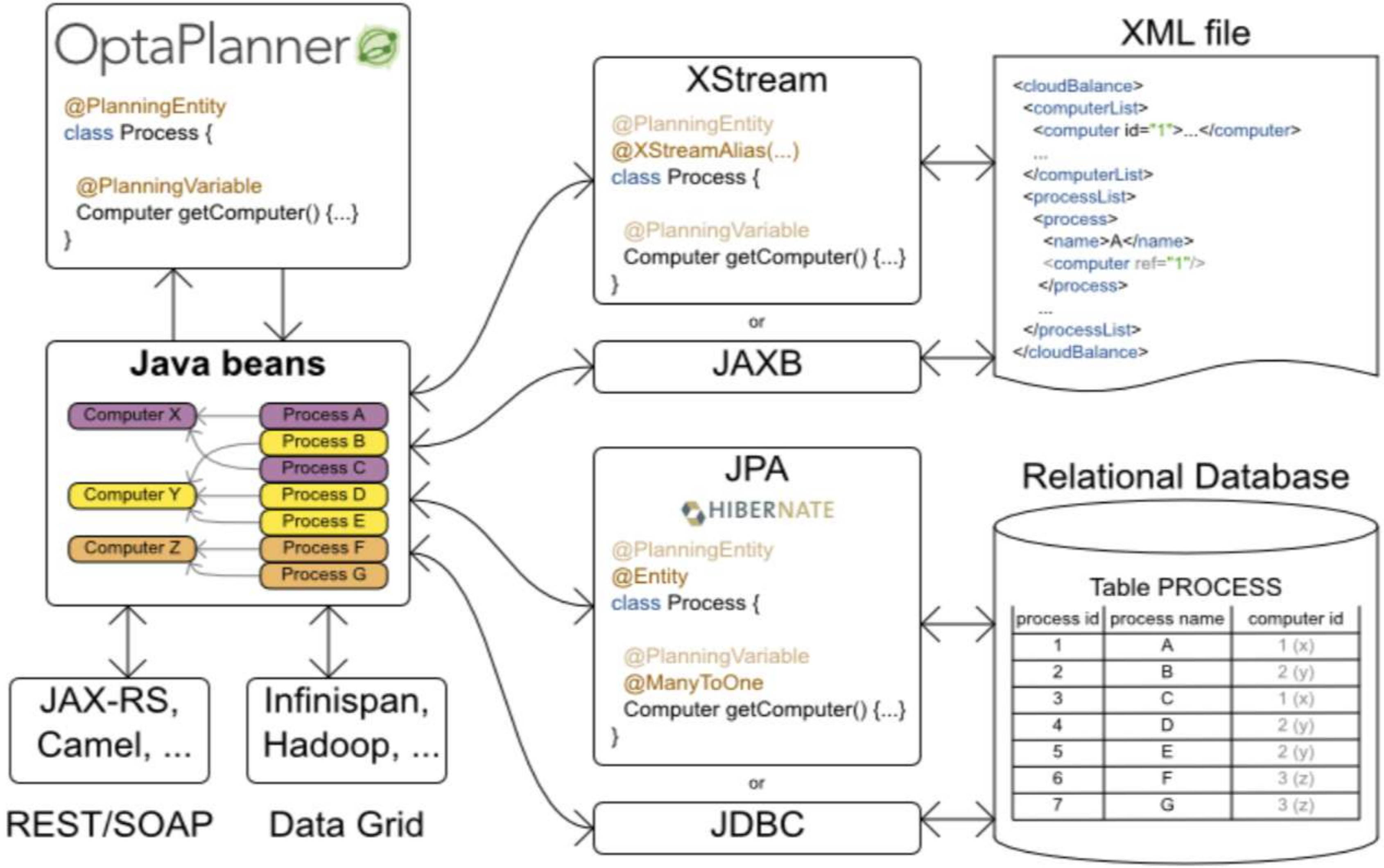
Drools Rule engine and Complex Event Processing Example: insurance rate calculation	**Drools Workbench** Design rules, decision tables, ...	**Drools Execution Server** REST/JMS service for business rules	
OptaPlanner Planning engine and optimization solver Example: employee rostering	**OptaPlanner Workbench** Design solvers, benchmarks, ...	**OptaPlanner Execution Server** REST/JMS service for optimization	redhat DM
jBPM Workflow engine Example: mortgage approval process	**jBPM Workbench** Design workflows, forms, ...	**jBPM Execution Server** REST/JMS service for workflows	redhat PAM

Lightweight, embeddable engines (jars)
which run in a Java VM

Web applications (wars)
which run on a Java Application Server

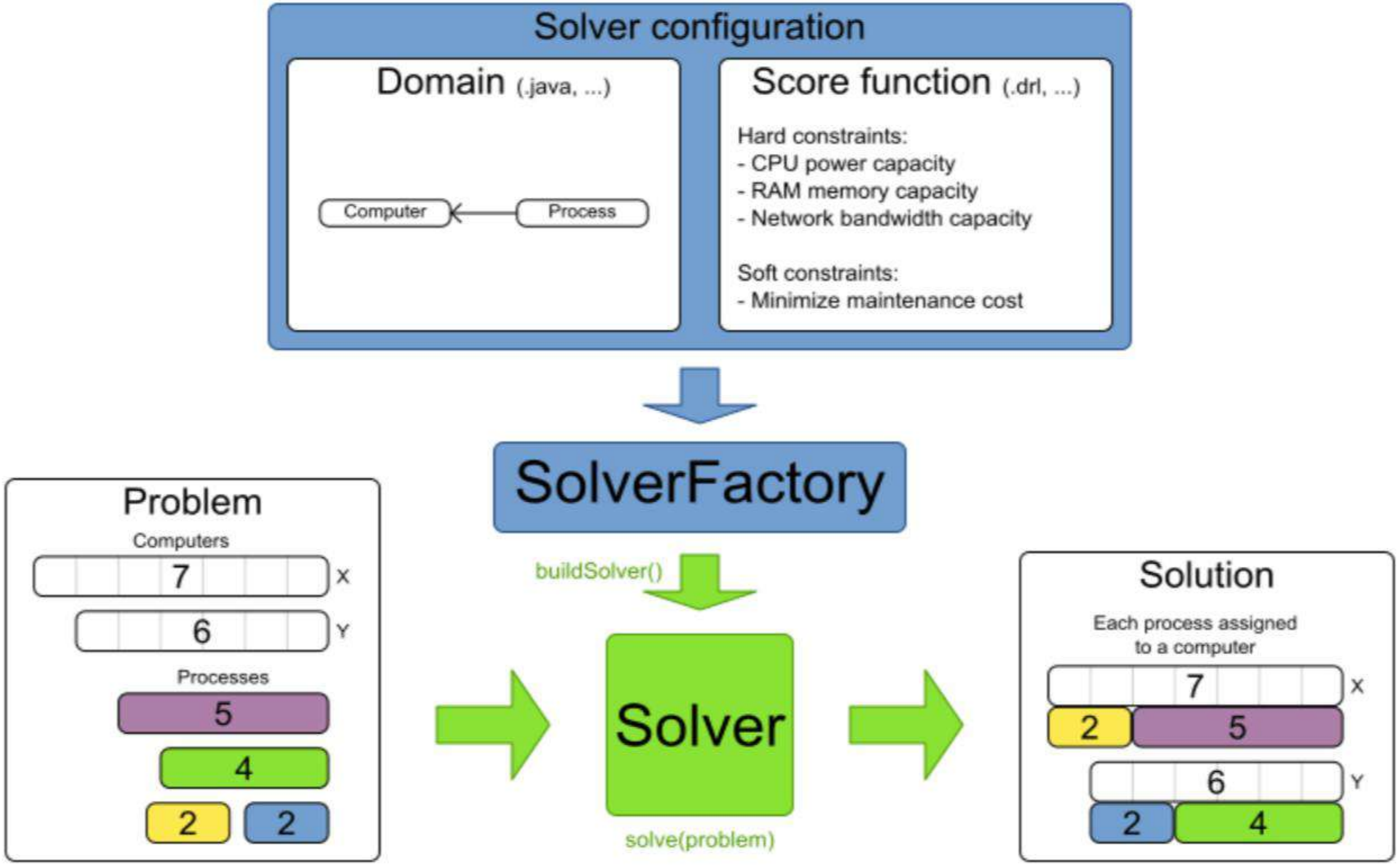
Integration overview

OptaPlanner combines easily with other Java and JEE technologies.



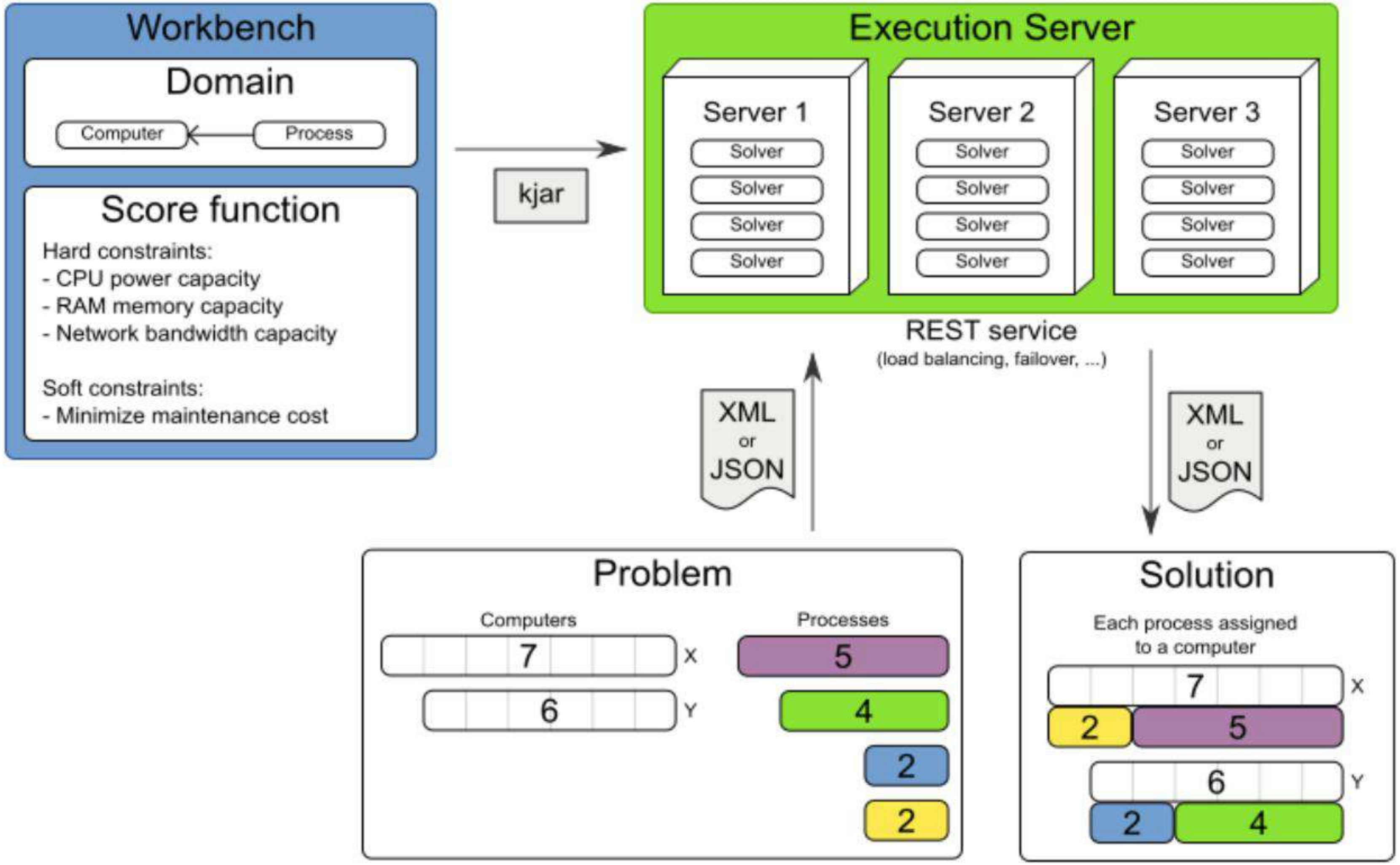
Input/Output overview

Use 1 SolverFactory per application and 1 Solver per dataset.



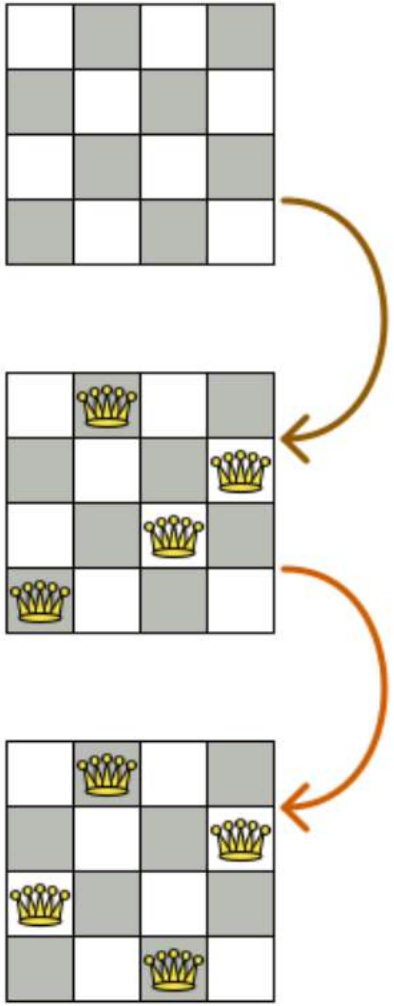
OptaPlanner Workbench and Execution Server

Define a use case in the Workbench and then deploy it to the Execution Server to solve it in the cloud.



General phase sequence

First a Construction Heuristic,
then a Local Search



Construction Heuristic

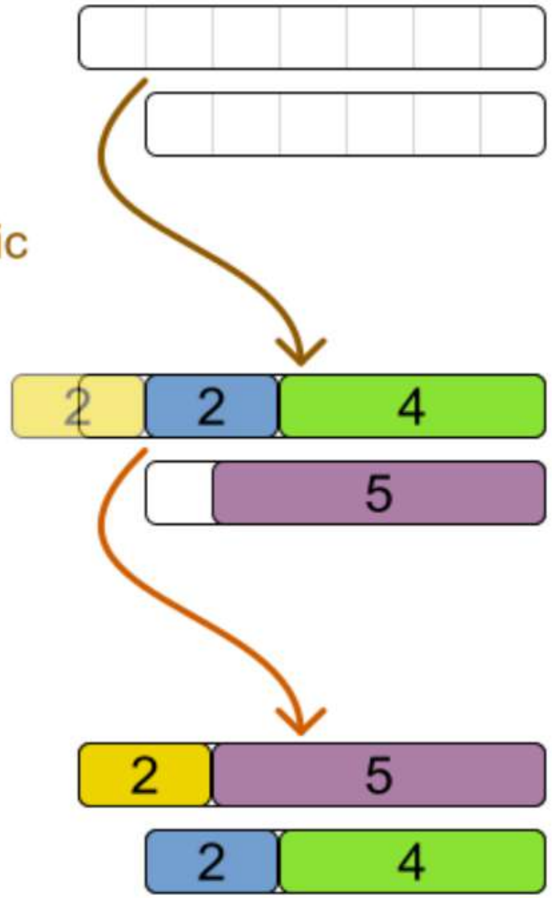
First Fit Decreasing

-1 hard / ? soft

Local Search

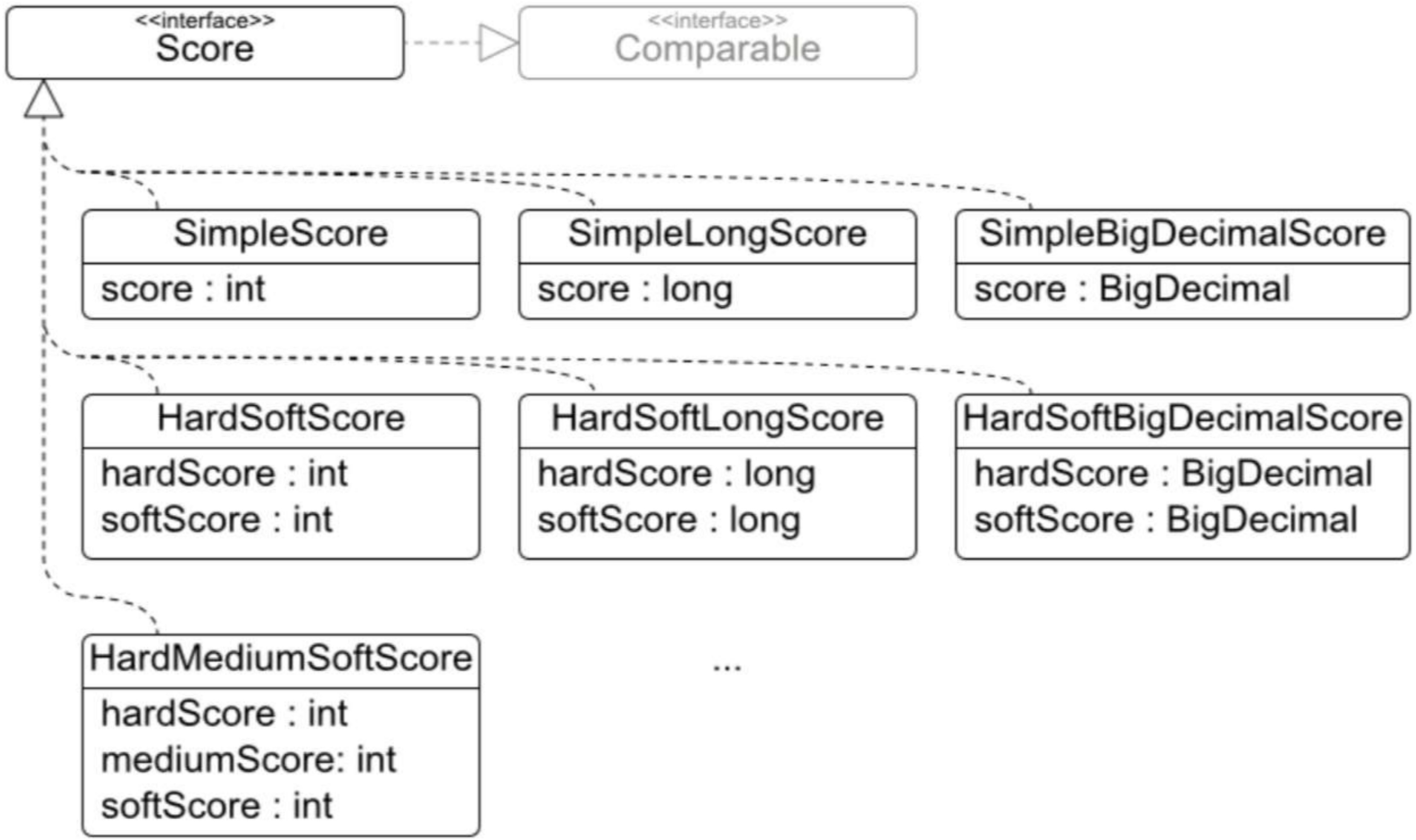
Tabu Search

0 hard / ? soft



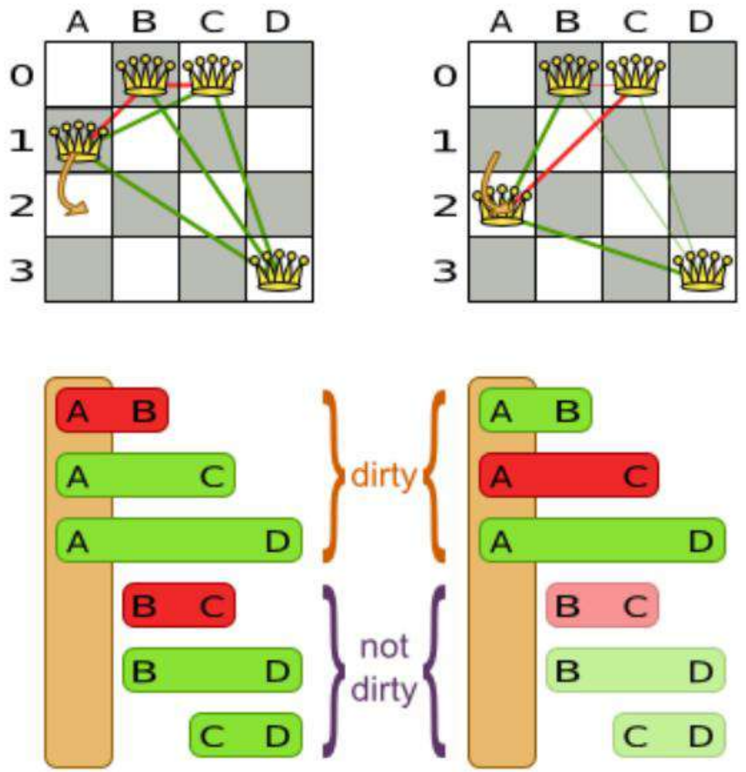
Score class diagram

Choose a Score implementation or write a custom one



Incremental score calculation

Incremental score calculation is much more scalable because only the delta is calculated.

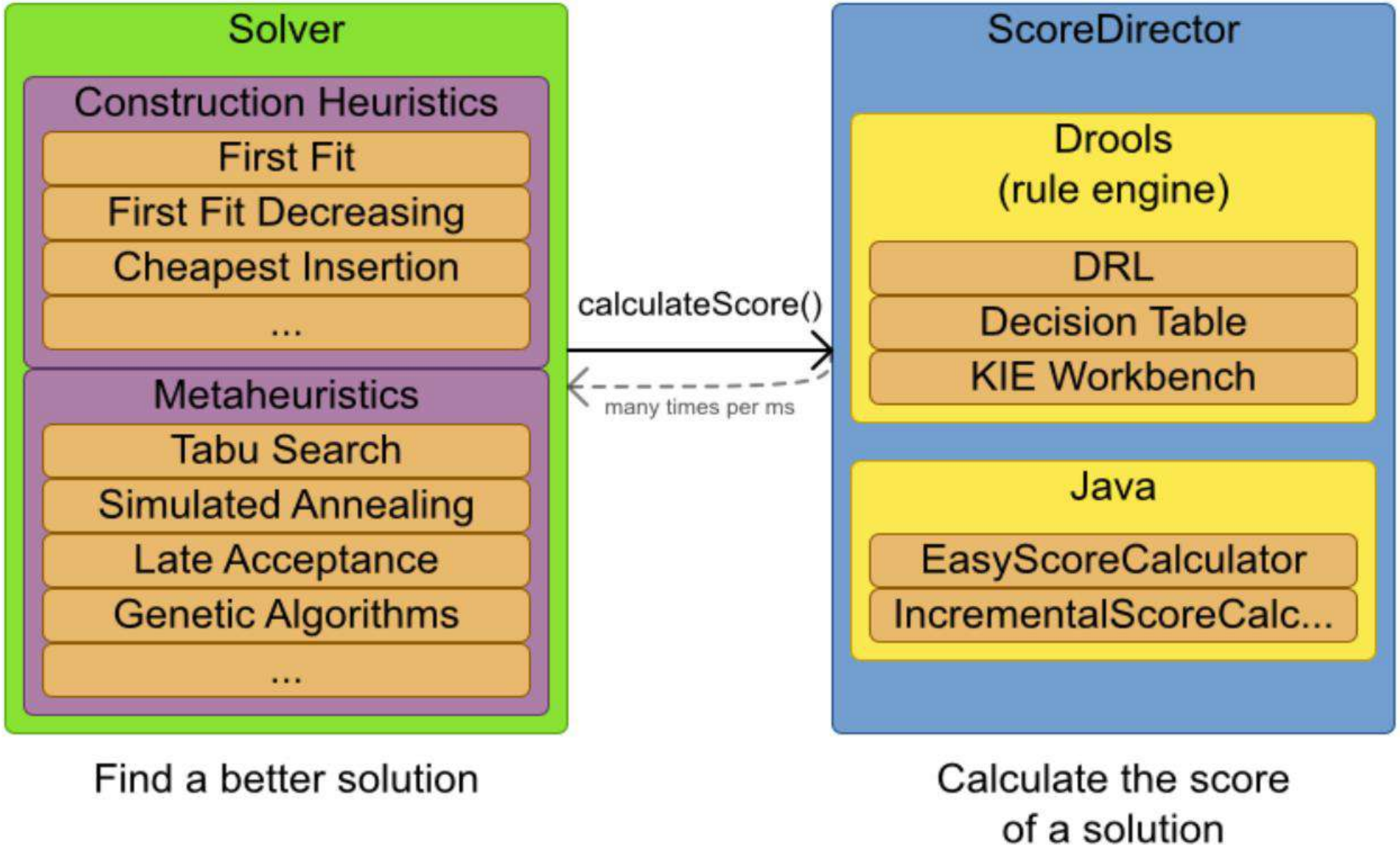


The rule engine
(with forward chaining)
only recalculates dirty tuples.

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
n	*n-1*	*of*	*n*(n-1)/2*	time /	*(n/2)*

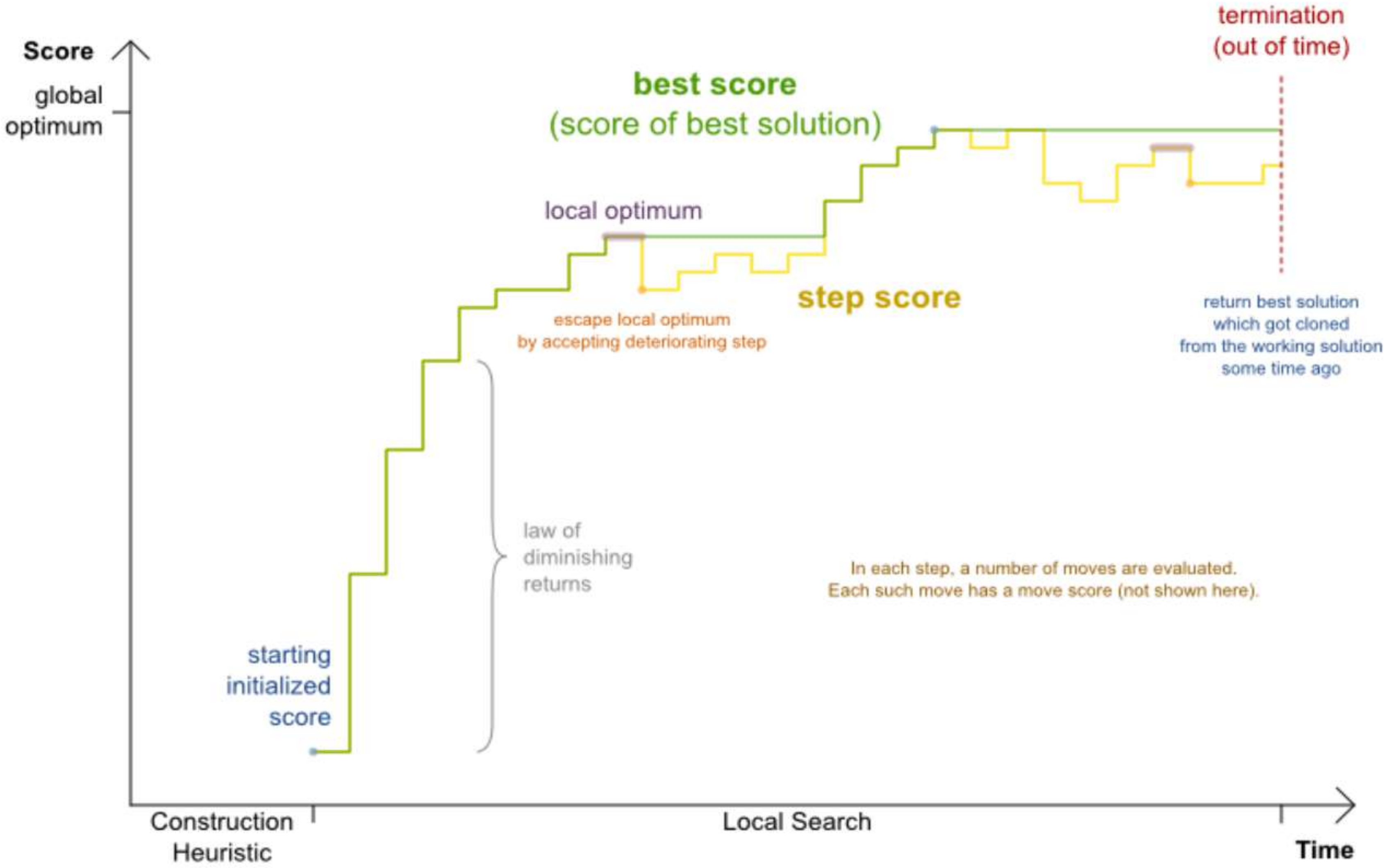
Architecture overview

The Solver wades through the search space of solutions efficiently.
The ScoreDirector calculates the score of every solution under evaluation.



Local Search score over time

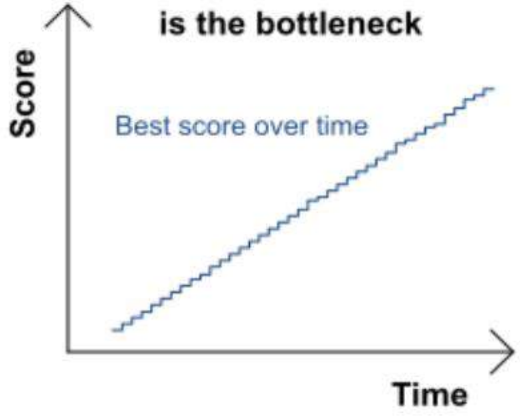
In 1 Local Search run, do not confuse starting initialized score, best score, step score and move score.



Let the best score statistic guide you

Where should we focus our energy to improve solution quality?

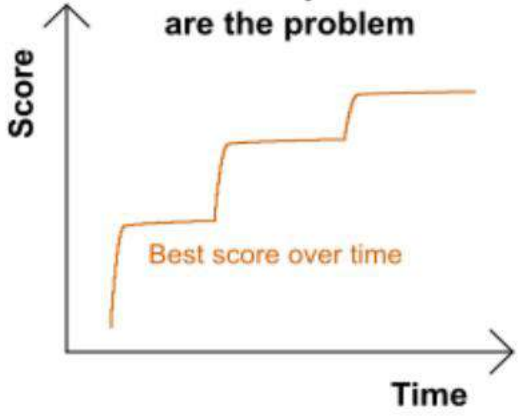
Performance is the bottleneck



Observations:
- Heavily improving every step
- No diminishing returns yet
- Solution not near optimal

Recommendations:
- Improve the score calculation speed: check info log
- Use better hardware
- Give it more time

Local optima are the problem



Observations:
- Some moves are lucky because they break out of a local optima

Recommendations:
- Add more moveSelectors
- Use constraint match statistic
- Add a course-grained custom move
- In score calculation, add a softer guiding constraint

This (meta)heuristic is depleted

Observations:
- Law of diminishing returns
- Solution likely near optimal

Recommendations:
- Benchmark other algorithms
- Power tweak parameters

Which one do I need to use?

Good answer: First Fit Decreasing with Late Acceptance

Better answer: try them all and use the best one

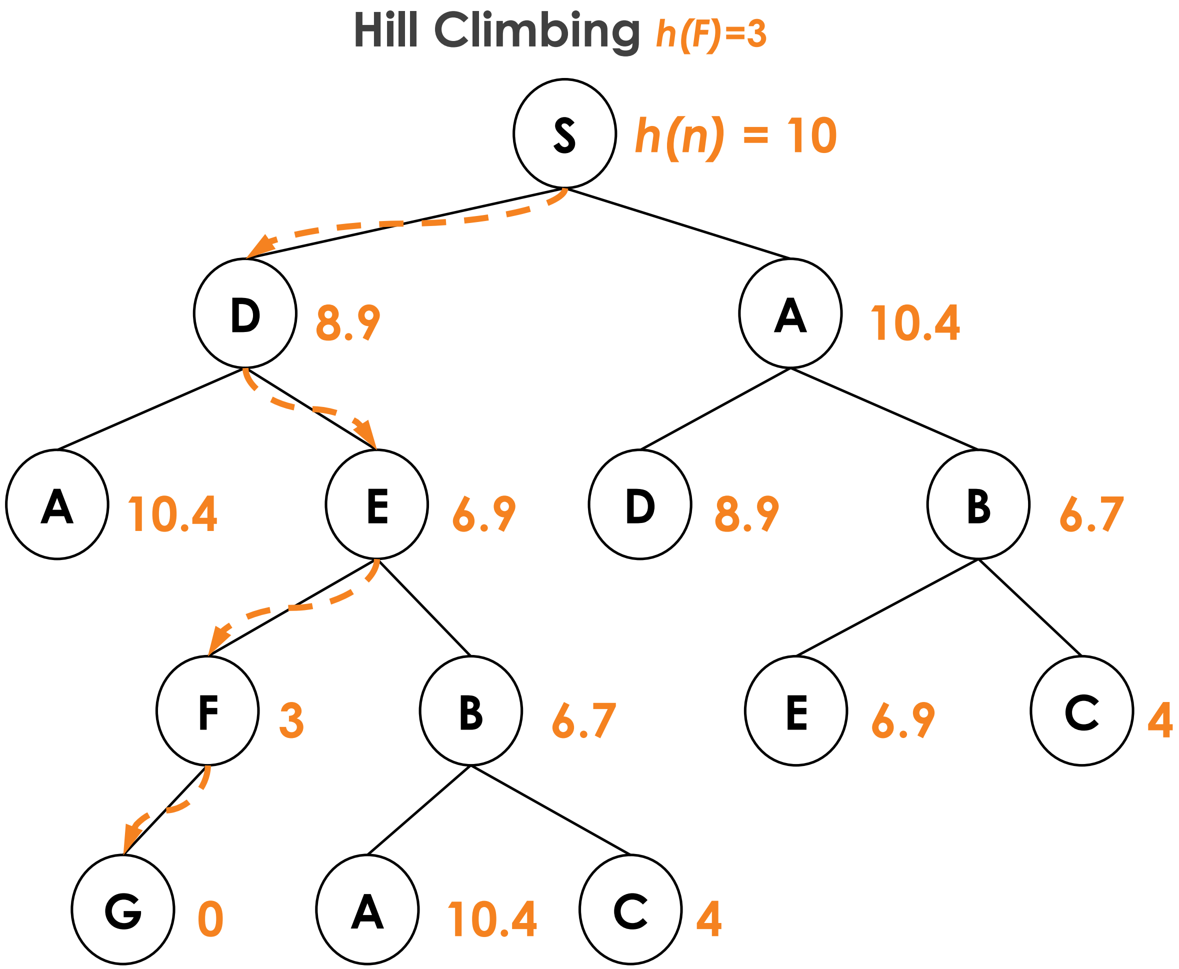
Planning problem use cases

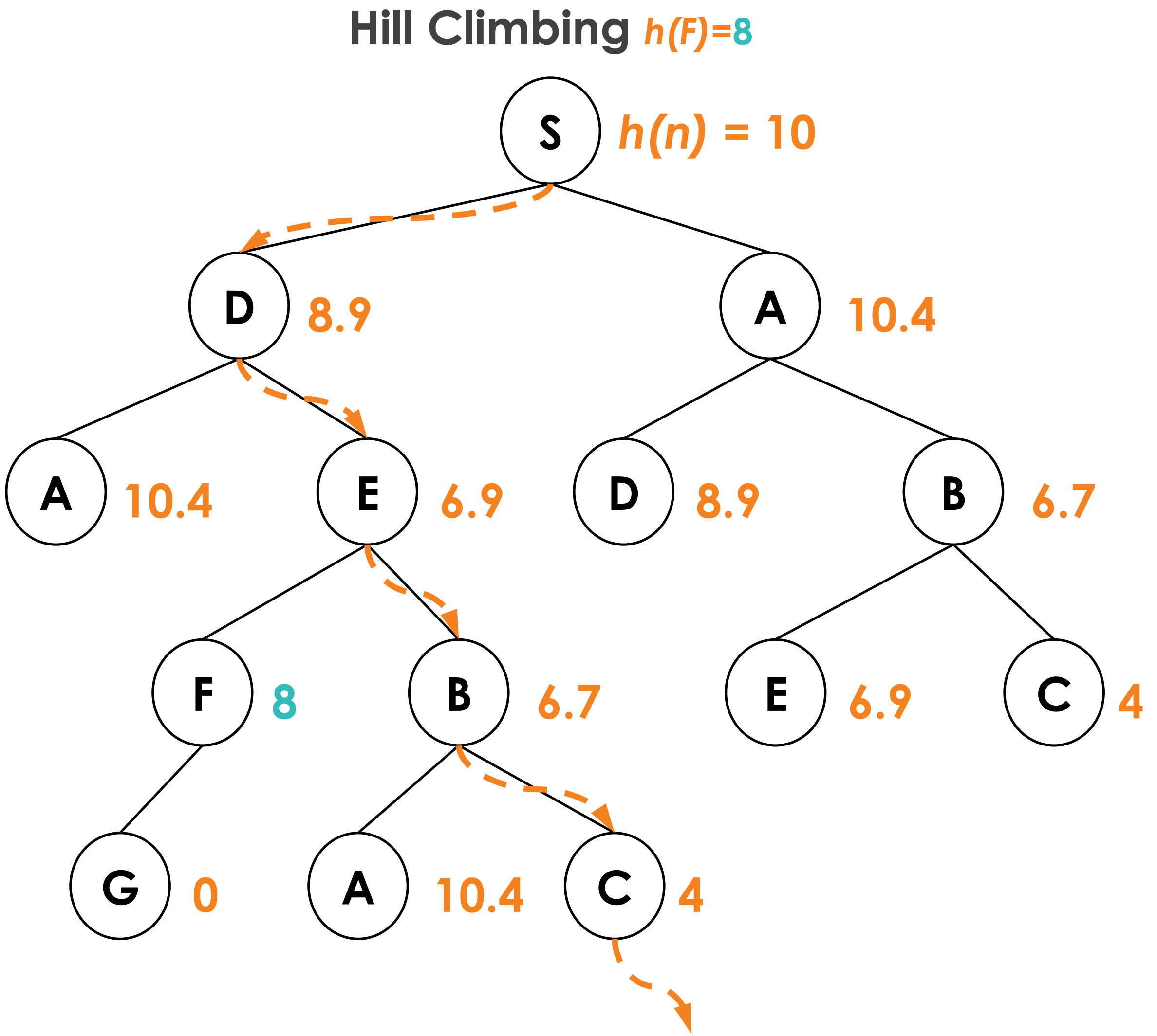
- **Agenda scheduling**: doctor appointments, court hearings, maintenance jobs, TV advertisements, ...
- **Educational timetabling**: lectures, exams, conference presentations, ...
- **Task assignment**: affinity/skill matchmaking for tax audits, wage calc, ...
- **Employee shift rostering**: nurses, repairmen, help desk, firemen, ...
- **Vehicle routing**: route trucks, buses, trains, boats, airplanes, ...
- **Bin packing**: fill containers, trucks, ships, storage warehouses, cloud computers nodes, prisons, hospitals, ...
- **Job shop scheduling**: assembly lines for cars, furniture, books, ...
- **Cutting stock**: minimize waste while cutting paper, steel, carpet, ...

Late Acceptance Hill Climbing (LAHC)

2.1 INFORMED SEARCH TECHNIQUES (2/2)

Late Acceptance Hill Climbing (LAHC)





2.1 INFORMED SEARCH TECHNIQUES (2/2)

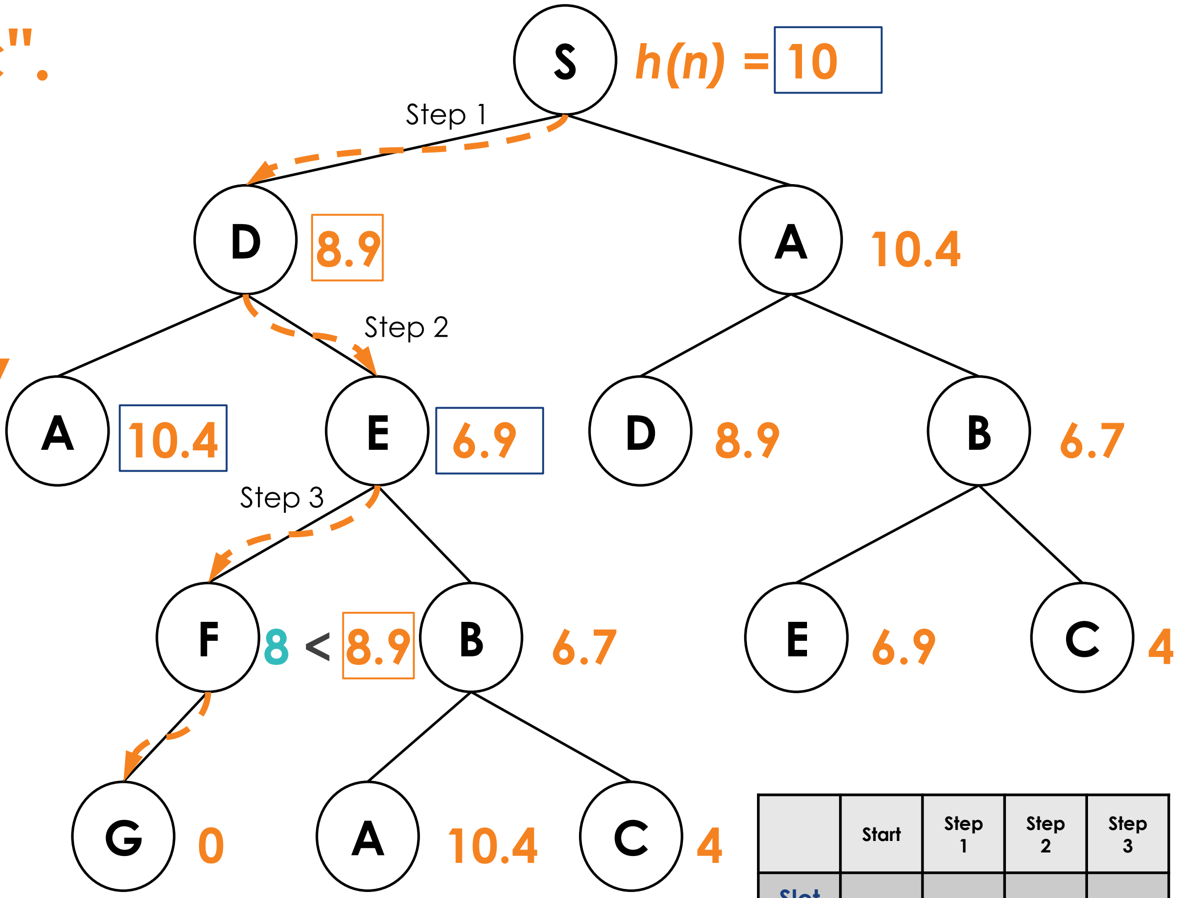
Late Acceptance Hill Climbing (LAHC)

E. K. Burke and Y. Bykov,. "The Late Acceptance Hill-Climbing Heuristic". European Journal of Operational Research.

- This paper introduces a new and very simple search methodology called Late Acceptance Hill-Climbing (LAHC).
- It is a local search algorithm, which accepts non-improving moves when a candidate cost function is better than it was a number of iterations before.

LAHC

LateAcceptanceSize = 2



	Start	Step 1	Step 2	Step 3
Slot 1	Nil	10	8.9	6.9
Slot 2	Nil	Nil	10	8.9

S 10<Nil D 8.9<Nil E 6.9<10 F 8<8.9

END OF APPENDICES