

OptaPlanner



Summary

[Analisys 3](#_Toc466233726)

[Planning Problems 4](#_Toc466233727)

[How to Install? 5](#_Toc466233728)

[Run the examples in an IDE 6](#_Toc466233729)

[Planner Configuration 7](#_Toc466233730)

# Analysis

OptaPlanner is **a lightweight, embeddable planning engine**. It´s an **open source software,** released under [the Apache Software License](https://www.optaplanner.org/code/license.html). It is written in Java, runs on any JVM and is available in [the Maven Central repository](https://www.optaplanner.org/download/download.html) too.

Optaplanner enables normal Java programmers to solve optimization problems efficiently using constraints satisfaction. Constraints apply on plain domain objects and can reuse existing code. Constraints differ from the common primitives of other programming languages in that they do not specify a step or sequence of steps to execute but rather the properties of the solution to be found. There’s no need to input difficult mathematical equations. Under the hood, OptaPlanner combines sophisticated optimization heuristics and metaheuristics (such as Tabu Search, Simulated Annealing and Late Acceptance) with very efficient score calculation.

As every organization faces scheduling puzzles like assigning a limited set of constrained resources (employees, assets, time and money) to provide products or services to customers, OptaPlanner optimizes such planning problems to **do more business with less resources.**

Some of the common use cases where OptaPlanner can be used are:

* [**Vehicle Routing**](https://en.wikipedia.org/wiki/Vehicle_routing_problem)**:** What is the optimal set of routes for a fleet of vehicles to traverse to deliver to a given set of customers?
* [**Employee Rostering**](https://en.wikipedia.org/wiki/Nurse_scheduling_problem)**:** Find an optimal way to assign employees to shifts with a set of hard and soft constraints.
* **Cloud Optimization:** What is the optimal assignment of processes to cloud computing resources (CPU, memory, disk)
* [**Job Scheduling**](https://en.wikipedia.org/wiki/Job_shop_scheduling)**:** Optimize the scheduling of jobs of varying processing times on a set of machines with varying processing power, trying to minimize the makespan.
* [**Bin Packing**](https://en.wikipedia.org/wiki/Bin_packing_problem)**:**  pack objects of different volumes into a finite number of bins or containers in a way that minimizes the number of bins used

# Planning Problems

Optaplanner is used to solve planning problems, and to these problems we call them [NP-hard](https://en.wikipedia.org/wiki/NP-hardness) problems. They had this name because implies that the time required to solve these problems using any currently known algorithm increases very quickly as the size of the problem grows (e.g. adding a destination to a vehicle routing problem, adding a shift to an employee rostering problem). This is one of the principal [unsolved problems in computer science today](https://en.wikipedia.org/wiki/List_of_unsolved_problems_in_computer_science).

As it is impossible to solve these problems, or find the best solution to these problems, in a limited timespan when scaling out, Business Resource Planner uses a set of sophisticated optimization heuristics and meta-heuristics (like [Tabu Search](https://en.wikipedia.org/wiki/Tabu_search), [Simulated Annealing](https://en.wikipedia.org/wiki/Simulated_annealing) and [Late Acceptance](http://www.cs.stir.ac.uk/research/publications/techreps/pdf/TR192.pdf)) to find an optimal solution to these problems.

As said, every organization has these kind of scheduling problems, and there is a lot to gain from optimizing these problems. In the remainder of this post we will walk you through several steps to get you started with Business Resource Planner/OptaPlanner to find an optimal solution to your business problem and start increasing productivity, reducing costs and increasing customer satisfaction.

Usually, a planning problem has at least 2 levels of constraints:

* *A*(negative) hard constraint*must not be broken. For example:*1 teacher cannot teach 2 different lessons at the same time*.*
* *A*(negative) soft constraint*should not be broken if it can be avoided. For example:*Teacher A does not like to teach on Friday afternoon*.*

Some problems have positive constraints too:

* Apositive soft constraint (or reward)*should be fulfilled if possible. For example:*Teacher B likes to teach on Monday morning*.*

Some basic problems (such as N Queens) only have hard constraints. Some problems have 3 or more levels of constraints, for example hard, medium and soft constraints.

These constraints define the score calculation (AKA fitness function) of a planning problem. Each solution of a planning problem can be graded with a score**. With Planner, score constraints are written in an Object Orientated language, such as Java code or Drools rules.** Such code is easy, flexible and scalable.

# How to Install?

* Download a release zip of OptaPlanner from [the OptaPlanner website](http://www.optaplanner.org/).
* Unzip it.
* Open the directory **examples** and run the script.

Linux or Mac:

$ cd examples

$ ./runExamples.sh

Windows:

$ cd examples

$ RunExamples.bat

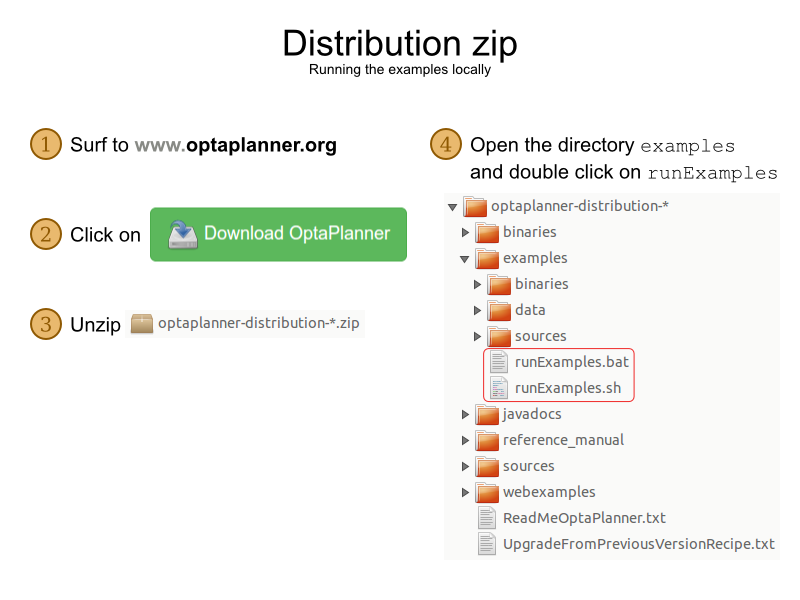
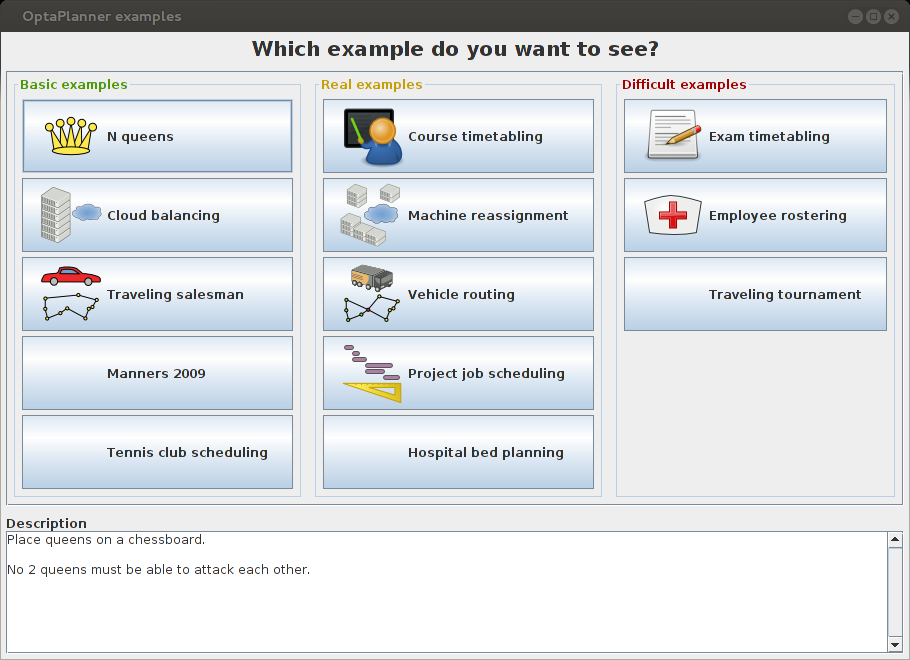


Figura 1 - Steps to Install

After these steps, it will appear a new interface that show examples that able any person to test a specific case.



Besides the GUI examples, there are also a set of web examples to try out:

* Download a JEE application server, such as JBoss EAP or WildFly [http://www.wildfly.org/] and unzip it.
* Download a release zip of OptaPlanner from the OptaPlanner website [http://
* www.optaplanner.org] and unzip it.
* Open the directory webexamples and deploy the optaplanner-webexamples-\*.war file on the JEE application server.
* Surf to http://localhost:8080/optaplanner-webexamples-\*/ (replace the \* with the actual version

# Run the examples in an IDE

Optaplanner have the capability of being configured in a familiar IDE like IntelliJ, Eclipse, and NetBeans, in a few steps you can run this program in environment that you are more comfortable with.

**In IntelliJ IDEA and NetBeans:**

* Open the file **examples/sources/pom.xml** as a new project, the maven integration will take care of the rest.
* In Eclipse, open a new project for the directory **examples/sources**.
* Add all the jars to the classpath from the directory **binaries** and the directory **examples/binaries**, except for the file **examples/binaries/optaplanner-examples-\*.jar**.
* Add the Java source directory **src/main/java** and the Java resources directory **src/main/resources**.

Or you can build it from source:

1. [Set up Git](http://help.github.com/set-up-git-redirect) and clone optaplanner from GitHub (or alternatively, download [the zipball](https://github.com/droolsjbpm/optaplanner/zipball/master)):

* $ git clone git@github.com: droolsjbpm/optaplanner.git optaplanner.

# Planner Configuration

Solving a planning problem with Planner consists out of 5 steps:

1. **Model your planning problem** as a class that implements the interface Solution, for example

the class NQueens.

2. **Configure a Solver**, for example a First Fit and Tabu Search solver for any NQueens instance.

3. **Load a problem data set** from your data layer, for example a 4 Queens instance. That is the

planning problem.

4. **Solve it** with Solver. Solve (planning Problem) which returns the best solution found.



**Solver Configuration by Java API**

A solver configuration can also be configured with the SolverConfig API. This is especially useful to change some values dynamically at runtime. For example, to change the running time based on user input, before building the Solver:

 SolverFactory solverFactory = SolverFactory.createFromXmlResource(

                "org/optaplanner/examples/nqueens/solver/nqueensSolverConfig.xml");

    SolverConfig solverConfig = solverFactory.getSolverConfig();

        TerminationConfig terminationConfig = solverConfig.getTerminationConfig();

        terminationConfig.setMinutesSpentLimit(userInput);

        Solver solver = solverFactory.buildSolver();

Every element in the solver configuration XML is available as a \*Config class or a property on a:

* Config class in the package namespaceorg.optaplanner.core.config.
* These \*Config classes are the Java representation of the XML format.
* They build the runtime components (of the package namespace org.optaplanner.core.impl) and assemble them into an efficient Solver.