

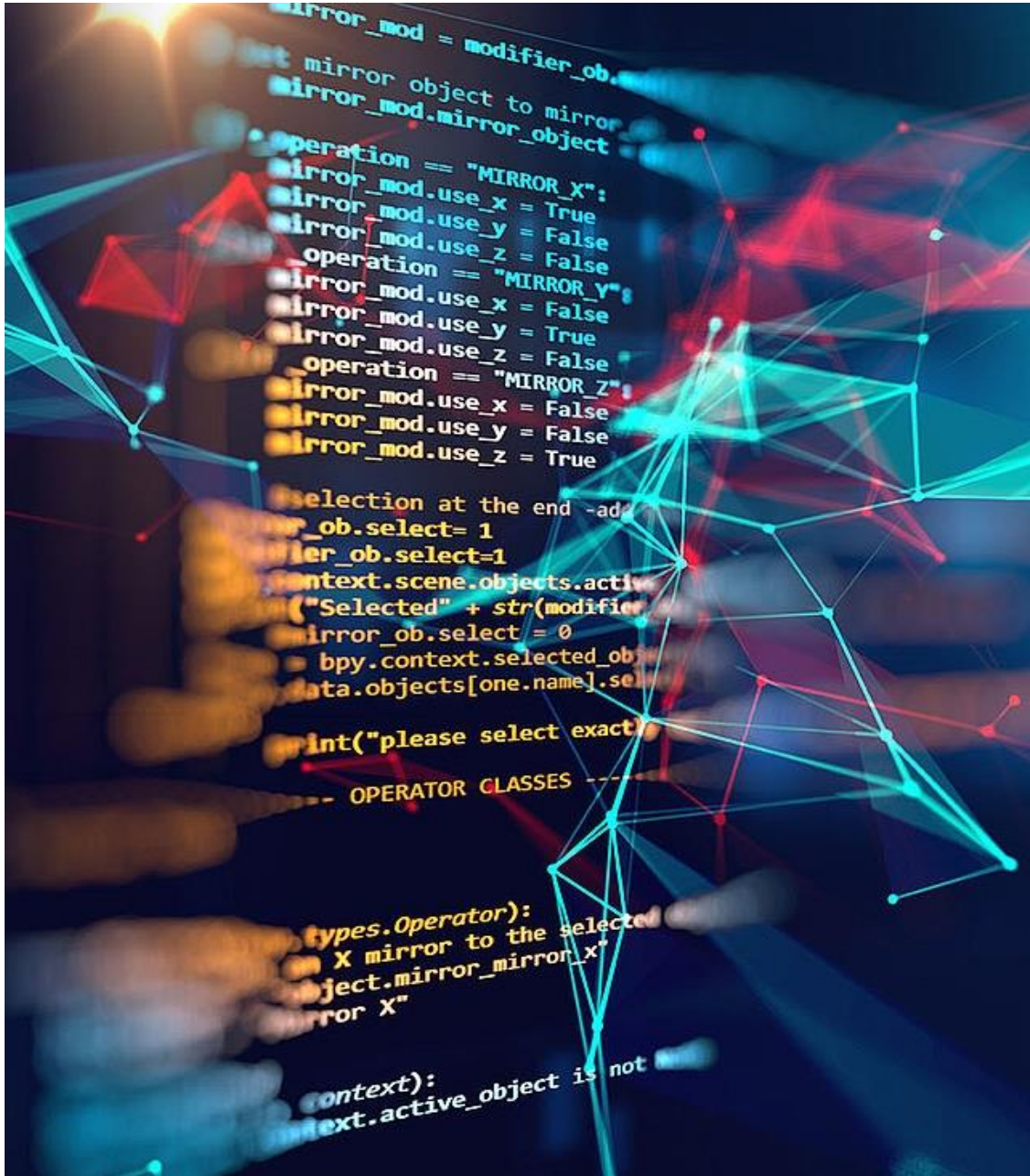
SIDION

Zuhören. Analysieren. Beraten.



CONSTRAINT SATISFACTION PROBLEMS – NEVER CODE THIS AT HOME!

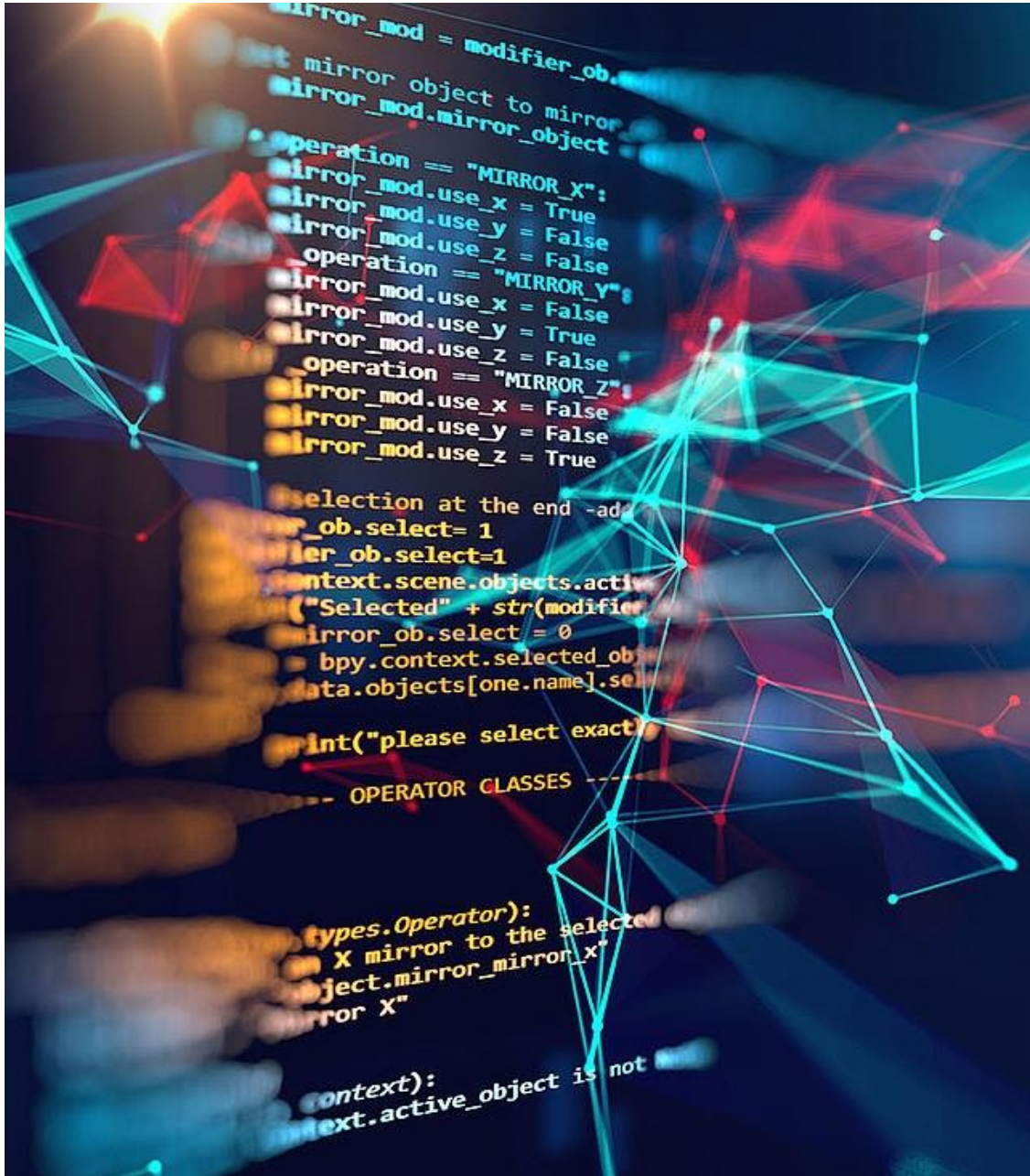
Matthias Koch | Java Forum Stuttgart | 07.07.2022



CONSTRAINT SATISFACTION PROBLEMS

Bedingungserfüllungsprobleme, CSPs

- Variablen
- Domänen (können für jede Variable unterschiedlich sein)
- Constraints



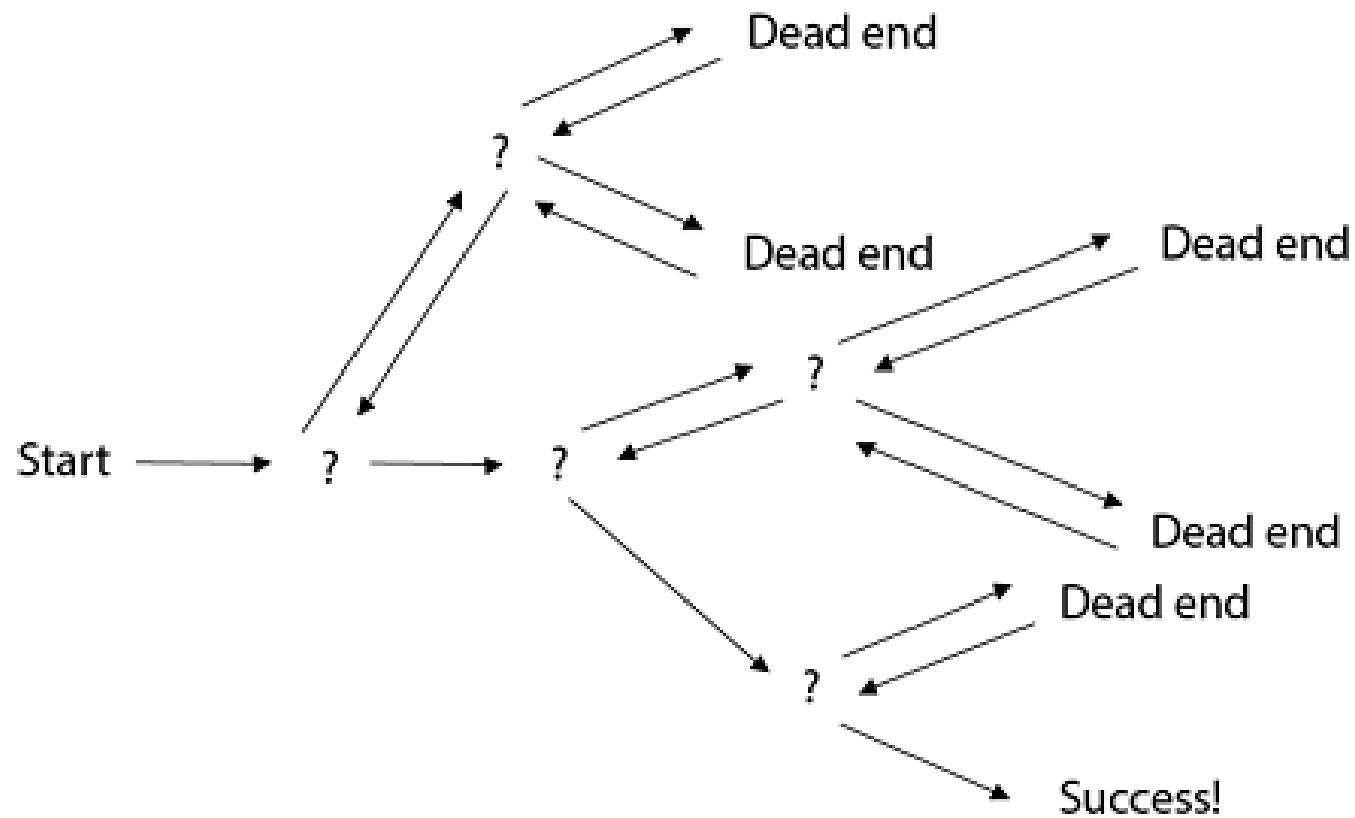
CONSTRAINT SATISFACTION PROBLEMS

- Sind schon sehr alt
 - AC-3 Algorithmus (Alan Mackworth, 1977)
- Es gibt sehr viele Anwendungsfälle dafür:
 - Operations Research (z.B. Travelling Salesman Problem)
 - Künstliche Intelligenz
 - Terminplanung
 - Spielpläne von Sportligen
 - Spiele
 - ...
 - SAT oder 3SAT → damit potentiell NP-vollständig

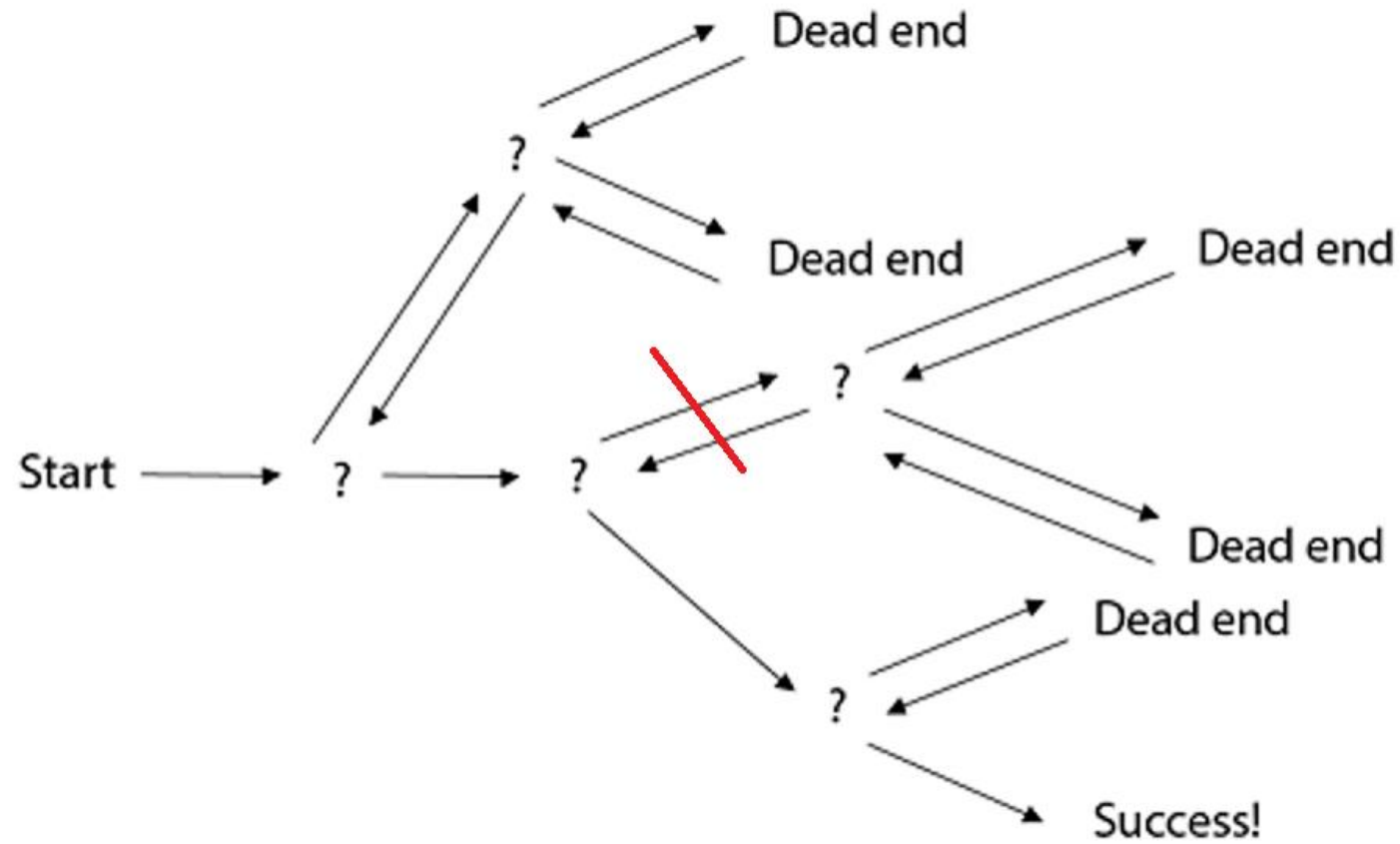
IMPLEMENTIERUNG EINER LÖSUNG

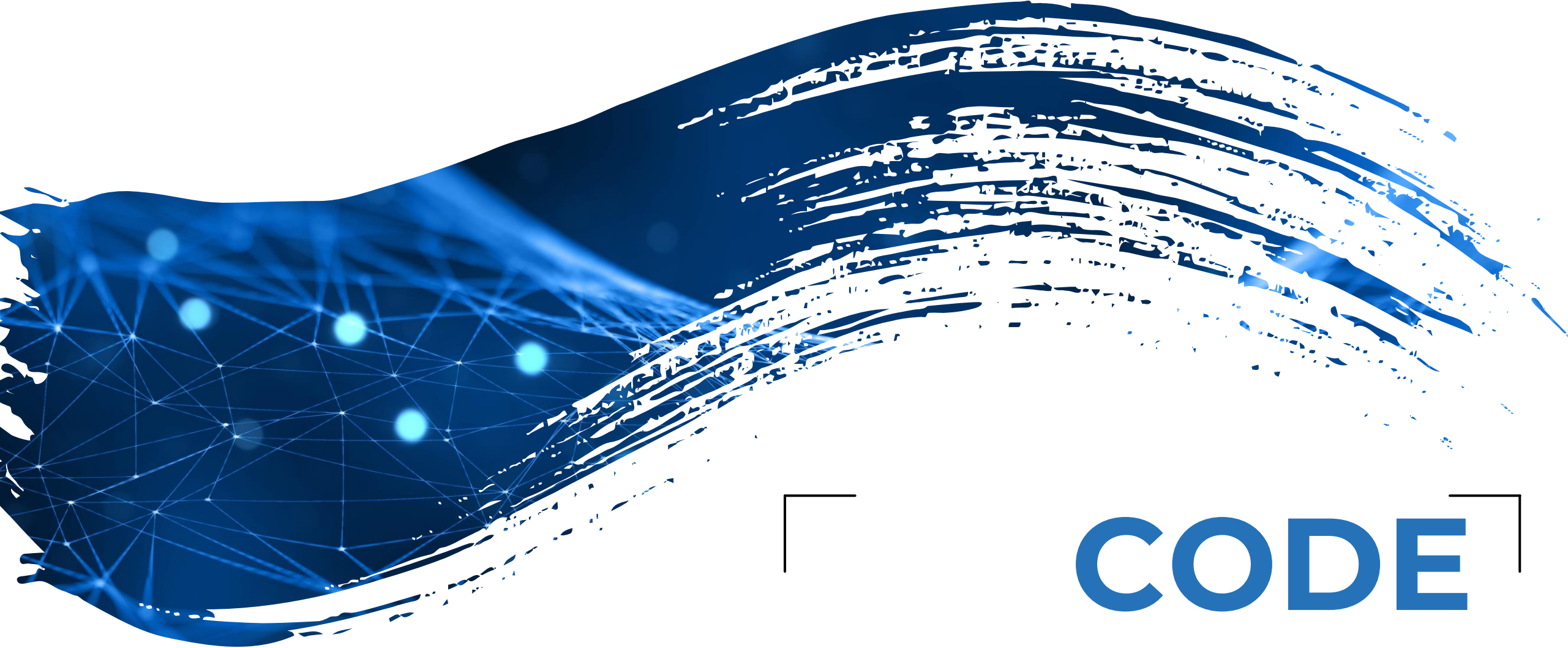
- Ein CSP kann immer durch Backtracking gelöst werden. Aber das ist nicht immer die schnellste Lösung!
- Es gibt Programmiersprachen wie Prolog und Picat, die Elemente zur einfachen Umsetzung eines Backtracking Algorithmus besitzen.
- In Programmiersprachen wie Java ist das explizit zu implementieren. Naive Implementierung: Eine Schleife über den Wertebereich jeder Variable.

BACKTRACKING PRINZIP



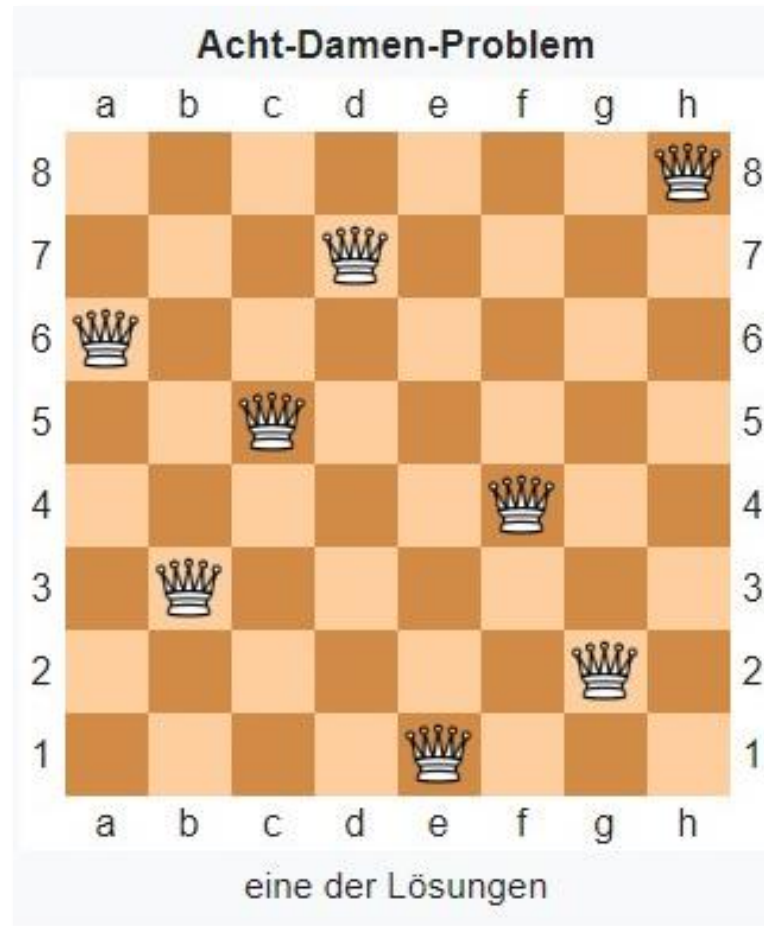
IDEE: PRUNING





CODE DEMO

DAMENPROBLEM



EINE MÖGLICHE IMPLEMENTIERUNG

```
public static Stream<List<Pos>> nQueens(int n) {  
    return IntStream.rangeClosed(1, n)  
        .mapToObj(i -> i)  
        .sorted(Comparator.reverseOrder())  
        .reduce( Stream.of(new ArrayList<Pos>()), (r, i) -> r.flatMap(qs -> IntStream.rangeClosed(1, n).mapToObj( j ->  
            new Pos(j, i))  
        . flatMap(p -> (qs.stream()).allMatch( q -> (q.x != p.x && q.y != p.y && abs(q.x -p.x) != abs(q.y - p.y))) ?  
            Stream.of(Stream.concat(qs.stream(), Stream.of(p))  
        .collect(Collectors.toList())) : Stream.<List<Pos>> empty()))), Stream::concat);  
}
```

SUDOKU

	3							
			1	9	5			
		8					6	
8				6				
4			8					1
				2				
	6					2	8	
			4	1	9			5
							7	

SUDOKU

1 2 5 6 7 9	3	1 2 4 5 6 7 9	2 6 7 8	4 7 8	2 6 7 8	1 4 5 7 8 9	1 2 4 5 9	2 4 7 8 9	1
2 6 7	2 4 7	2 4 7	1	9	5	3 4 7 8	2 3 4	2 3 4 7 8	2
1 2 5 7 9	1 2 4 5 7 9	8	2 3 7	3 4 7	2 3 4 7	1 4 5 7 9	6	2 3 4 7 9	3
8	1 2 5 7 9	1 2 3 5 7 9	3 5 7 9	6	1 4 7	3 4 5 7 9	2 3 4 5 9	2 3 4 7 9	4
4	2 5 7 9	2 3 5 6 7 9	8	3 5 7	3 7	3 5 6 7 9	2 3 5 9	1	5
1 5 6 7 9	1 5 7 9	1 5 6 7 9	3 5 7 9	2	1 4 7	3 4 5 6 7 8 9	3 4 5 9	3 4 6 7 8 9	6
1 5 7 9	6	1 4 5 7 9	3 5 7	3 5 7	3 7	2	8	3 4 9	7
2 3 7	2 7 8	2 3 7	4	1	9	3 6	3	5	8
1 2 3 5 9	1 2 4 5 8 9	1 2 3 4 5 9	2 3 5 6 7 8	3 5 8	2 3 6 8	1 4 6 9	7	3 4 6 9	9
a	b	c	d	e	f	g	h	i	

MAP OF AUSTRALIA



SEND + MORE = MONEY

$$\begin{array}{r} \text{SEND} \\ + \text{MORE} \\ \hline = \text{MONEY} \end{array}$$



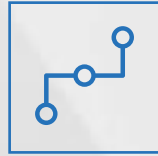
ERSTELLUNG VON SPORTLIGEN

- Einige kommerzielle Produkte vorhanden
- Optimierung des Spielplans nach folgenden Kriterien:
 - Heim- und Auswärtsspiele (approx.) alternierend
 - Ortskonflikte vermeiden
 - Top-Spiele gleichmäßig über die Saison verteilen
 - Minimierung der Reisewege

CHOCO-SOLVER



Erster Release 2003!



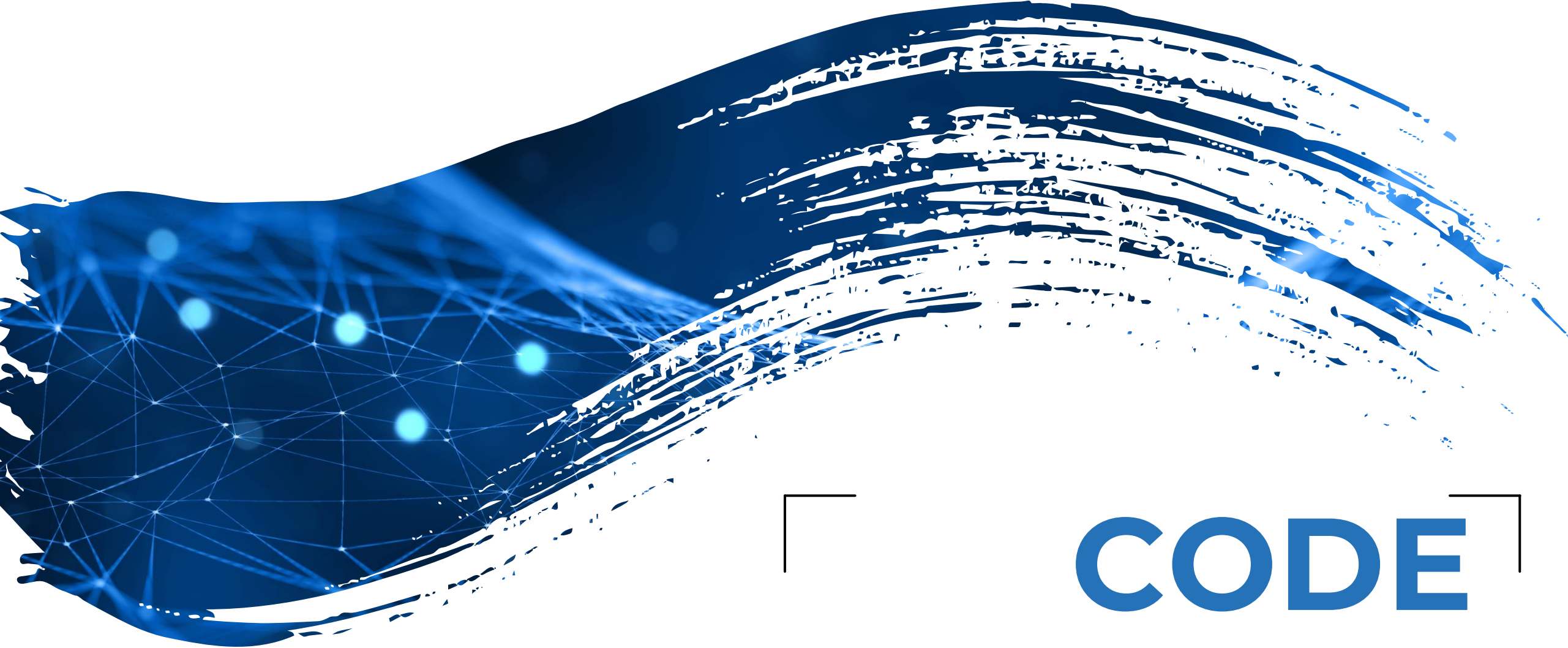
Open Source (BSD4-
Clause-Lizens)



IMT Atlantique



Choco-solver



CODE DEMO

CHOCO-SOLVER CHEAT SHEET

Model

- `Model model = new Model („n-queens problem“);`

Variables + Domains

- `IntVar[] vars = model.intVarArray("Q", GRID_SIZE, 1, GRID_SIZE, false);`
- Variablen gibt es vom Typ Integer, Boolean, Set und Real

Constraints

- `model.arithm(vars[i], "!=" , vars[j], "-", j - i).post();`
- Es gibt arithmetische, relationale und logische Expressions

Solver

- `Solver solver = model.getSolver();`
- `List<Solution> solutions = solver.findAllSolutions();`
- `Solver.showStatistics();`

SEARCH STRATEGIES - BEISPIEL

org.choco-solver choco-solver 4.10.8

MODULE PACKAGE **CLASS** USE TREE DEPRECATED INDEX HELP

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD SEARCH:

Module org.chocosolver.solver
Package org.chocosolver.solver.search.strategy.selectors.variables
Class DomOverWDeg

java.lang.Object
 org.chocosolver.solver.search.strategy.selectors.variables.AbstractCriterionBasedVariableSelector
 org.chocosolver.solver.search.strategy.selectors.variables.DomOverWDeg

All Implemented Interfaces:
ICause, IMonitorContradiction, IMonitorRestart, ISearchMonitor, VariableSelector<IntVar>, IVariableMonitor<Variable>

Direct Known Subclasses:
DomOverWDegRef

```
public class DomOverWDeg
extends AbstractCriterionBasedVariableSelector
implements IMonitorRestart
```

Implementation of DowOverWDeg[1].

[1]: F. Boussemart, F. Hemery, C. Lecoutre, and L. Sais, Boosting Systematic Search by Weighting Constraints, ECAI-04.

Since:
12/07/12

Author:
Charles Prud'homme

ALTERNATIVE?

Maven Repository Search for groups, artifacts, categories Search

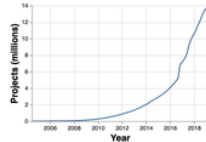
Home » Categories » Constraint Solvers

Constraint Solvers

Sort: **popular** | newest

-  **1. OptaPlanner Core** 95 usages
[org.optaplanner » optaplanner-core](#) Apache
OptaPlanner solves planning problems. This lightweight, embeddable planning engine implements powerful and scalable algorithms to optimize business resource scheduling and planning. Aggregate dependency to bring in optaplanner-core and all the score directors.
Last Release on May 10, 2022
-  **2. Choco Solver** 24 usages
[org.choco-solver » choco-solver](#) BSD
Open-source constraint solver.
Last Release on Jan 10, 2022
-  **3. JaCoP** 2 usages
[org.jacop » jacop](#) AGPL
Finite domain solver written in java.
Last Release on Nov 20, 2020
-  **4. JaCoP** 1 usages
[de.sciss » jacop](#) AGPL
Java Constraint Programming solver (fork)
Last Release on Sep 1, 2018
-  **5. Poirot** AGPL
[de.sciss » poirot](#)
A Scala front-end for the JaCoP constraints solver library
Last Release on Jun 19, 2020

Indexed Artifacts (28.1M)



Popular Categories

- Aspect Oriented
- Actor Frameworks
- Application Metrics
- Build Tools
- Bytecode Libraries
- Command Line Parsers
- Cache Implementations
- Cloud Computing
- Code Analyzers
- Collections
- Configuration Libraries
- Core Utilities
- Date and Time Utilities
- Dependency Injection
- Embedded SQL Databases
- HTML Parsers
- HTTP Clients
- I/O Utilities
- JDBC Extensions
- JDBC Pools
- JPA Implementations
- JSON Libraries
- JVM Languages

SOFT CONSTRAINTS

What can OptaPlanner do?

OptaPlanner optimizes plans and schedules with *hard constraints* and *soft constraints*.

It **reduces costs** substantially, improves **service quality**, fulfills **employee wishes** and **lowers carbon emissions**.

Vehicle routing (VRP)

Quicker routes for a fleet of vehicles.

[Learn more](#)

Employee rostering

Assign shifts to employees by skills and availability.

[Learn more](#)

Maintenance scheduling

Timely upkeep of machinery and equipment.

[Learn more](#)

Conference scheduling

Schedule speakers and talks by availability and topic.

[Learn more](#)

School timetabling

Compacter schedules for teachers and students.

[Learn more](#)

Task assignment

Assign tasks by priority, skills and affinity.

[Learn more](#)

Cloud optimization

Bin packing and defragmentation of cloud resources.

[Learn more](#)

Job shop scheduling

Reduce makespan for assembly lines.

[Learn more](#)



FRAGEN

Mitdenken
Mitgestalten
Mitarbeiten
bei sidion

QUELLENANGABEN

- <https://choco-solver.org/>
- David Kopec – Algorithmen in Java, Rheinwerk Computing, 2021
- <https://www.javatpointom/backtracking-introduction>
- <https://de.wikipedia.org/wiki/Damenproblem>
- <https://de.wikipedia.org/wiki/Sudoku>
- <https://www.chegg.com/homework-help/algorithms-1st-edition-chapter-10.2-solutions-9780023606922>
- <https://www.spiegel.de/wissenschaft/mensch/kryptischer-geldwunsch-raetsel-der-woche-a-1236245.html>
- <https://javadoc.io/doc/org.choco-solver/choco-solver/latest/org.chocosolver.solver/org/chocosolver/solver/search/strategy/selectors/variables/DomOverWDeg.html>
- <https://www.optaplanner.org/>



**VIELEN
DANK**