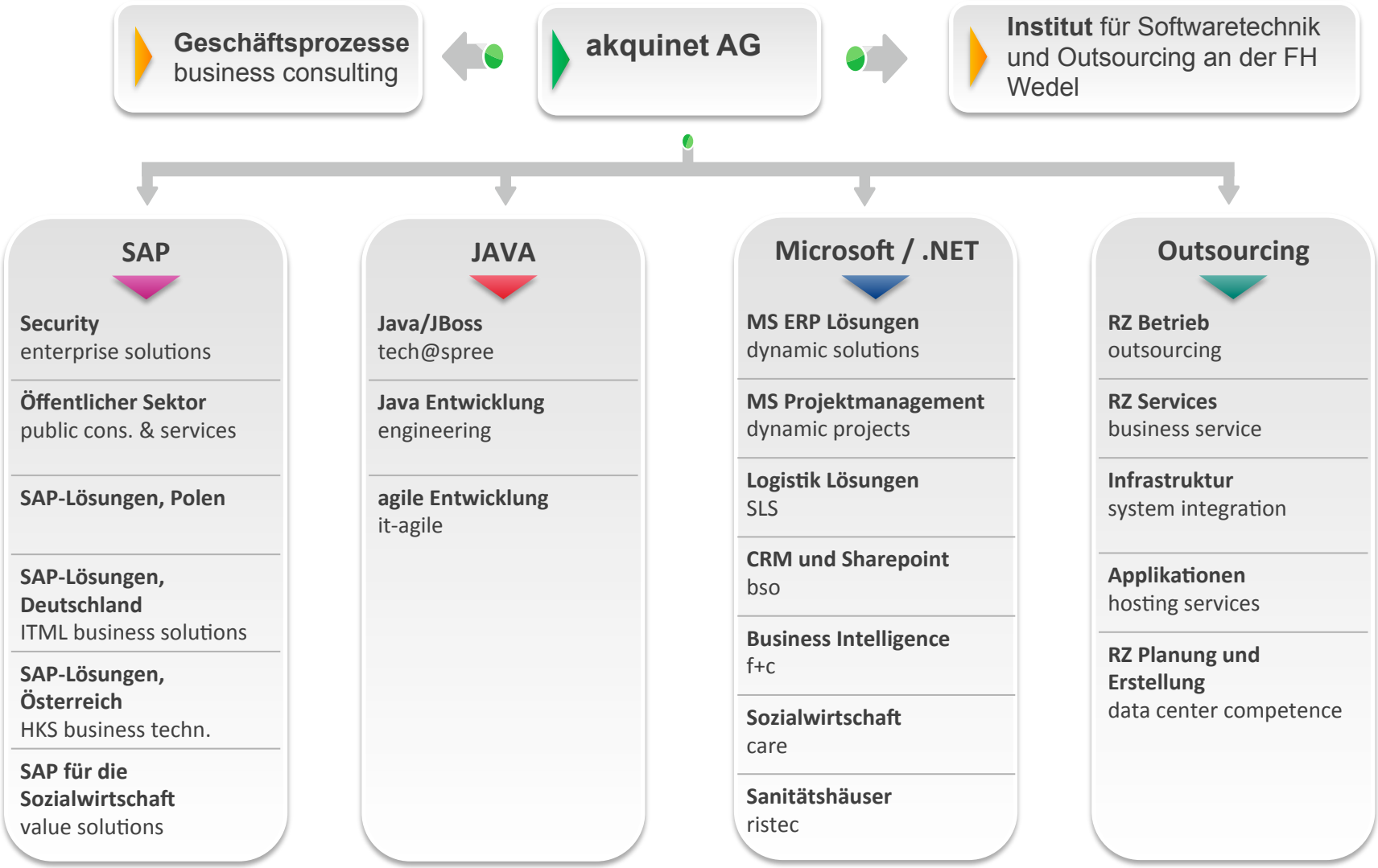
OptaPlanner hilft bei verteilten Schulstandorten

Dr. Torsten Fink – torsten.fink@akquinet.de

Markus Müller – markus.mueller@akquinet.de

akquinet

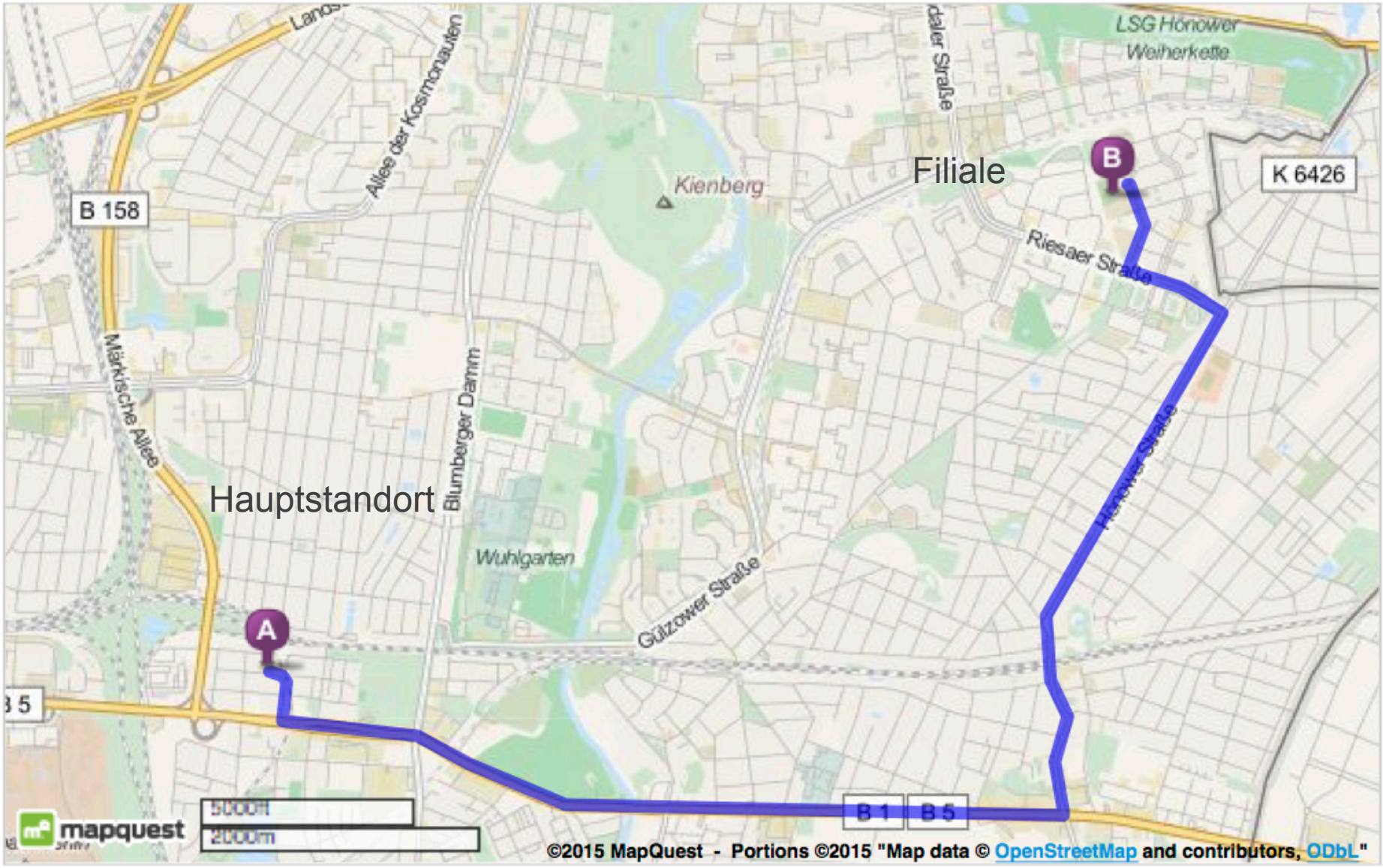
Unternehmensstruktur




harald schröder
DIPL.-ING. ARCHITEKT
Bremen Berlin
Otto-Nagel-Gymnasium
Perspektive Hofseite


OHV ST 545

Geschätzte Gesamt-Fahrtzeit: **8.79 Kilometer - etwa 14 Minuten**



Der Schultag mit seinen Pausen

1	08:00-08:45
2	08:45-09:30
Pause	20 min
3	09:50-10:35
4	10:35-11:20
Pause	10 min
5	11:30-12:15
6	12:15-13:00
Pause	30 min
7	13:30-14:15
8	14:15-15:00
Pause	5 min
9	15:05-15:50
10	15:50-16:35

Bisher etwa **70 Standortwechsel pro Woche**:

Wie lässt sich die Zufriedenheit aller Beteiligten wieder verbessern?

Ein Stundenplan der weniger Standortwechsel erfordert..

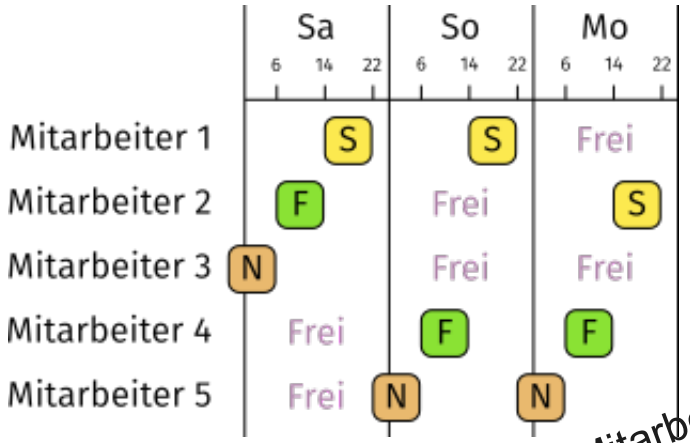
Jede Woche etwa..

- 705 Schüler
- 21 Klassen
- 72 Lehrer
- 1278 unterrichtete Schulstunden
- 50 mögliche Zeiträumen

Klassisches Optimierungsproblem

1. Zielfunktion
   - Gewinn maximieren
   - Ökologische Auswirkungen minimieren
   - .. Standortwechsel minimieren
2. Begrenzte Ressourcen
   - Betriebsmittel (Maschinen, Budget usw.)
   - Zeit
   - .. Klassenräume
3. Bedingungen
   - Mitarbeiterzufriedenheit vs. Überstunden
   - Wartungszeiten
   - .. Lehrplan

Nurse rostering

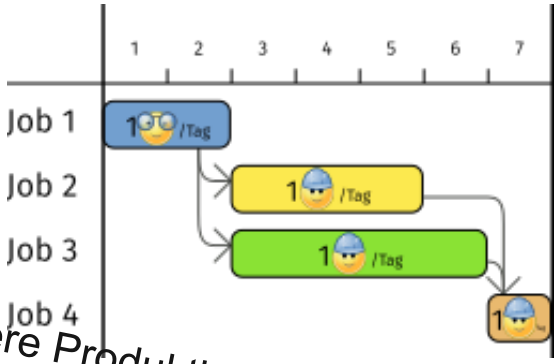


Zufriedene Mitarbeiter

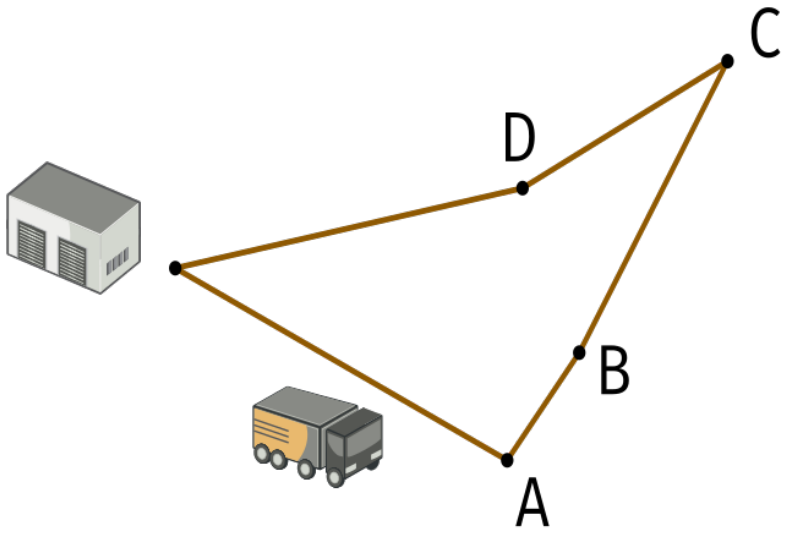
Vehicle routing (TSP)

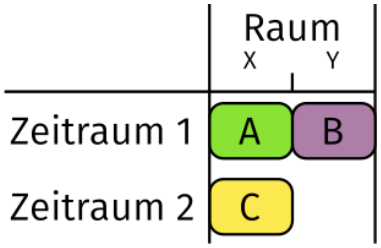
Geringerer Kraftstoffverbrauch

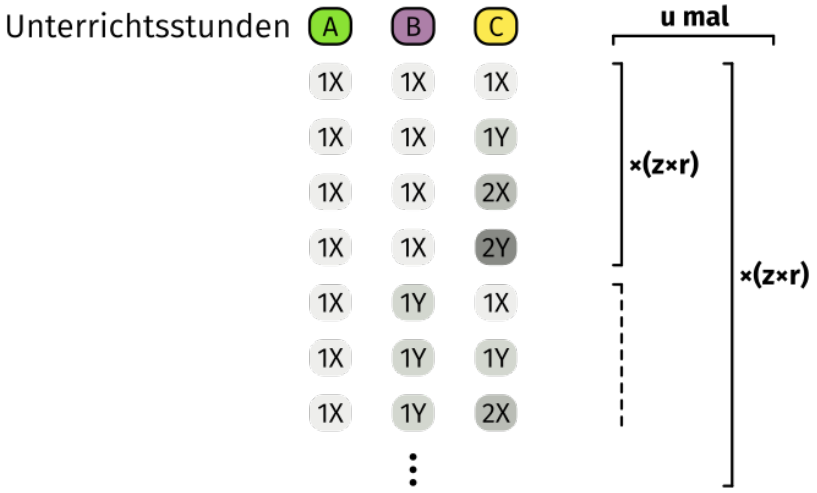
Job shop scheduling



Kürzere Produktionsdauer





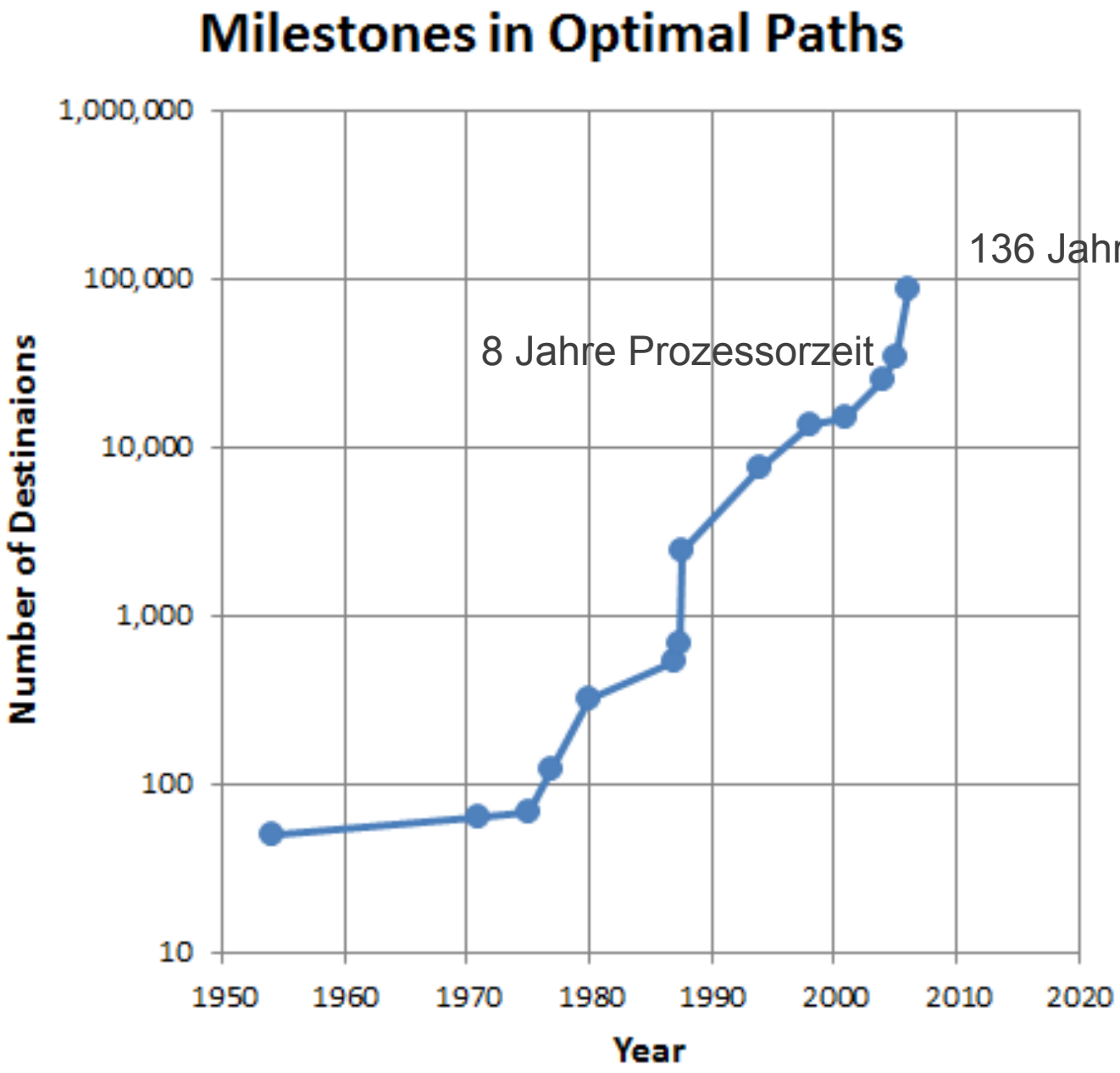


Lösungsraum: **n!**

# Wegpunkte	Lösungsraum
4	24
100	10^{157}
1000	10^{2567}
10000	10^{35659}

Lösungsraum: **$(z \times r)^u$**

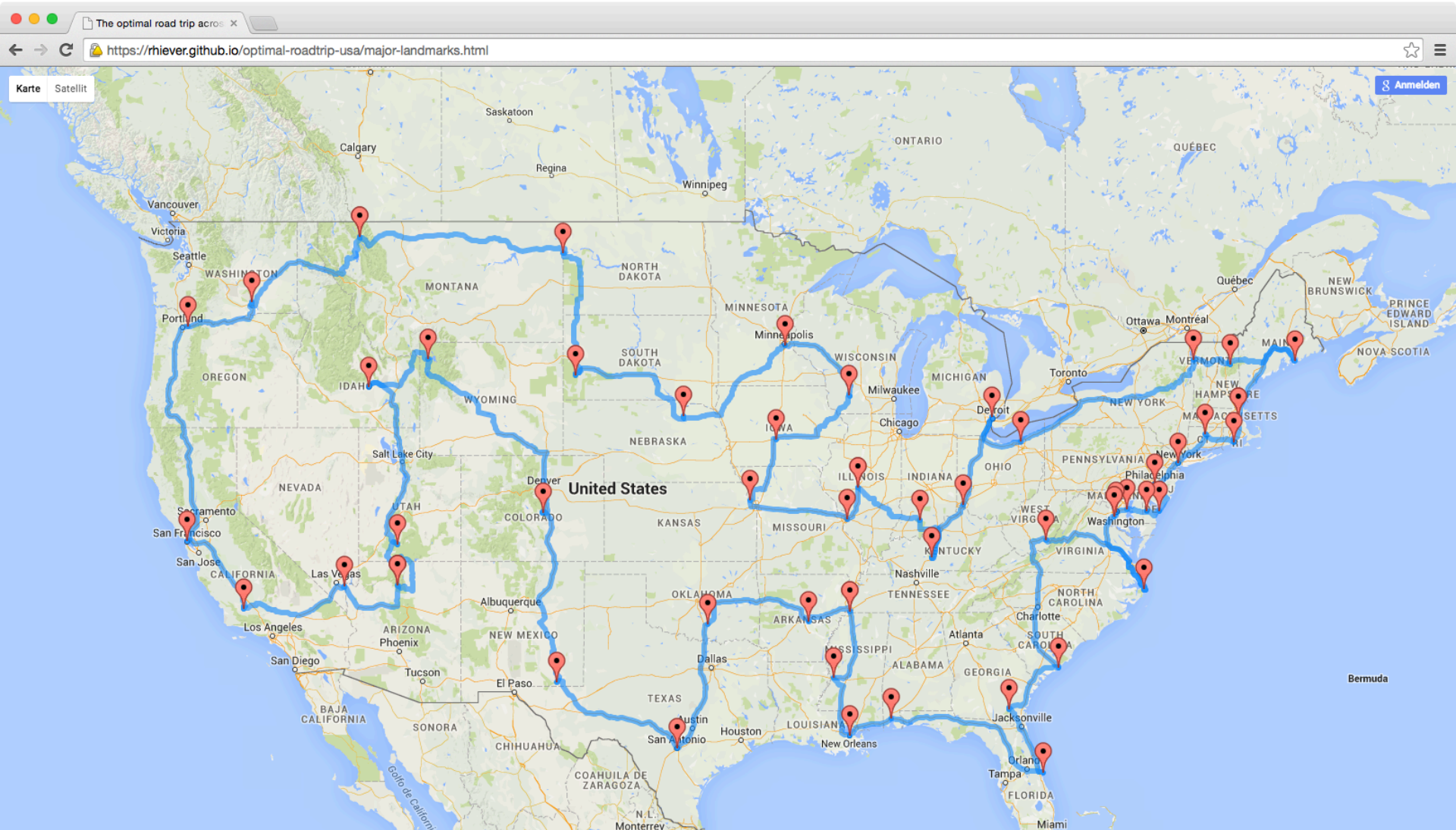
# Zeiträume	# Räume	# Unterrichtsstunden	
2	2	3	64
36	6	100	10^{233}
36	18	400	10^{1124}
36	36	800	10^{2490}



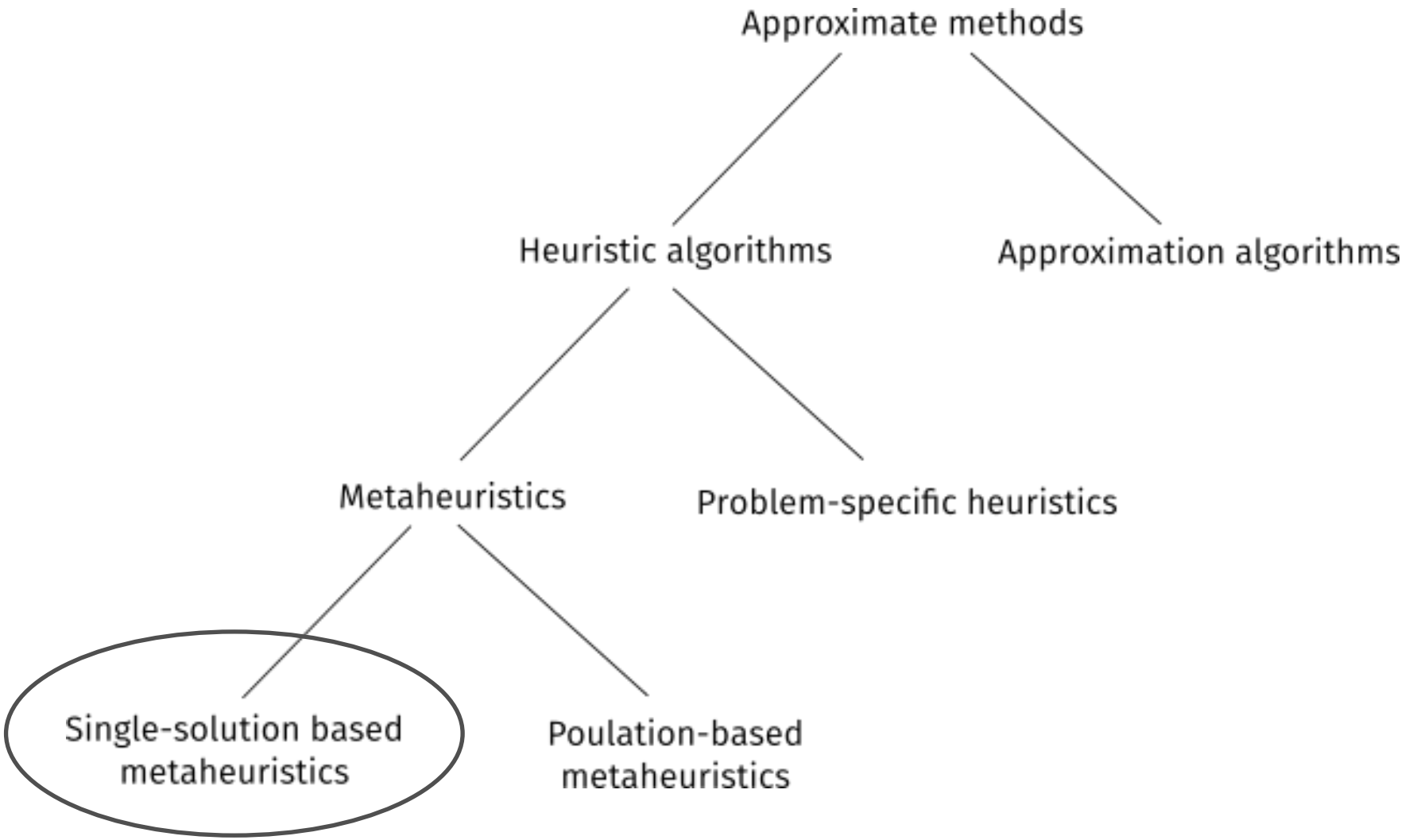
https://blogs.synopsys.com/optimizationcraft/2013/11/11/32/

“The ultimate American road trip revealed:
Data scientist uses algorithms to plot the best route across the United States”

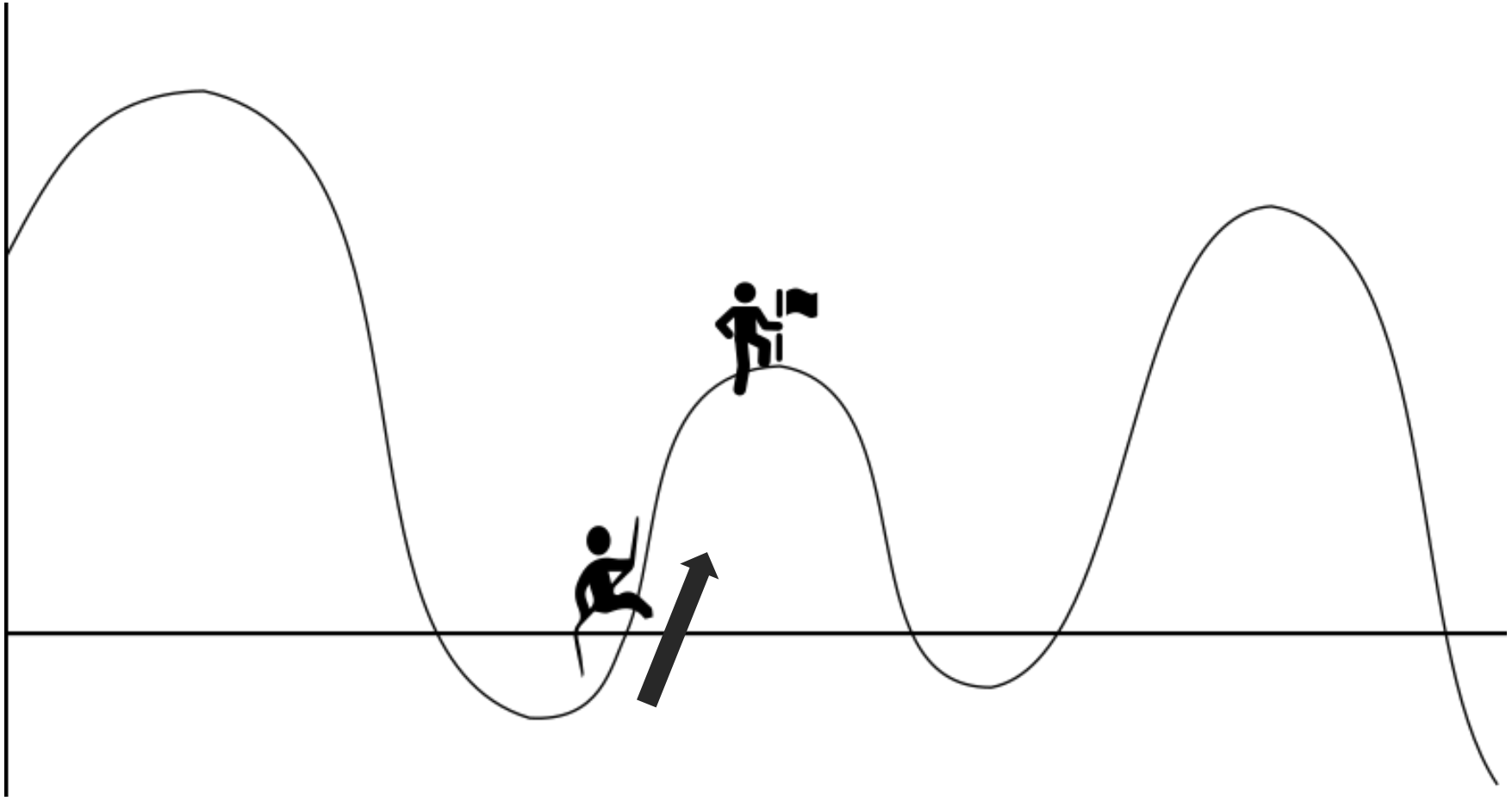
(DailyMail 13. März 2015)



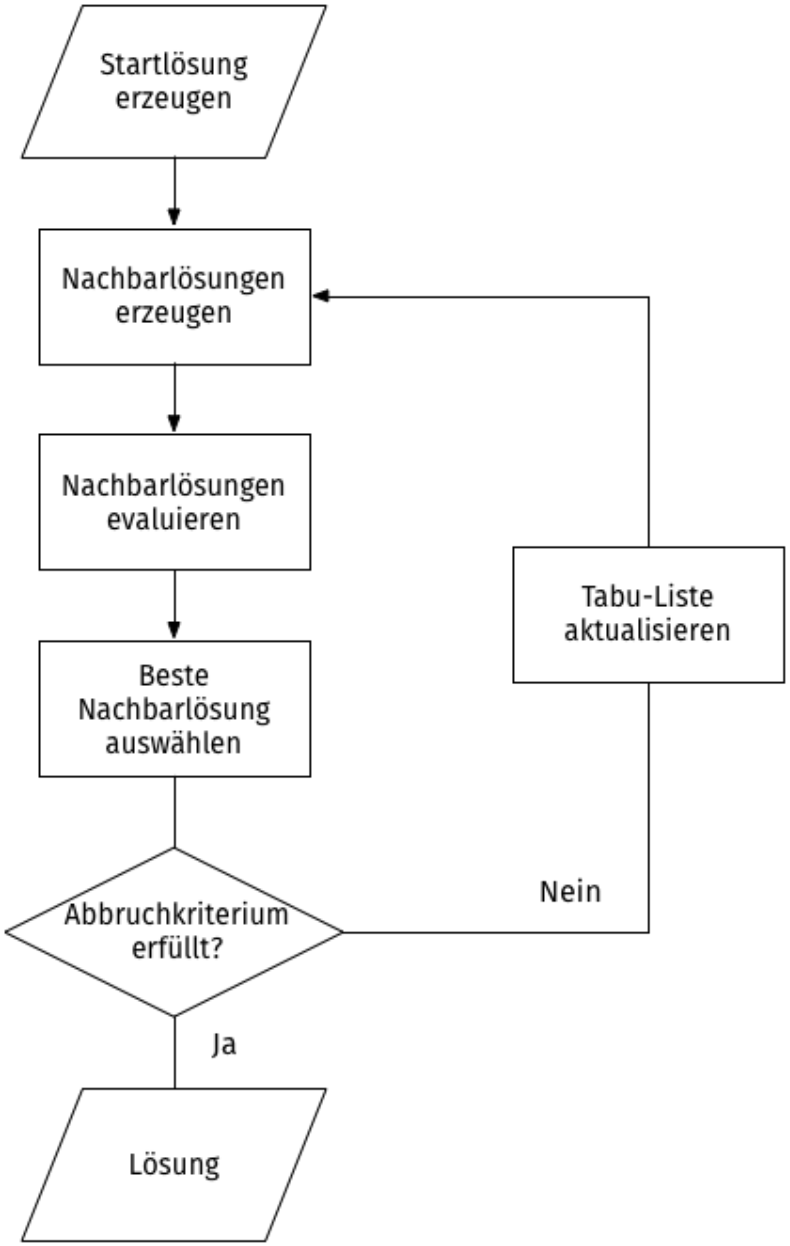
Näherungsweise Lösung von Optimierungsproblemen



Wie funktionieren Lokale Suchverfahren?



Lokale Optima vermeiden: Die Tabu-Suche



Zusammengefasst:

- Durch die Bauarbeiten entstehen Standortwechsel, die den Ablauf des Schulalltages stören.
- Alle Beteiligten sind mit der Situation unzufrieden.
- Die Software, mit der die Stundenpläne erstellt werden, kann die Standortwechsel nicht bei der Erstellung der Pläne berücksichtigen.
- Es handelt sich um ein Optimierungsproblem, welches nicht effizient und exakt zu lösen ist.
- Es existieren Lösungsverfahren zur näherungsweisen Ermittlung eines optimalen Stundenplanes mit weniger Standortwechseln.

Unsere Lösung mit…

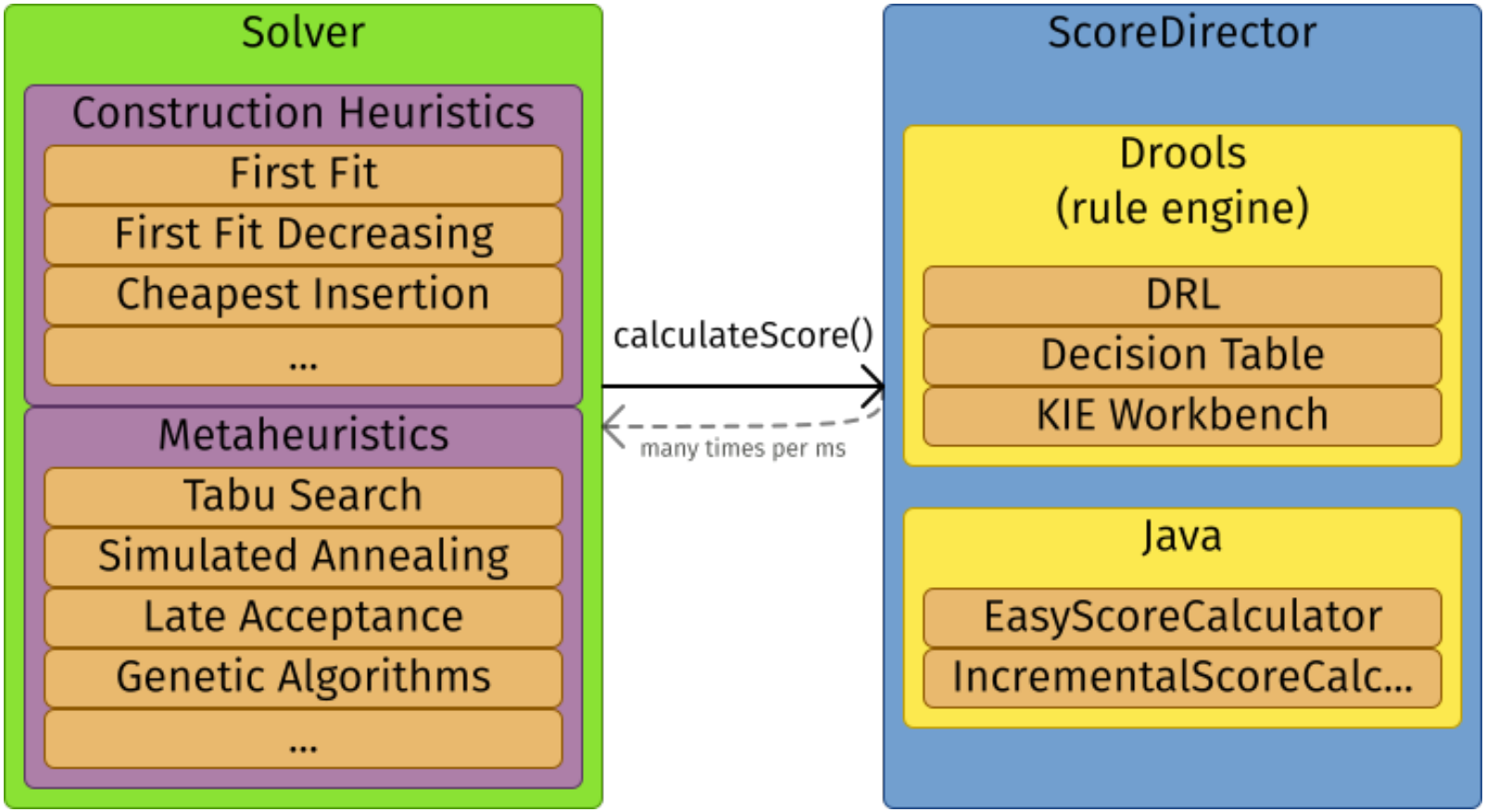
OptaPlanner 

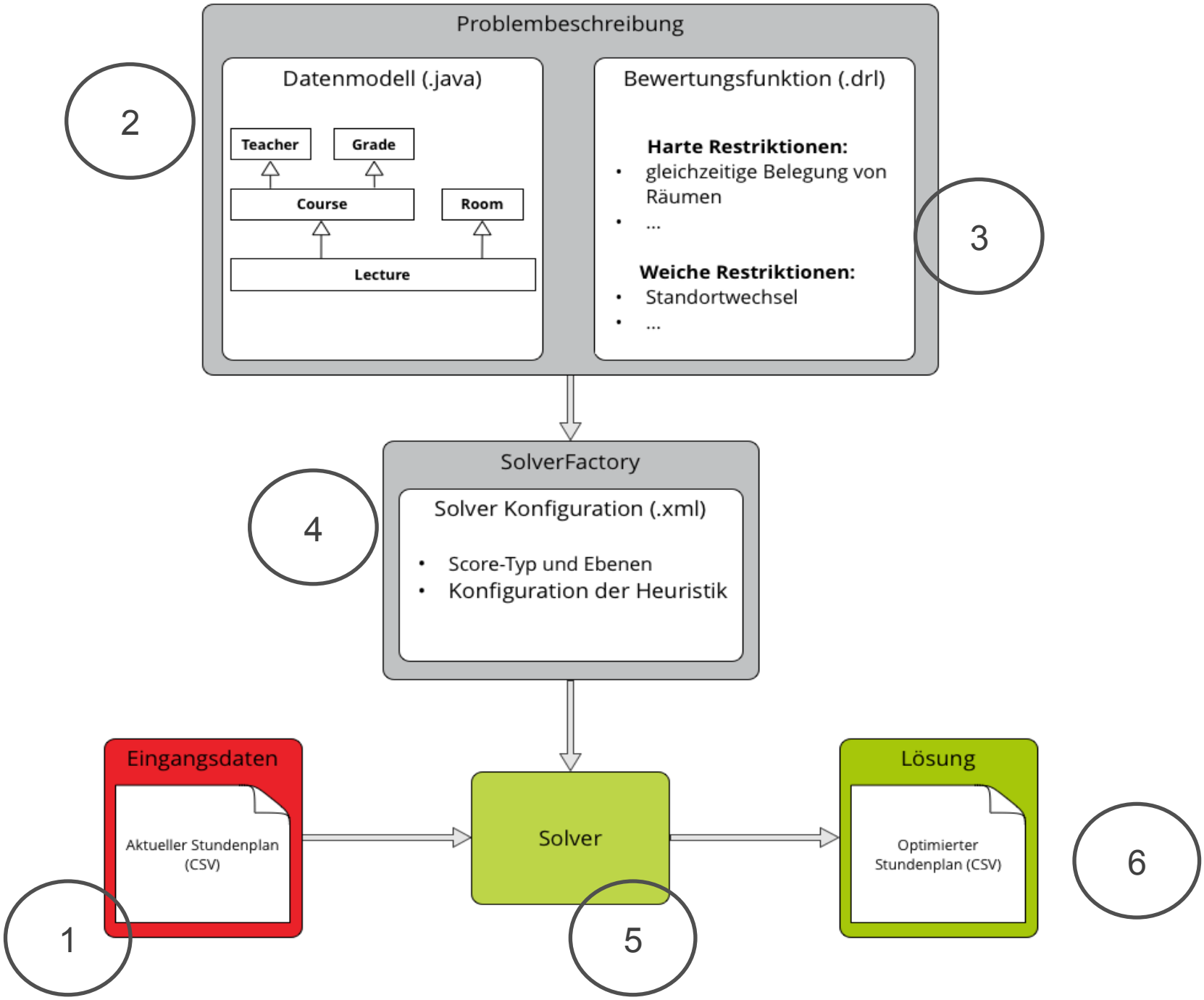
... eine Java Planning Engine

Architekturüberblick

Lösungskandidaten finden

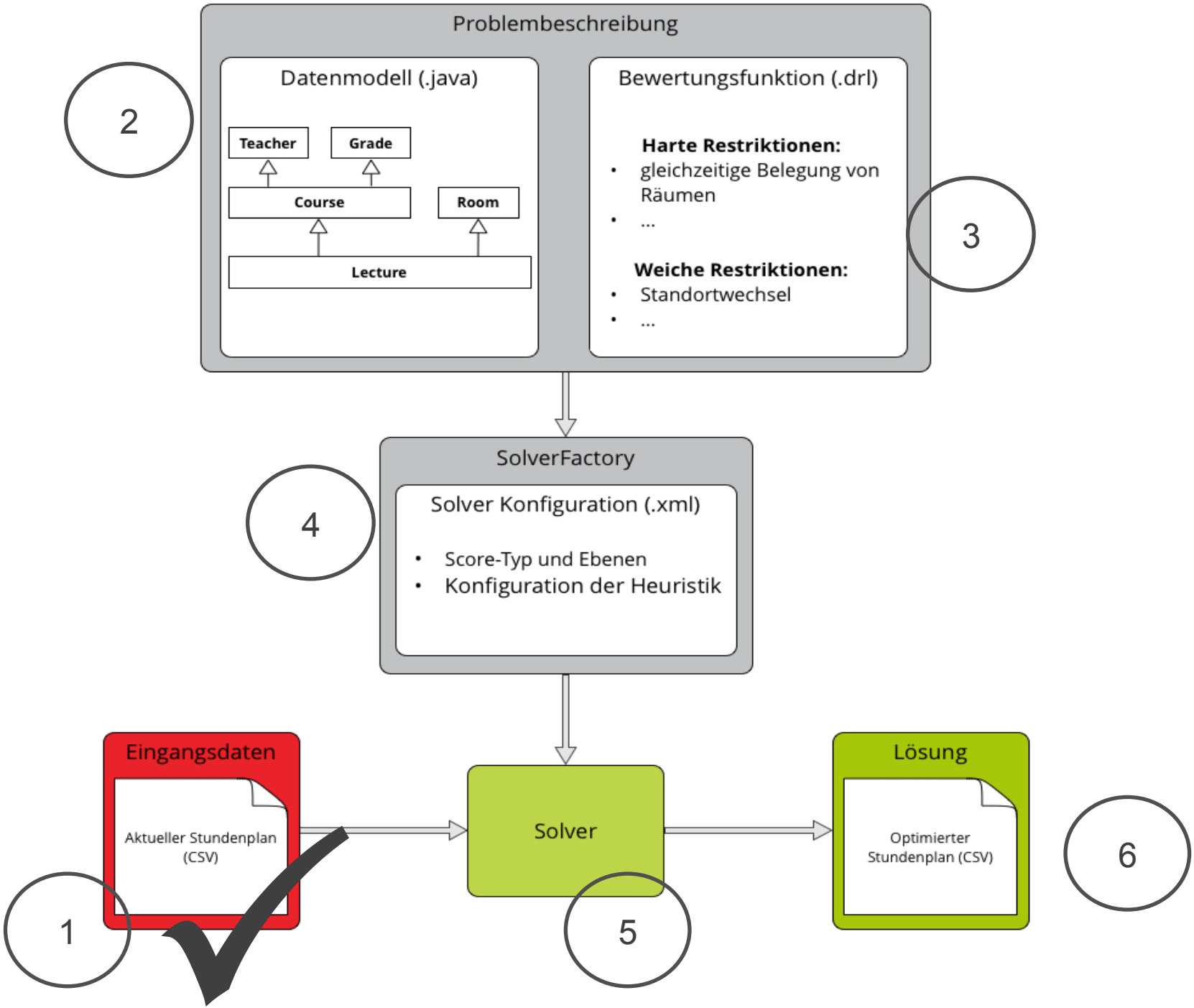
Lösungskandidaten bewerten



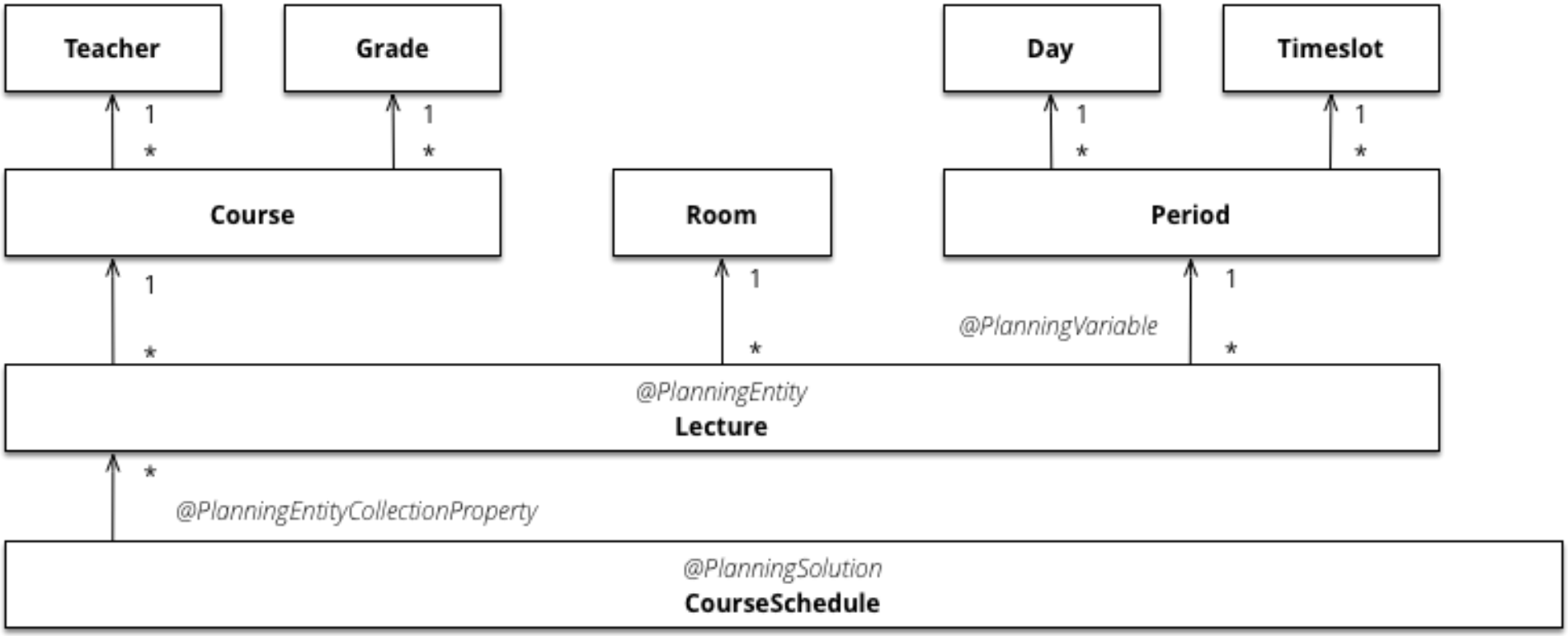

Problembeschreibung
Datenmodell (.java)
Teacher
Grade
Course
Room
Lecture
Bewertungsfunktion (.drl)
Harte Restriktionen:
• gleichzeitige Belegung von Räumen
• ...
Weiche Restriktionen:
• Standortwechsel
• ...
2
3
SolverFactory
Solver Konfiguration (.xml)
• Score-Typ und Ebenen
• Konfiguration der Heuristik
4
Eingangsdaten
Aktueller Stundenplan (CSV)
Solver
Lösung
Optimierter Stundenplan (CSV)
1
5
6

1. Eingangsdaten

```
Arnd    5   5   Ma  F13 177 1   10.3    ----1-----------   0
Arnd    5   6   Ma  F13 177 1   10.3    ----1-----------   0
Bau     1   7   Ek  F24 154 0   9.3     ----1-----------   0
Bau     1   8   Ek  F24 154 0   9.3     ----1-----------   0
```


Problembeschreibung
Datenmodell (.java)
Teacher
Grade
Course
Room
Lecture
Bewertungsfunktion (.drl)
Harte Restriktionen:
• gleichzeitige Belegung von Räumen
• ...
Weiche Restriktionen:
• Standortwechsel
• ...
2
3
SolverFactory
Solver Konfiguration (.xml)
• Score-Typ und Ebenen
• Konfiguration der Heuristik
4
Eingangsdaten
Aktueller Stundenplan (CSV)
Solver
Lösung
Optimierter Stundenplan (CSV)
1
5
6

2. Modellierung des Problems



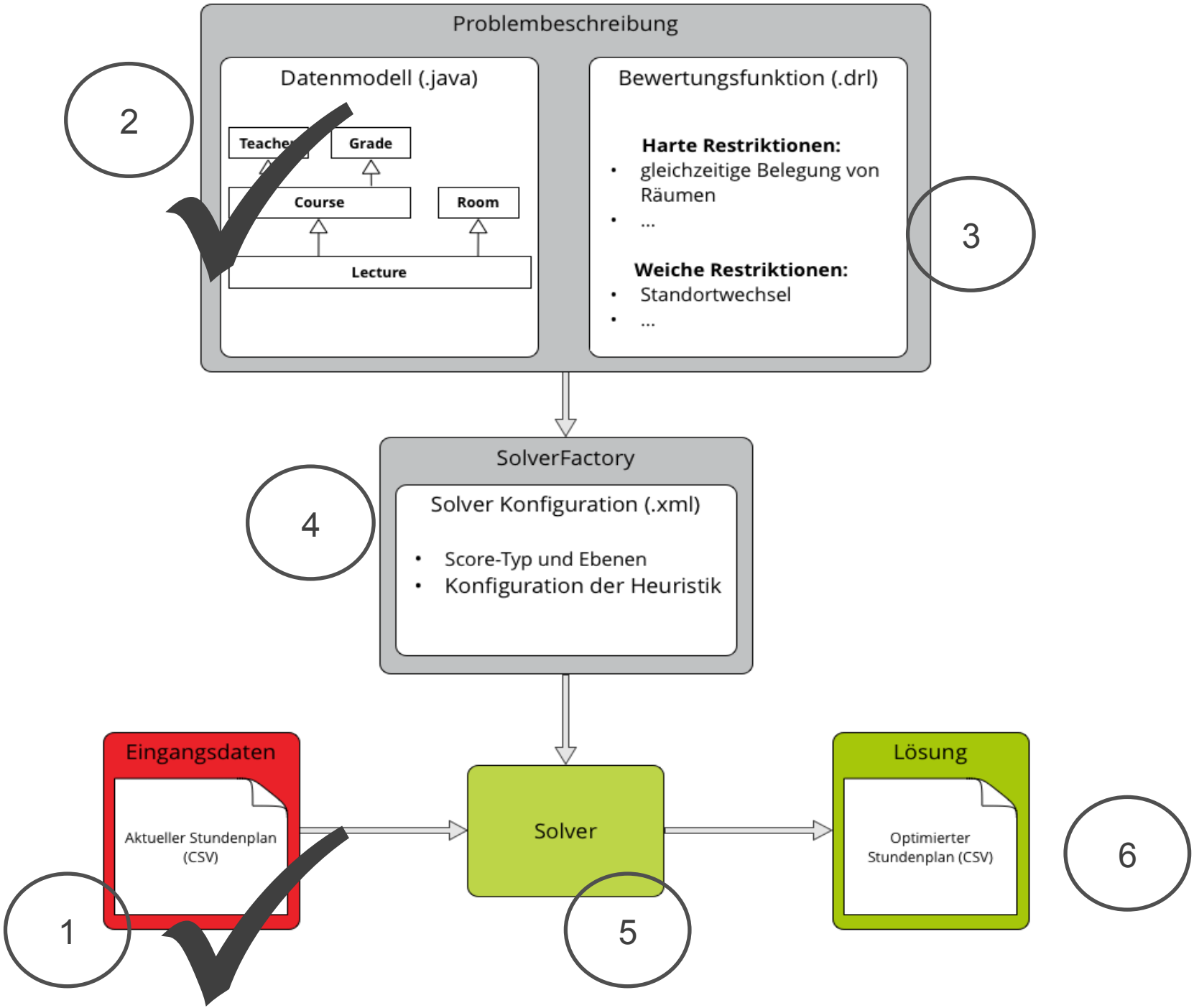
2. Modellierung des Problems

`@PlanningEntity`	Die Klasse, deren Objekte während der Optimierung verändert werden. In diesem Beispiel die `Lecture`.
`@PlanningVariable`	Die Eigenschaft der Klasse, die verändert wird. In diesem Beispiel die `Period`.
`@PlanningSolution`	Eine Klasse, die alle Daten zur Planung enthält, der `CourseSchedule`.

Drei Hauptelemente (Annotationen)

```
@PlanningEntity
public class Lecture {

private Course course;
private Period period;

private ArrayList subsequentLectures = new ArrayList<>();
private ArrayList coupledLectures = new ArrayList<>();

@PlanningVariable(valueRangeProviderRefs = {"periodRange"})
public Period getPeriod() {
return period;
}
}
```

```
@PlanningSolution
public class CourseSchedule implements Solution<BendableScore>
{
    private List<Teacher> teachers;
    private List<Course> courses;

    private List<Day> days;
    private List<Timeslot> timeslots;
    private List<Period> periods;

    private List<Room> rooms;

    private List<Lecture> lectures;

    @ValueRangeProvider(id = "periodRange")
    public List<Period> getPeriods() {
        return periods;
    }

    @PlanningEntityCollectionProperty
    public List<Lecture> getLectures() {
        return lectures;
    }
```

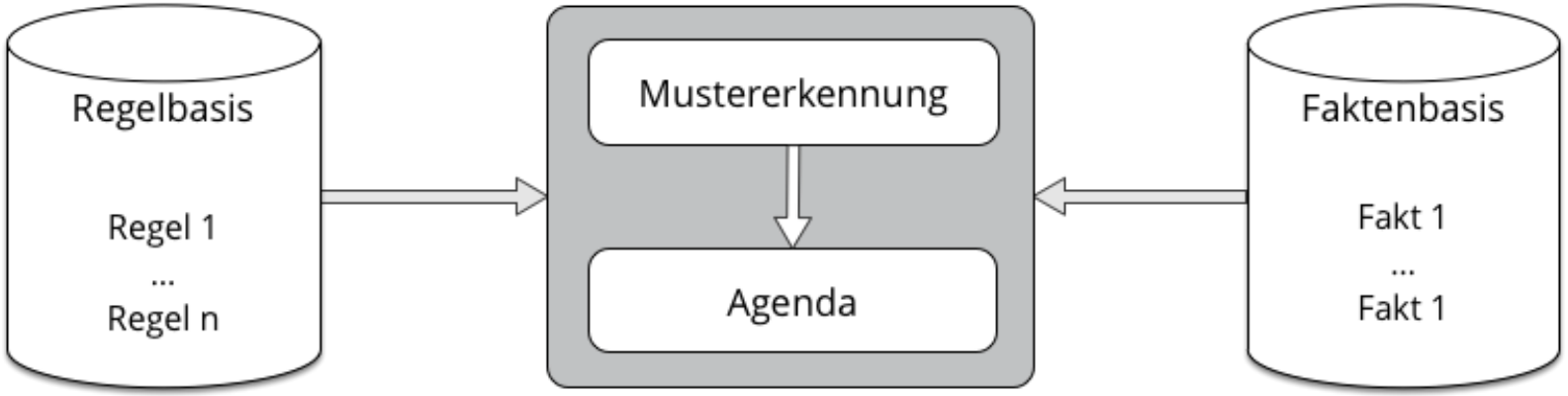

Problembeschreibung
Datenmodell (.java)
Teacher
Grade
Course
Room
Lecture
Bewertungsfunktion (.drl)
Harte Restriktionen:
• gleichzeitige Belegung von Räumen
• ...
Weiche Restriktionen:
• Standortwechsel
• ...
2
3
SolverFactory
Solver Konfiguration (.xml)
• Score-Typ und Ebenen
• Konfiguration der Heuristik
4
Eingangsdaten
Aktueller Stundenplan (CSV)
Solver
Lösung
Optimierter Stundenplan (CSV)
1
5
6

3. Bedingungen und Bewertungsfunktion

- Offensichtliche Bedingungen wie:
  - Ein Lehrer kann nicht an zwei Orten zur gleichen Zeit sein.
  - Ein Raum kann nicht von mehreren Klassen zur gleichen Zeit benutzt werden.

- Aber auch (nicht so offensichtliche) Bedingungen, die im Dialog mit dem Inhaber des Problems ermittelt werden müssen:
  - Zeitliche Vorlieben einzelnen Lehrer
  - Kurssystem der Oberstufe, „Was kann parallel stattfinden, was nicht?“

3. Bedingungen und Bewertungsfunktion



- Bedingungen in Form von Drools Regeln
- Es gibt harte und weiche Bedingungen
- Weiche Bedingungen stellen das Optimierungskriterium dar

3. Bedingungen und Bewertungsfunktion

- Um verschiedene Lösungen vergleiche zu können muss für eine Lösung (Solution) ein Score-Wert angeben werden können.
- In unserem Fall hat der Score-Wert sechs Ebenen:

	Standortwechsel in Pause mit:				
Harte Restriktionen	5 min	10 min	20 min	30 min	> 30min

Bsp.: `0/0/-15/-14/-8/-33`

3. Bedingungen und Bewertungsfunktion

```
rule "roomOccupancy"
    when
        $leftLecture : Lecture($leftId : id, period != null,
                               $period : period,
                               course.room != null,
                               $leftRoom : course.room)

        $rightLecture : Lecture(period == $period,
                                course.room == $leftRoom,
                                id > $leftId)
    then
        scoreHolder.addHardConstraintMatch(kcontext,0 ,-1);
end
```

Harte Bedingung - „Gleichzeitige Belegung eines Raumes“

3. Bedingungen und Bewertungsfunktion

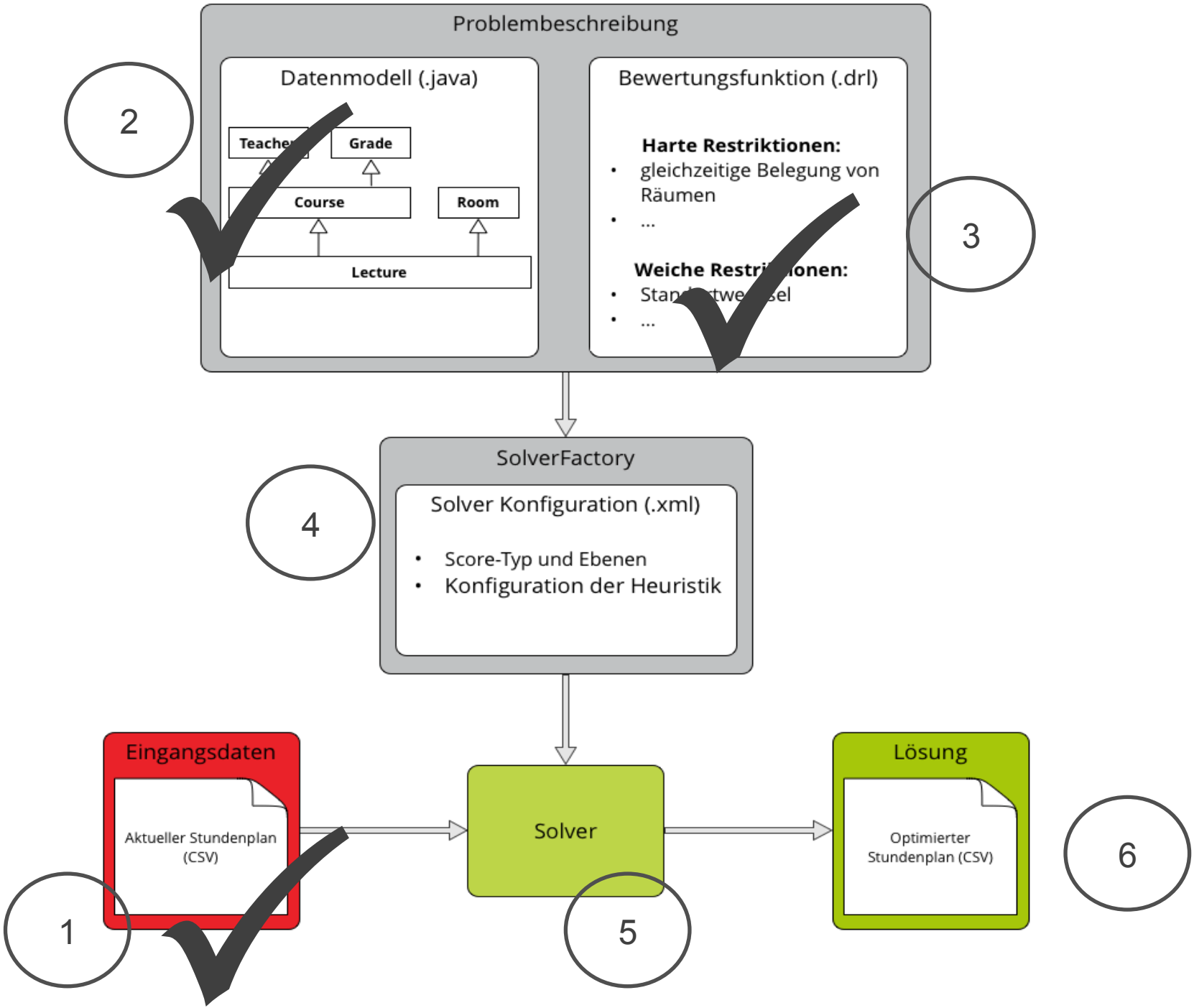
```
rule "teacherChangesBuildinginBreak"
    when
        //Lehrer wechselt Standort in einer Pause
    then
end

rule "teacherChangesBuildingin30minBreak" extends
"teacherChangesBuildinginBreak"
    when
        eval($leftTimeslotIndex == 5)
    then
        scoreHolder.addSoftConstraintMatch(kcontext,3, -2);
end
```

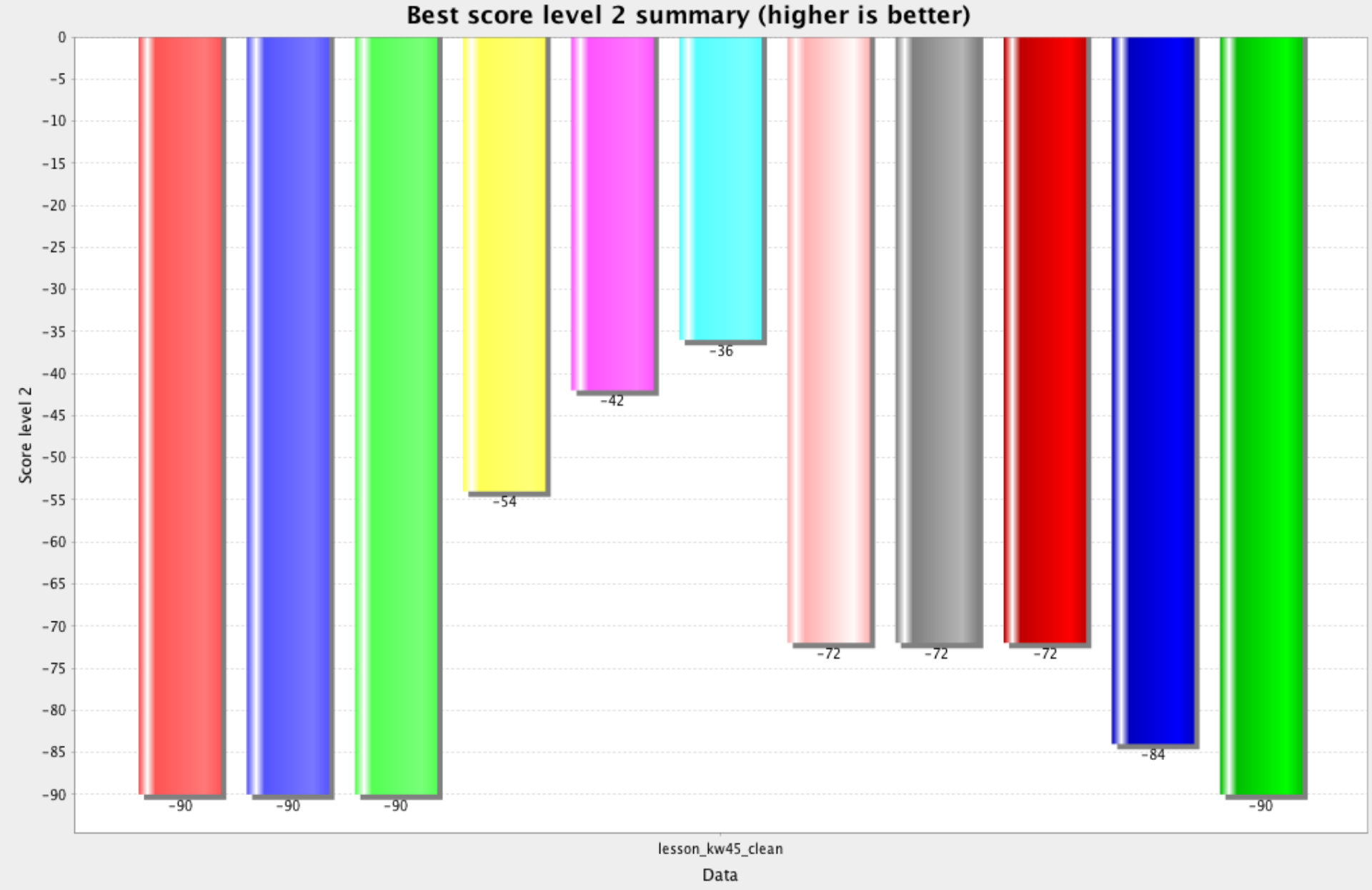
Weiche Bedingung - „Gebäudewechsel“

4. Konfiguration und Auswahl des Lösungsverfahrens

```
<solver>
    <solutionClass>CourseSchedule</solutionClass>
    <entityClass>Lecture</entityClass>

    <scoreDirectorFactory>
        <scoreDefinitionType>BENDABLE</scoreDefinitionType>
        <bendableHardLevelsSize>1</bendableHardLevelsSize>
        <bendableSoftLevelsSize>5</bendableSoftLevelsSize>
        <scoreDrl>scoreRules.drl</scoreDrl>
    </scoreDirectorFactory>

    <--!<constructionHeuristic></constructionHeuristic>-->

    <localSearch> .. </localSearch>
</solver>
```

Konfiguration des Solvers


Problembeschreibung
Datenmodell (.java)
Teacher
Grade
Course
Room
Lecture
Bewertungsfunktion (.drl)
Harte Restriktionen:
• gleichzeitige Belegung von Räumen
• ...
Weiche Restriktionen:
• Standortwechsel
• ...
2
3
SolverFactory
Solver Konfiguration (.xml)
• Score-Typ und Ebenen
• Konfiguration der Heuristik
4
Eingangsdaten
Aktueller Stundenplan (CSV)
Solver
Lösung
Optimierter Stundenplan (CSV)
1
5
6


Best score level 2 summary (higher is better)
Score level 2
0
-5
-10
-15
-20
-25
-30
-35
-40
-45
-50
-55
-60
-65
-70
-75
-80
-85
-90
-90
-90
-90
-54
-42
-36
-72
-72
-72
-84
-90
lesson_kw45_clean
Data
Exhaustive Search
Move tabu
Tabu Swap Pillar Swap and Lecture Change
Tabu Late Acceptance 500 Swap Lecture Change
Tabu Late Acceptance 900 Swap Lecture Change
Tabu Late Acceptance 1800 Swap Lecture Change (favorite)
Tabu Late Acceptance 500 Swap Pillar Swap and Lecture Change
Tabu Late Acceptance 100 Swap Pillar Swap and Lecture Change
Tabu Late Acceptance 900 Swap Pillar Swap and Lecture Change
Simulated Annealing Swap Pillar Swap and Lecture Change
Hill Climbing Swap Pillar Swap and Lecture Change

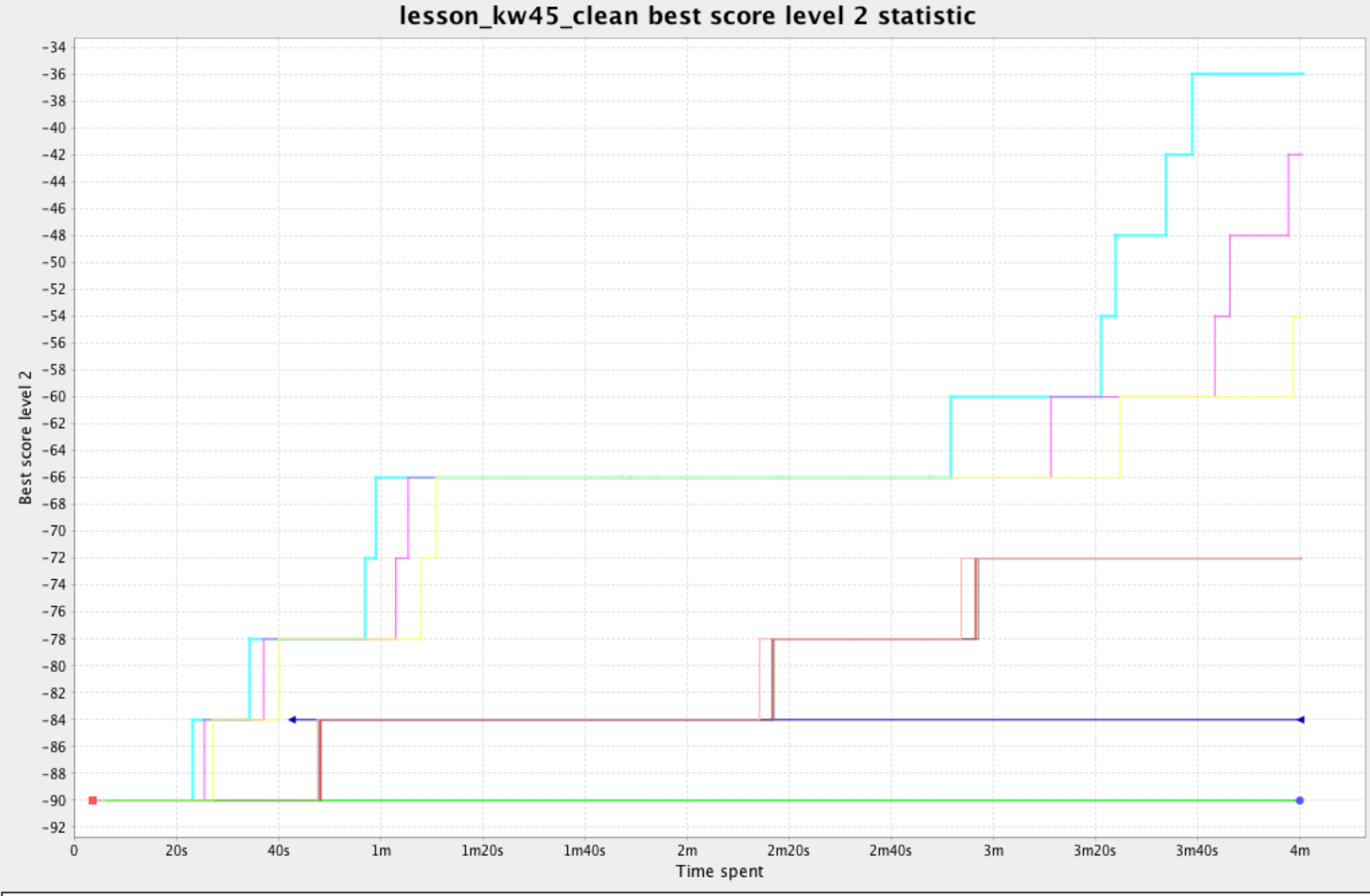
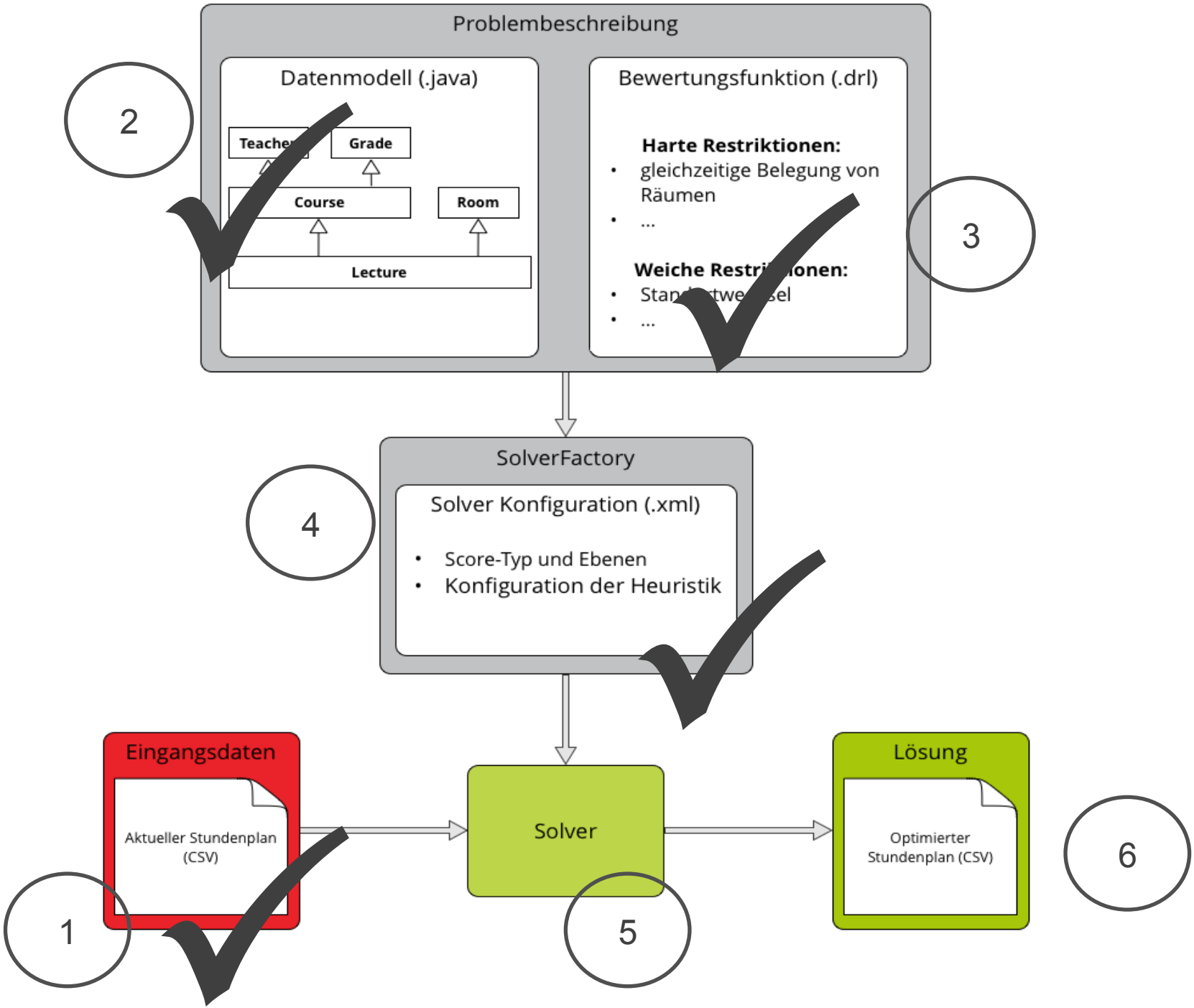
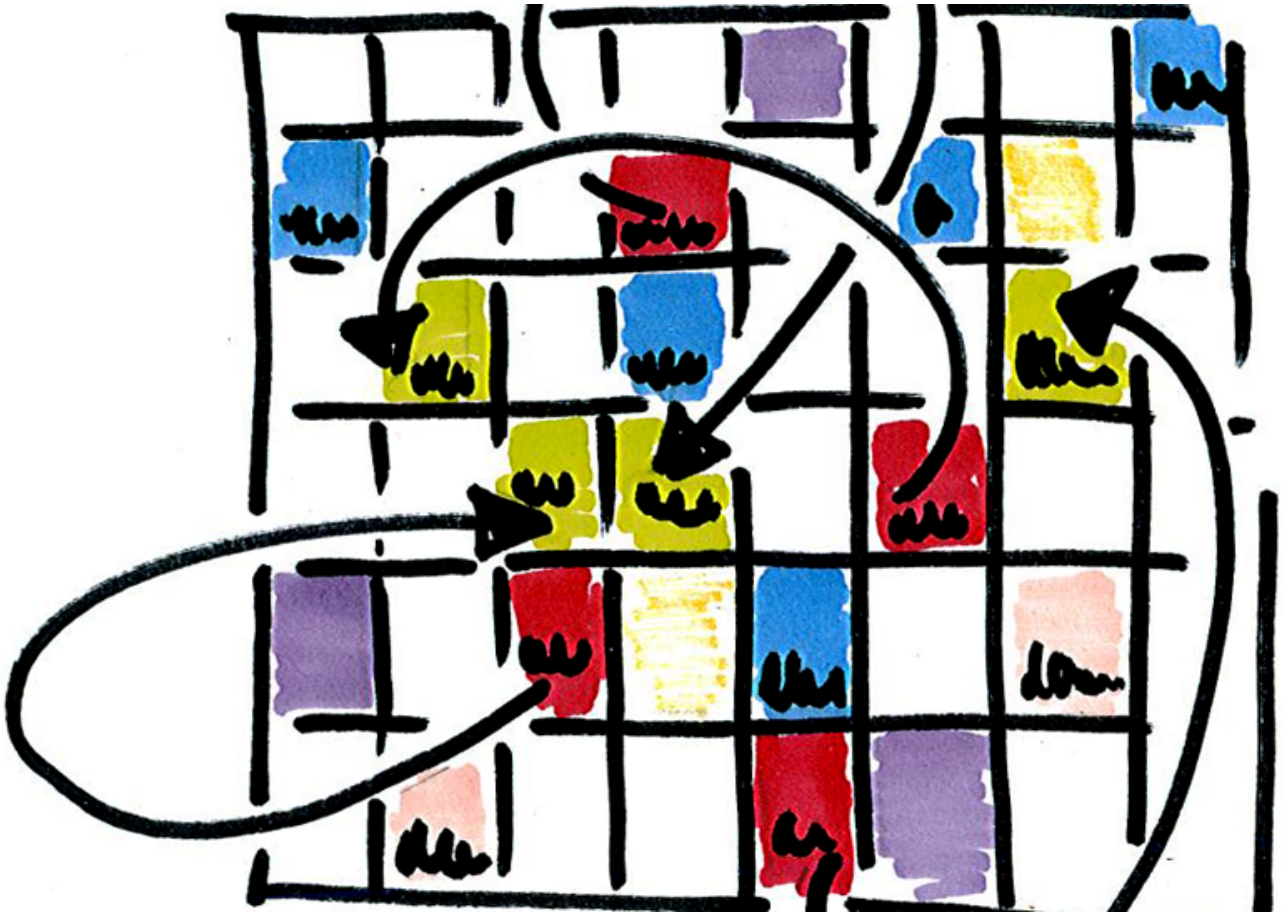

lesson_kw45_clean best score level 2 statistic
Best score level 2
-34
-36
-38
-40
-42
-44
-46
-48
-50
-52
-54
-56
-58
-60
-62
-64
-66
-68
-70
-72
-74
-76
-78
-80
-82
-84
-86
-88
-90
-92
0
20s
40s
1m
1m20s
1m40s
2m
2m20s
2m40s
3m
3m20s
3m40s
4m
Time spent
Exhaustive Search
Move tabu
Tabu Swap Pillar Swap and Lecture Change
Tabu Late Acceptance 500 Swap Lecture Change
Tabu Late Acceptance 900 Swap Lecture Change
Tabu Late Acceptance 1800 Swap Lecture Change (favorite)
Tabu Late Acceptance 500 Swap Pillar Swap and Lecture Change
Tabu Late Acceptance 100 Swap Pillar Swap and Lecture Change
Tabu Late Acceptance 900 Swap Pillar Swap and Lecture Change
Simulated Annealing Swap Pillar Swap and Lecture Change
Hill Climbing Swap Pillar Swap and Lecture Change

Table 3.1. Examples overview

Example	Domain	Size	Competition?	Special features used
N queens	» 1 entity class » 1 variable	» Entity <= 256 » Value <= 256 » Search space <= 10^616	» Pointless (cheatable)	None
Cloud balancing	» 1 entity class » 1 variable	» Entity <= 2400 » Value <= 800 » Search space <= 10^6967	» No » Defined by us	» Real-time planning
Traveling salesman	» 1 entity class » 1 chained variable	» Entity <= 980 » Value <= 980 » Search space <= 10^2927	» Unrealistic » TSP web	» Real-time planning
Dinner party	» 1 entity class » 1 variable	» Entity <= 144 » Value <= 72 » Search space <= 10^310	» Unrealistic	» Decision Table spreadsheet for score constraints
Tennis club	» 1 entity class	» Entity <= 72 » Value <= 7	» No	» Fairness score


Problembeschreibung
Datenmodell (.java)
Teacher
Grade
Course
Room
Lecture
Bewertungsfunktion (.drl)
Harte Restriktionen:
• gleichzeitige Belegung von Räumen
• ...
Weiche Restriktionen:
• Standortwechsel
• ...
2
3
SolverFactory
Solver Konfiguration (.xml)
• Score-Typ und Ebenen
• Konfiguration der Heuristik
4
Eingangsdaten
Aktueller Stundenplan (CSV)
Solver
Lösung
Optimierter Stundenplan (CSV)
1
5
6

5. Ablauf der Optimierung



Solver starten:

```
Solution s = scheduleSolutionBuilder.extractDomain();

SolverFactory solverFactory = SolverFactory
                            .createFromXmlResource(SOLVER_CONFIG)
;

final Solver solver = solverFactory.buildSolver();

solver.addEventListener(event ->…);

solver.solve(s);
```

DefaultSolver - Solving started: time spent (1352), best score (0/0/-90/-42/-16/-34), **environment mode (REPRODUCIBLE),** random (JDK with seed 0).

LocalSearchDecider - Move index (4), **score (-66/0/-90/-51/-16/-32), accepted (false),** move (LectureChangeMove: setting new starting Period[Period-40]).

LocalSearchDecider - Move index (5), score (-4/0/-90/-42/-16/-34), accepted (false), move (LectureChangeMove: setting new starting Period[Period-2]).

LocalSearchDecider - Move index (6), score (-16/0/-96/-42/-14/-34), accepted (false), move (LectureChangeMove: setting new starting Period[Period-32]).

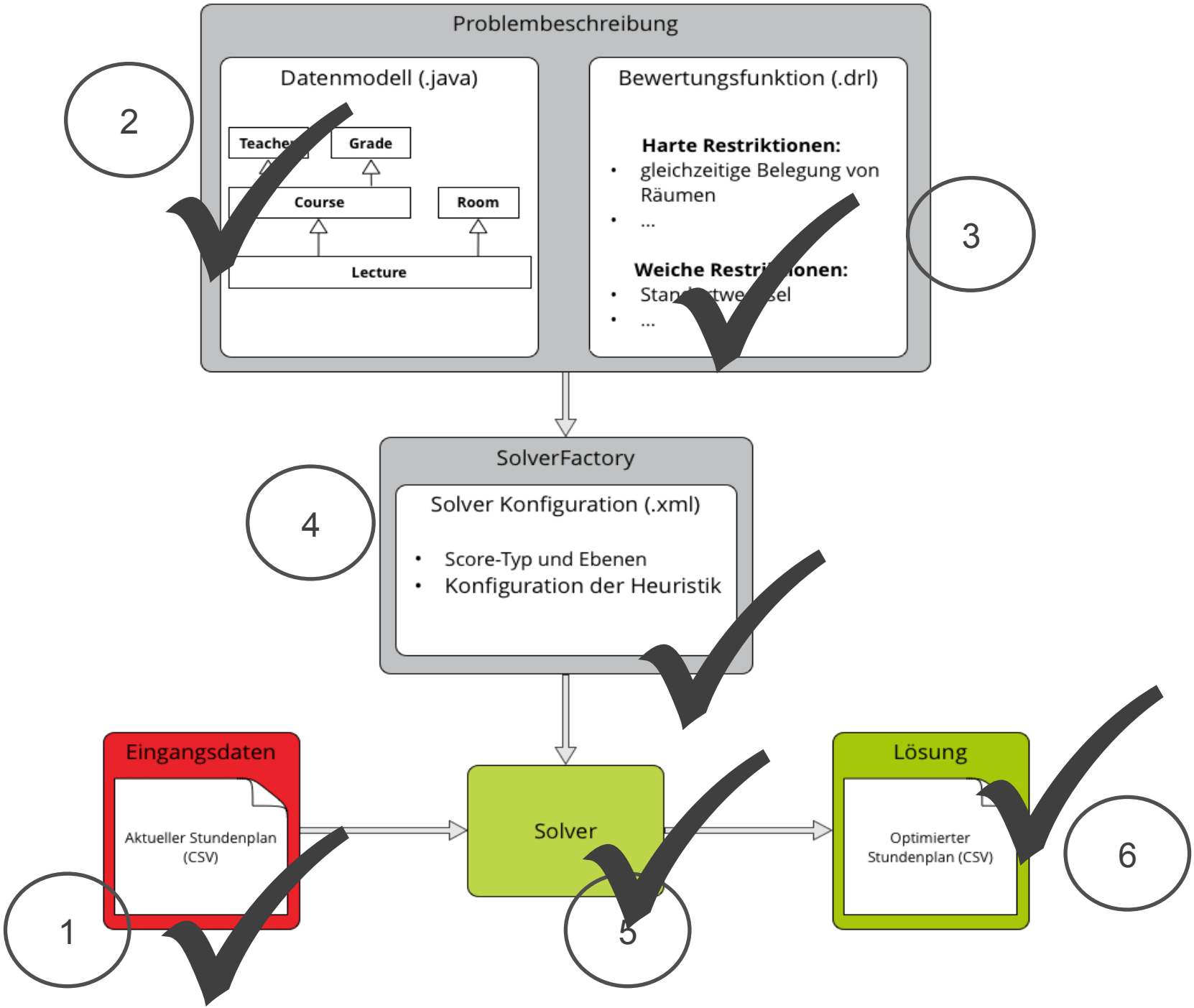
LocalSearchDecider - Move index (7), **score (0/0/-90/-42/-16/-33), accepted (true),** move (LectureChangeMove: setting new starting Period[Period-0]).

Main - **Best Solution changed, new Score: 0/0/-90/-42/-16/-33**

DefaultLocalSearchPhase - LS step (0), time spent (1982), score (0/0/-90/-42/-16/-33), new best score (0/0/-90/-42/-16/-33), accepted/selected move count (4/7), picked move (LectureChangeMove: SubsequentLecturesId 237 setting new starting Period[Period-0]).

6. Ergebnisse

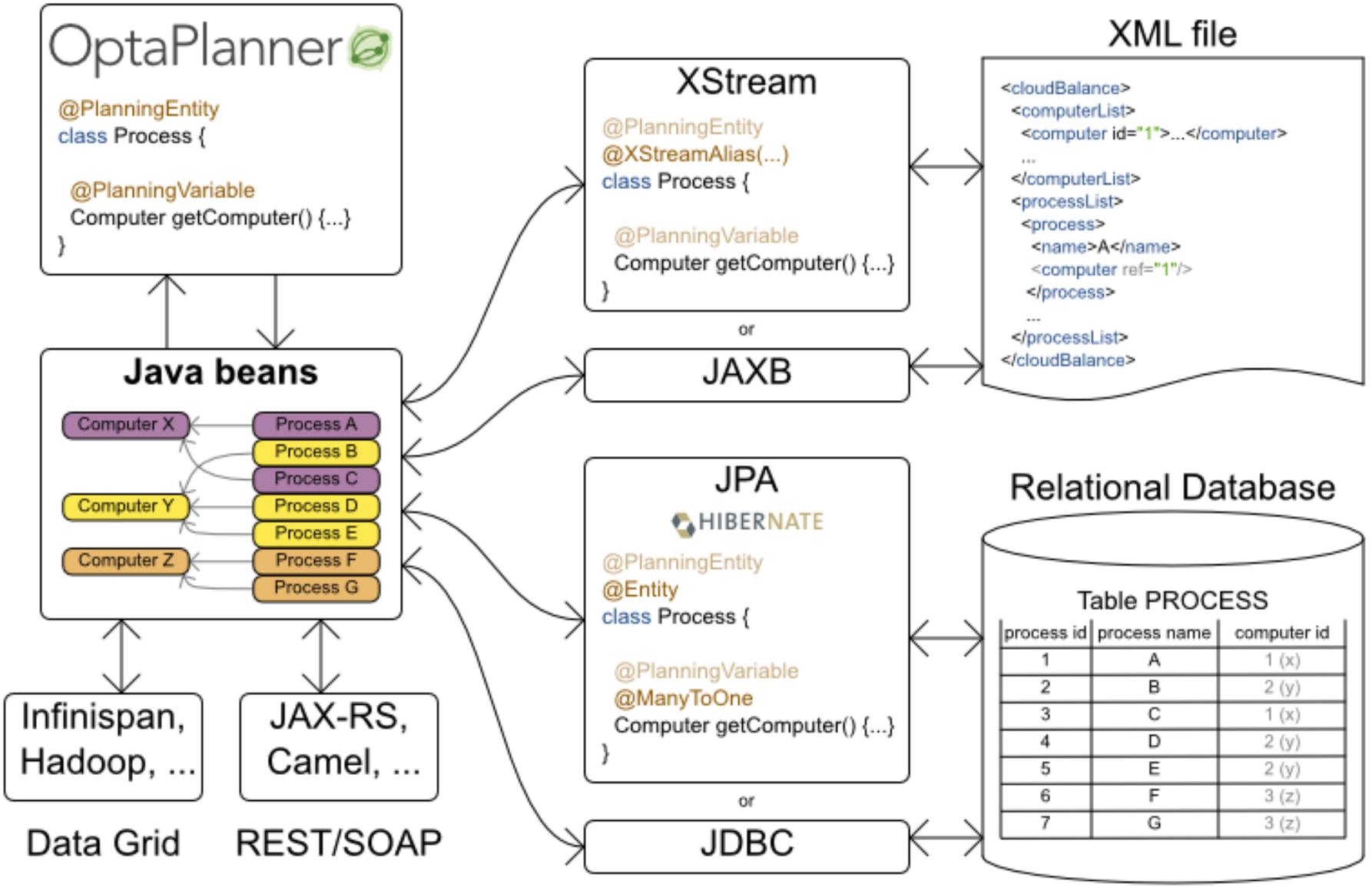
			Standortwechsel in Pause mit:					Summe
	Score-Wert	Verletzung harte Restriktionen	5 min	10 min	20 min	30 min	> 30min	
Ausgangslösung	0/0/-15/-14/-8/-34	0	0	15	14	8	34	71
Lösung 1. Schritt	0/0/-15/-14/-8/-33	0	0	15	14	8	33	70
Lösung n Schritten	0/0/-2/-9/-2/-9	0	0	2	9	2	9	22


Problembeschreibung
Datenmodell (.java)
Teacher
Grade
Course
Room
Lecture
Bewertungsfunktion (.drl)
Harte Restriktionen:
• gleichzeitige Belegung von Räumen
• ...
Weiche Restriktionen:
• Standortwechsel
• ...
2
3
SolverFactory
Solver Konfiguration (.xml)
• Score-Typ und Ebenen
• Konfiguration der Heuristik
4
Eingangsdaten
Aktueller Stundenplan (CSV)
Solver
Lösung
Optimierter Stundenplan (CSV)
1
5
6

$$
\begin{aligned}
&\textit{minimize } \textstyle\sum_{i=1}^{n} s_i \\
&\textit{subject to} \\
&d_s - r_{ij}(t) \leq 0,\ \forall i, j,\ i \neq j,\ 0 \leq t \leq \max_i (T_{max_i}) \\
&v_i(t) - v_{max_i} \leq 0,\ \ \forall i,\ \ 0 \leq t \leq \max_i (T_{max_i}) \\
&a_i(t) - a_{max_i} \leq 0,\ \ \forall i,\ \ 0 \leq t \leq \max_i (T_{max_i})
\end{aligned}
$$

Integration overview

OptaPlanner combines easily with other Java and JEE technologies.



Erfahrungen aus dem Einsatz von OptaPlanner

- Verschiedene Datensets beschaffen (Größe und Beschaffenheit)
- Frühzeitig eine Möglichkeit zur Visualisierung von Lösungen bereitstellen
- .. Benchmarks schreiben und ausführen
- Beim Entwickeln von Regeln: Metrik „Score calculation count per step“ im Auge behalten

- Sehr gute Dokumentation, aktive und interessierte Community
  - Ausführliche Beispielprobleme decken viele praxisrelevante Use Cases ab (unterstützt am Anfang die Modellierung und erleichtert Einschätzung der Erfolgsaussichten)

OptaPlanner Roadmap

- Multithreading (Verteilung der Entscheidung über den nächsten Schritt auf mehrere Threads)
- Financial Optimization
- Execution Server, bisher eigene Implementierung z.B. mit Hilfe JMS oder Camel Komponente

akquinet

Vielen Dank für die Aufmerksamkeit!

Anregungen und Fragen?

Aktuelles: blog.akquinet.de
@akquinet
optaplanner.org