



How to Build Smarter Decision Models Capable To Make Optimal Decisions

Jacob Feldman, PhD
OpenRules, Inc., CTO
www.openrules.com

San Jose, Nov 4-6 2013

Decision Management Tools

Software Tools for Decision Management www.decision-tools.org

Tool Catalogues

Decision Model Collection

This website maintains **Live Catalogues of Decision Management Software** currently available on the market:

- **[Business Decisions and Rules Management Systems](#)**
- **[Predictive Analytics Tools](#)**
- **[Complex Event Processing and Real-Time Intelligence Tools](#)**
- **[Decision Modeling Tools](#)**
- **Decision Optimization**
 - **[Constraint Programming Solvers](#)**
 - **[Linear Programming Tools](#)**
- **[Business Process Management Software](#)**

www.decision-tools.org






Live Catalogues of Optimization Tools

Contains detailed profiles of open source and commercial tools:

- 30 Constraint Solvers
- 11 Linear Solvers

Tool profiles are maintained by their authors

Constraint Programming Solvers

Sponsored by:      [Become A Sponsor](#)

Catalogue of Constraint Programming Tools

Brieflists of [Constraint Programming](#) tools may be found at these sources: www.scp3.org, www.constraint-solvers.com, Wikipedia.Constraint-Solvers

This catalogue contains detailed profiles of constraint programming tools submitted by their developers to this website. The authors have an inclusive access and are solely responsible for the quality of the provided information about their tools.

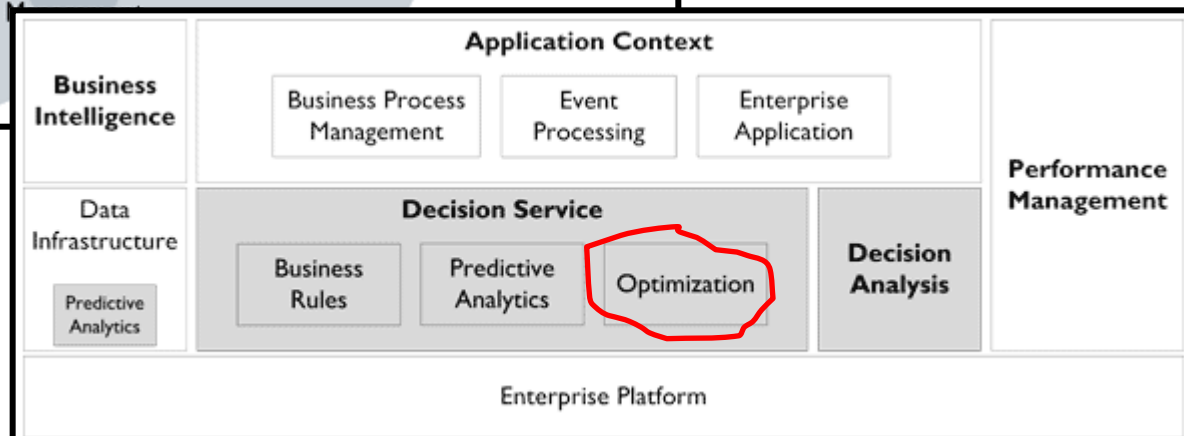
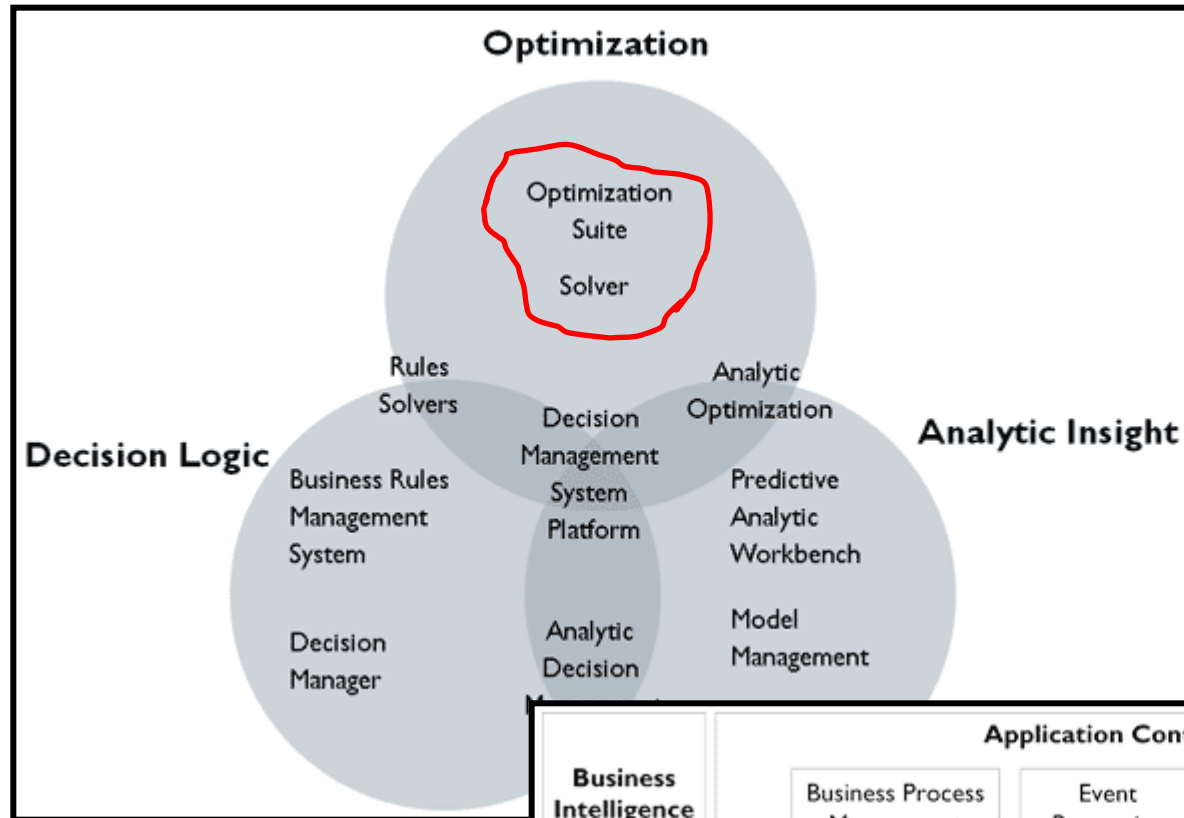
By clicking on the button "Add/Modify/Remove", authorized people may create profiles of their tools and keep them in the up-to-date status.

The Open-source buttons below allow you to generate the latest Open-Source Catalogue with all products sorted alphabetically:

[Generate Short Catalogue](#)
[Generate Complete Catalogue](#)
[Add/Modify/Remove](#)

Product Name	Brief Description	Website Forum	Implementation Language	Modeling Language	Support of Scheduling, Routing, ...	Integration with LP/MIP/SAT solvers	Solver Competitions, Awards	License, Pricing	First Release Year	Latest Release Date and Number	Total Number of Downloads	Number of Commercial Users	Submitted By	Last Update
AMPL	The Ames package integrates externally supplied solution algorithms for mathematical programming and constraint programming with model development, application integration, and application deployment.	Website Forum	Mostly C++, but also Java, Fortran, and C# are used.	AMPL	Scheduling is supported via ACTIVITIES and RESOURCES	LP, MIP, QP, MDP, NLP, MINLP, MCP, etc supported beyond CP	AMPLIS has been the optimization technology used to win the last three Franz Edelman Awards, see here	Commercial, but free for academic institutions	AMPLIS 1993, AMPLIS-CP 2012	-	-	several	Chia Kuo Pungen Decision Technology chia.kuo@delta.com	The Jun 27 07:46:18 GMT+01:00 2013
AMPL	AMPL is an efficient, multi-platform modeling system based on a high-level algebraic modeling language. It facilitates rapid development of optimization models and integration of this model into applications. AMPL supports a large variety of solvers including several constraint programming solvers and provides streamlined database access.	Website Forum	C, C++	AMPL	-	Supports most LP and MIP solvers, as well as quadratic, nonlinear, mixed-integer quadratic and nonlinear, semi-definite and some combinatorial programming, global optimization and complementarity problems.	Ralph Feun, David Gay and Brian Kuyghaven won awarded the 2012 INFORMS Impact Prize as the engineers of one of the most important algebraic modeling languages.	Commercial, free version for academic courses	1982	20130623	-	many	Vincent Zivich zivich@cam.ac.uk	Aug 10, 2013
CbcOpt	A finite sets constraint solver (as an Eclipse library) with special interfaces on one end and regular set interfaces on the other (minimum and maximum for sets of integers, and union for sets of sets.)	Website	Prolog	Prolog	-	(ECLIPSe integration)	-	ECLIPSe license	2004	2005.1.0	-	-	Francoise Astard astard@univ-lorraine.fr	Mon Jul 08 22:38:21 GMT+01:00 2013
CoSOL	Constraint Solving Platform for Engineering and Research	Website	C++	C++	-	-	Third International CSP Solver Competition	Apache License	2003	-	-	-	Stefano Celesia celesia@fct.unl.ac.uk	May 24, 2013
COCO	COCO is a Java library for constraint satisfaction problems (CSP) and constraint programming (CP).	Website	Java	Chess, MiniZinc	Scheduling (Cumulative Diffs, Regular, ...) Routing (Clubs, Substems, Test, Graphs, ...)	Explanation Engine	-	BSD (free)	1999	March 2013 Chess 12-02 101 forms version of Chess 3.0	>20100 since 2014	-	Charles Fred Heume heume@maths.univ-lorraine.fr	Mon Jul 15 16:19:44 GMT+01:00 2013
CocoHe	A mix of the set-lazy classes solver designed from the ground up with top class performance in mind.	-	C++	MiniZinc	-	LP propagator, built in SAT solver	-	closed source	-	-	-	-	Geoffrey Chu geoffrey.chu@univ-lorraine.fr	May 24, 2013
COCOHe	COCOHe is a black-box CSP solving API for the JVM, with the Scala and Java languages in mind.	Website	Scala	Scala, Java, and XOSP 2.1	-	-	Participated in CP2010, CP2012 and CSC09 competitions under the name COHe	GPL	2007	-	-	-	Julien Verc verc@univ-lorraine.fr	May 24, 2013
COCOHe	COCOHe is a constraint programming DSL (Domain-Specific Language) embedded in Scala. A SAT-based constraint solver Solver is used as a back-end constraint solver.	Website	Scala	Scala	-	Integrated with Sugar and Java (Java based SAT solver), Any SAT solver using DRONES CNF format can be used.	-	BSD 3-Clause License	2011	September 1, 2013 Version 2.2.0	-	-	Shinya Waki waki@univ-lorraine.fr	Sep 3, 2013
COCOHe	COCOHe is a library for constraint programming in Java.	Website	Java	Java	-	-	-	GPL	2003	January 24, 2009 Version 1.0.0	About 6000 since 2003	-	Shinya Waki waki@univ-lorraine.fr	Sep 3, 2013
ECLIPSe	Constraint Logic Programming system, consisting of runtime core, set of libraries, modeling and control language, development environment, host language interface, and interfaces to third-party solvers.	Website Forum	C, Prolog	ECLIPSe Extended Prolog, Minisat	-	Tight LP/MIP integration with Cplex, Xpress-MP, CDD/QR and Quesha	-	Open Source, Minisat Public License	1992	2013.6.1	>10000	several	Frankfurt Schimpf frankfurt@univ-lorraine.fr	May 24, 2013
Gecode	An open source C++ constraint solver	Website Forum	C++	MiniZinc, AMPL	scheduling bin packing	-	-	Open Source MIT license	2003	2013.4.2.0	more than 40000, also included in Linux distributions (Debian, Ubuntu, Gentoo, OpenSUSE and FreeBSD)	-	Christian Schulte schulte@univ-lorraine.fr	Fri Jul 19 14:23:08 GMT+01:00 2013
IBM ILOG CPLEX	-	-	-	-	-	-	-	-	-	-	-	-	Yves Ben-Haim YVESB@il.ibm.com	Sep 2, 2013
IBM ILOG CPLEX Optimizer	IBM ILOG CPLEX Optimizer is a multi-threaded optimizer based on Constraint Programming that solves constraint satisfaction and optimization problems. Special emphasis is placed on modeling and solving scheduling problems for which CP Optimizer provides simple but powerful modeling concepts such as resources, activities and machines. For many applications, the robust and efficient automatic search of CP Optimizer comes on the head for the user to write and maintain their own search strategy. However, for those applications that need special propagation algorithms and dedicated search strategies, CP Optimizer also provides the capability of defining custom constraints as well as full control over the solution search process.	Website Forum	C++	OPL, C++, Java, C#	-	Suggests scheduling algorithms to minimize scheduling, alternative resource allocation, projects work-breakdown structures, temporal event trees as resource tightness, non-convex costs, ... and solving (requires variables, resources, duration, variables expressions).	IBM ILOG CPLEX Optimizer is a component of IBM ILOG CPLEX Optimizer Suite that also contains CPLEX Optimizer for LP/MIP/QCP/CP Optimizer can be tightly integrated with CPLEX Optimizer in user applications.	Commercial, but free for academic users	CP Optimizer 1.0, May 2007	CPLEX, Version 12.1.0, June 2013	-	-	Philippe Labadie labadie@il.ibm.com	Aug 14, 2013
SCIP	SCIP (Solving Constraint Integer Programs) is a general purpose solver for constraint programming, integer programming, and global optimization.	Website Forum	C++	AMPL, MiniZinc, GAMS, Pyomo, etc.	-	-	-	Open Source, MIT License	2003	2013.4.2.0	-	-	Yves Ben-Haim YVESB@il.ibm.com	Thu Jul 18 17:07:31

Decision Optimization within Decision Management Platforms



Decision Optimization

- Decision Optimization helps *business* people to:
 - make *better decisions* subject to different *business* and *time* constraints
 - consider alternative decisions
 - find *optimal* (or close to optimal) decisions
- Finds the *best* possible resource utilization to achieve a desired optimization objective such as:
 - minimizing expenses or travel time
 - maximizing ROI or service level
- Relies on proven mathematical techniques such as:
 - Constraint Programming (CP) and Linear Programming (LP)
 - Integrated in modern decision modeling frameworks

Typical Optimization Applications

- Scheduling and Resource Allocation
- Complex Configuration Problems
- Supply Chain Management
- Staff Rostering
- Vehicle Routing

Delivery planning

File Solve Layer Optimization Options Help

Locator: [dropdown]

Computation completed...

Map Geographical locations Sites Vehicles Deliveries Routing Plan

Moulding Shop - Gantt View

File Break Schedule Help

all [dropdown] 1on Tue Wed Thu Fri Sat Sun Mc

M0
M1
M2
M3

Name: CSP-A19 Start: 22:01: End: 00:01: B. start: B. end:
Prod.: 1680 (nb parts) Cons.: 3600 (kg)

CST [dropdown]

8000
6400
4800
3200
1600
0

Mon Tue Wed

Available (blue) Con (red)

Planning

Play Pause Step Reset R M E N R M E N

	S	M	T	W	T	F	S	S	S	M	T	W	T	F	S	S	S	M	T	W	T	F	S				
MORMAL Valerie	R	M	M	M	E	R	R	E	E	E	N	N	N	E	N	E	N	M	M	N	N	E	R	3	6	6	6
NEVE Olivier	R	M	M	M	E	R	R	E	E	E	N	N	N	E	N	E	N	M	M	N	N	E	R	6	4	6	5
CAMU Bernard	R	M	M	M	E	R	R	E	E	E	N	N	N	E	N	E	N	M	M	N	N	E	R	6	5	4	6
CONTENT Frederik	M	E	E	E	E	M	M	N	N	N	R	R	R	R	N	R	R	N	R	R	N	R	N	6	4	4	5
MARELLA Eric	E	E	E	E	M	N	N	M	M	M	R	R	R	R	R	R	R	N	R	R	N	R	R	9	4	4	5
SEPELS Antonio	N	E	E	E	E	M	N	M	M	M	R	R	R	R	R	R	R	N	R	R	N	R	R	9	4	4	4
JADOUL Juan	R	N	N	N	E	N	E	M	M	M	E	M	M	E	E	R	R	R	R	R	R	R	R	9	4	4	4
VAN DEN HURK Jan	R	N	N	N	R	R	R	R	R	R	M	M	M	M	E	E	M	M	E	E	R	R	R	9	4	4	4
ARCOS Christophe	R	N	N	N	R	R	R	R	R	R	N	N	N	E	M	R	R	M	M	M	E	E	E	9	4	4	4
MOUSSA Simon	R	R	R	R	R	N	R	R	R	N	N	N	N	E	M	R	R	M	M	M	E	E	E	9	4	4	4
LIENARD Olivier	R	R	R	R	N	R	R	R	R	N	N	N	N	E	E	R	R	M	M	M	E	E	E	9	4	4	4
PEPN Sandra	R	R	R	N	N	R	N	R	R	R	E	E	R	R	E	E	M	M	M	M	M	M	M	9	4	4	4

9 3 3 1 2 2 8 9 3 3 3 3 9 9 3 3 3 3 9
 1 3 3 3 3 1 1 3 3 3 3 3 1 1 3 3 3 3 1
 1 3 3 4 3 3 1 1 3 3 3 3 1 1 3 3 3 3 1
 1 3 3 4 4 4 2 1 3 3 3 3 1 1 3 3 3 3 1

Constraint Satisfaction Problem - CSP

- CSP represents a decision optimization problems defining decision variables subject to constraints
- Typical CSP structure:
 - 1. Problem Definition (what to do)**
 - a. Define Decision Variables with all possible values
 - b. Define Constraints on the decision variables
 - 2. Problem Resolution (how to do it)**
 - a. Find Solution(s) that defines a value for each variable such that all constraints are satisfied or
 - b. Find an optimal solution that minimizes/maximizes a certain objective (a function of decision variables)

JSR-331 “Java CP API” Standard

- JSR-331 “Java Constraint Programming API” – an official Java Community Process (JCP) standard
www.jsr331.org
- JSR-331 covers key concepts and design decisions related to the standard representation and resolution of constraint satisfaction and optimization problems
- Utilizes de-facto standardized decisions from multiple CP solvers
- JSR-331 current implementations:
 - **4 CP-based solvers**
 - **6 LP-based solvers**

Simple CSP in Java (JSR-331)

```
public static void main(String[] args) {
    // ==== PROBLEM DEFINITION =====
    Problem p = ProblemFactory.newProblem("Test");
    // ===== Define variables
    Var x = p.variable("X", 1, 10);
    Var y = p.variable("Y", 1, 10);
    Var z = p.variable("Z", 1, 10);
    Var cost = x.multiply(3).multiply(y).minus(z.multiply(4));
    // ===== Define and post constraints
    p.post(x, "<", y); // X < Y
    p.post(x.plus(y), "=", z); // X + Y = Z

    // === PROBLEM RESOLUTION =====
    p.log("=== Find Solution:");
    Solver solver = p.getSolver();
    Solution solution = solver.findSolution();
    if (solution != null)
        solution.log();
    else
        p.log("No Solution");
    p.log("Cost " + cost);
}
```



Solution solution = solver.findOptimalSolution(cost);

Moving From Java to Business Decision Modeling

- Building business-oriented decision modeling facilities on top of the JSR-331
- Allowing business analysts to define a decision optimization problem
 - Using business concepts and decision variables (glossary)
 - Using predefined or custom constraints oriented to business people
- Relying on the standard solvers to solve the problem
- OpenRules® [Rule Solver](#) provides a decision optimization environment using intuitive decision tables managed in Excel or Google Docs

Example: Where is Zebra?

1. There are five houses
2. The Englishman lives in the red house
3. The Spaniard owns the dog
4. Coffee is drunk in the green house
5. The Ukrainian drinks tea
6. The green house is immediately to the right of the ivory house
7. The Old Gold smoker owns snails
8. Kools are smoked in the yellow house
9. Milk is drunk in the middle house
10. The Norwegian lives in the first house
11. The man who smokes Chesterfields lives in the house next to the man with the fox
12. Kools are smoked in the house next to the house where the horse is kept
13. The Lucky Strike smoker drinks orange juice
14. The Japanese smokes Parliaments
15. The Norwegian lives next to the blue house

Glossary

Glossary glossary					
	Decision Variables	Business Concept	Attribute	Domain	Unknown
Colors	green	Zebra Problem	green	0-4	TRUE
	ivory		ivory	0-4	TRUE
	blue		blue	0-4	TRUE
	red		red	0-4	TRUE
	yellow		yellow	0-4	TRUE
People	Norwegian		norwegian	0-4	TRUE
	Ukrainian		ukrainian	0-4	TRUE
	Japanese		japanese	0-4	TRUE
	Englishman		englishman	0-4	TRUE
	Spaniard		spaniard	0-4	TRUE
Drinks	juice		juice	0-4	TRUE
	tea		tea	0-4	TRUE
	milk		milk	0-4	TRUE
	water		water	0-4	TRUE
	coffee		coffee	0-4	TRUE
Pets	snail		snail	0-4	TRUE
	dog		dog	0-4	TRUE
	fox		fox	0-4	TRUE
	horse		horse	0-4	TRUE
	ZEBRA		zebra	0-4	TRUE
Cigarettes	Chesterfield		chesterfield	0-4	TRUE
	Parliament		parliament	0-4	TRUE
	Lucky		lucky	0-4	TRUE
	OldGolds		oldGolds	0-4	TRUE
	Kools		kools	0-4	TRUE

Decision “FindZebra”

Decision FindZebra	
Decisions	Execute Rules
All Diff Constraints	:= AllDiffConstraints()
Zebra Constraints 1	:= ZebraConstraints1()
Zebra Constraints 2	:= ZebraConstraints2()

All Different Constraints

DecisionTable AllDiffConstraints
ActionAllDiff
Variables
green,ivory,blue,red,yellow
Norwegian,Ukrainian,Japanese,Englishman,Spaniard
juice,tea,milk,water,coffee
snail,dog,fox,horse,ZEBRA
Chesterfield,Parliament,Lucky,OldGolds,Kools

Zebra Constraints 1

DecisionTable ZebraConstraints1			
ActionXoperY			
X <oper> Y			
Englishman	=	red	The Englishman lives in the red house
Spaniard	=	dog	The Spaniard owns the dog
coffee	=	green	Coffee is drunk in the green house
Ukrainian	=	tea	The Ukrainian drinks tea
OldGolds	=	snail	The Old Golds smoker owns snails
Kools	=	yellow	Kools are smoked in the yellow house
milk	=	2	Milk is drunk in the middle house
Norwegian	=	0	The Norwegian lives in the first house
Lucky	=	juice	The Lucky Strike smoker drinks orange juice
Japanese	=	Parliament	The Japanese smokes Parliament

Zebra Constraints 2

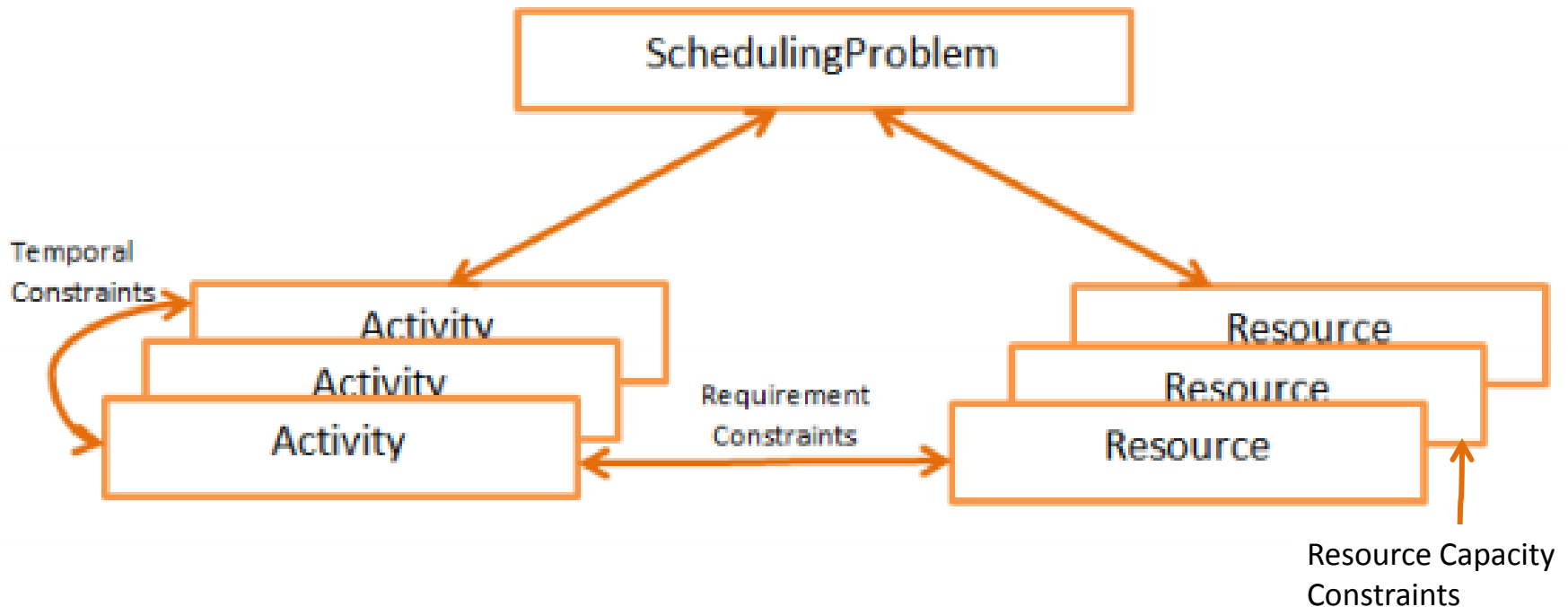
DecisionTable ZebraConstraints2	
ActionConstraint	
Constraint	
<pre>{ Var green = getVar("green"); Var ivory = getVar("ivory"); solver().linear(green,"=",ivory.plus(1)); }</pre>	The green house is immediately to the right of the ivory house
<pre>{ Var Chesterfield = getVar("Chesterfield"); Var fox = getVar("fox"); Constraint right = linear(Chesterfield,"=",fox.plus(1)); Constraint left = linear(Chesterfield,"=",fox.minus(1)); right.or(left); }</pre>	The man who smokes Chesterfields lives in the house next to the man with the fox
<pre>{ Var horse = getVar("horse"); Var Kools = getVar("Kools"); Constraint right = linear(Kools,"=",horse.plus(1)); Constraint left = linear(Kools,"=",horse.minus(1)); right.or(left); }</pre>	Kools are smoked in the house next to the house where the horse is kept.
<pre>{ Var Norwegian = getVar("Norwegian"); Var blue = getVar("blue"); linear(Norwegian,"=",blue.plus(1)).or(linear(Norwegian,"=",blue.minus(1))); }</pre>	The Norwegian lives next to the blue house

Zebra Execution Results

House #1: fox yellow Kools water Norwegian
House #2: Chesterfield Ukrainian horse tea blue
House #3: OldGolds milk red Englishman snail
House #4: Lucky juice ivory Spaniard dog
House #5: Parliament Japanese ZEBRA green coffee

OpenRules® Rule Solver

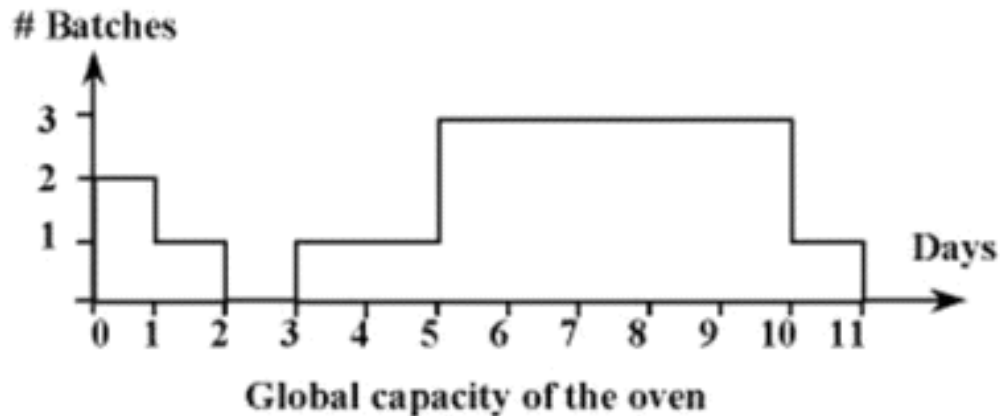
- Includes decision table templates for various binary and global constraints
- Includes decision table templates for scheduling and resource allocation problems:



Real-world Resource Allocation Decisions

- Many service providers have to make every day decisions by allocation their limited capacity resources to satisfy various customer requests
- Let's consider a small business that fires batches of bricks in one or several ovens (a resource with a limited capacity)
- This business has to make resource allocation decisions on a constant basis by allocating their limited resources to customer orders while satisfying scheduling and resource constraints

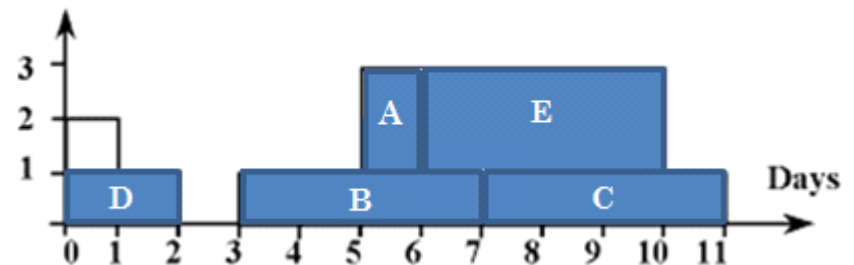
Example of a Resource Allocation Problem



- A** 2 batches, 1 day
- B** 1 batch, 4 days
- C** 1 batch, 4 days
- D** 1 batch, 2 days
- E** 2 batches, 4 days

5 Activities

Possible Solutions:



Decision “DefineOvenSchedule”

This decision consists of 5 sub-decisions:

Decision DefineOvenSchedule	
Decisions	Execute Rules
Define Schedule	:= DefineSchedule()
Define Activities	:= DefineActivities()
Define Oven as Recoverable Resource	:= DefineOvenAsResource()
Define Oven Availability	:= SetOvenCapacities()
Define Resource Requirement Constraints	:= ResourceRequirementConstraints()

Define Schedule and Activities

DecisionTable DefineSchedule	
ActionSchedule	
Origin	Horizon
0	11

A 2 batches, 1 day

B 1 batch, 4 days

C 1 batch, 4 days

D 1 batch, 2 days

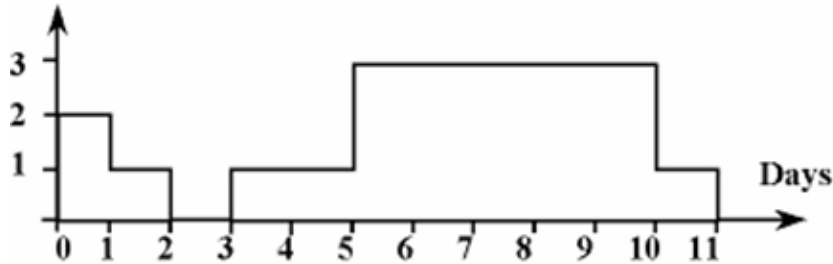
E 2 batches, 4 days

5 Activities

DecisionTable DefineActivities	
ActionAddActivity	
Name	Duration
A	1
B	4
C	4
D	2
E	4

Define Oven and its Availability

DecisionTable DefineOvenAsResource		
ActionAddResource		
Name	Resource Type (Consumable/Recoverable)	Max Capacity
Oven	Recoverable	3



DecisionTable SetOvenCapacities			
ActionResourceCapMax			
Resource	From	To	Capacity
Oven	0	1	2
Oven	1	2	1
Oven	2	3	0
Oven	3	5	1
Oven	5	10	3
Oven	10	11	1

Define Resource Requirements

- A** 2 batches, 1 day
- B** 1 batch, 4 days
- C** 1 batch, 4 days
- D** 1 batch, 2 days
- E** 2 batches, 4 days

DecisionTable ResourceRequirementConstraints		
ActionActReqResource		
Activity	Required Resource	Required Capacity
A	Oven	2
B	Oven	1
C	Oven	1
D	Oven	1
E	Oven	2

Run Decision from Java to Find One Feasible Solution

```
public class Main {  
  
    public static void main(String[] args) {  
  
        String fileName = "file:rules/Decision.xls";  
        System.setProperty("OPENRULES_MODE", "Solve");  
        Decision decision = new Decision("DefineOvenSchedule", fileName);  
        decision.execute();  
    }  
}
```

One Feasible Solution

SOLUTION:

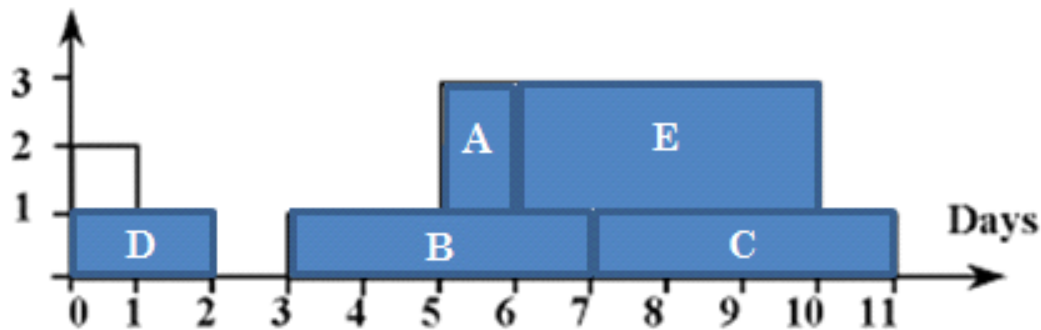
A[5 -- 1 --> 6) requires oven[2]

B[3 -- 4 --> 7) requires oven[1]

C[7 -- 4 --> 11) requires oven[1]

D[0 -- 2 --> 2) requires oven[1]

E[6 -- 4 --> 10) requires oven[2]



Run Decision from Java to Find ALL Feasible Solutions

```
public class Main {  
  
    public static void main(String[] args) {  
  
        String fileName = "file:rules/Decision.xls";  
        System.setProperty("OPENRULES_MODE", "Solve");  
        Decision decision = new Decision("DefineOvenSchedule", fileName);  
        decision.put("MaxSolutions", "10");  
        decision.execute();  
    }  
}
```

All Feasible Solutions

Solution 1:

A[5 -- 1 --> 6) requires Oven[2]
B[3 -- 4 --> 7) requires Oven[1]
C[7 -- 4 --> 11) requires Oven[1]
D[0 -- 2 --> 2) requires Oven[1]
E[6 -- 4 --> 10) requires Oven[2]

Solution 2:

A[5 -- 1 --> 6) requires Oven[2]
B[7 -- 4 --> 11) requires Oven[1]
C[3 -- 4 --> 7) requires Oven[1]
D[0 -- 2 --> 2) requires Oven[1]
E[6 -- 4 --> 10) requires Oven[2]

Solution 3:

A[9 -- 1 --> 10) requires Oven[2]
B[3 -- 4 --> 7) requires Oven[1]
C[7 -- 4 --> 11) requires Oven[1]
D[0 -- 2 --> 2) requires Oven[1]
E[5 -- 4 --> 9) requires Oven[2]

Solution 4:

A[9 -- 1 --> 10) requires Oven[2]
B[7 -- 4 --> 11) requires Oven[1]
C[3 -- 4 --> 7) requires Oven[1]
D[0 -- 2 --> 2) requires Oven[1]
E[5 -- 4 --> 9) requires Oven[2]

Another Real-World Example

“Workforce Management”

- Workforce management is central to efficient operations and good customer service.
- Proper scheduling of employees can mean the difference between profitability and business failure.
- As the manager, you are required to hire and set the weekly work schedule for your employees.

Employee Scheduling Requirements

- The required levels for the week are as follows:
 - Total employees required

Mon	Tue	Wed	Thu	Fri	Sat	Sun
5	8	9	10	16	18	12

- Available employees:

Employee Type	Total	Cost per Day
F/T	14	\$100
P/T	4	\$150

Possible Solution:

	M	T	W	T	F	S	S
FT	5	8	9	10	14	14	12
PT	0	0	0	0	2	4	0

- Assuming the same staffing requirements continue week after week, **what is the minimal staffing cost?**

Decision “DefineEmployeeSchedule”

- Start with a decision
- Presented in Excel using OpenRules Rule Solver

Mon	Tue	Wed	Thu	Fri	Sat	Sun
5	8	9	10	16	18	12

Decision DefineEmployeeSchedule	
Decisions	Execute Rules
Define Employee Daily Demand	:= EmployeeDailyDemand()
Define Total Cost	:= DefineTotalCost()

Employee Type	Total	Cost per Day
F/T	14	\$100
P/T	4	\$150

Decision's Glossary

- Decision Variables
- For each day one for FT and one for PT

Glossary glossary				
Decision Variable	Business Concept	Attribute	Domain	Unknown
Mon FT	Roster	monFT	0-14	TRUE
Mon PT		monPT	0-4	TRUE
Tue FT		tueFT	0-14	TRUE
Tue PT		tuePT	0-4	TRUE
Wed FT		wedFT	0-14	TRUE
Wed PT		wedPT	0-4	TRUE
Thu FT		thuFT	0-14	TRUE
Thu PT		thuPT	0-4	TRUE
Fri FT		friFT	0-14	TRUE
Fri PT		friPT	0-4	TRUE
Sat FT		satFT	0-14	TRUE
Sat PT		satPT	0-4	TRUE
Sun FT		sunFT	0-14	TRUE
Sun PT		sunPT	0-4	TRUE
Total Cost		totalCost	0-20000	TRUE

Decision Table

“EmployeeDailyDemand”

Decision Table EmployeeDailyDemand				
ActionXoperYcompareZ				
Variable	Arith Oper	Variable	Compare Oper	Value
Mon FT	+	Mon PT	=	5
Tue FT	+	Tue PT	=	8
Wed FT	+	Wed PT	=	9
Thu FT	+	Thu PT	=	10
Fri FT	+	Fri PT	=	16
Sat FT	+	Sat PT	=	18
Sun FT	+	Sun PT	=	12

Mon	Tue	Wed	Thu	Fri	Sat	Sun
5	8	9	10	16	18	12

Decision Table “DefineTotalCost”

Decision Table DefineTotalCost		
ActionScalProd		
Name of the Scalar Product	Numbers	Variables
Total Cost	100,150,100,150,100,150, 100,150,100,150,100,150, 100,150	Mon FT, Mon PT, Tue FT, Tue PT, Wed FT, Wed PT, Thu FT, Thu PT, Fri FT, Fri PT, Sat FT, Sat PT, Sun FT, Sun PT

Employee Type	Total	Cost per Day
F/T	14	\$100
P/T	4	\$150

Run Decision from Java to Find One Feasible Solution

```
public class Main {  
  
    public static void main(String[] args) {  
  
        String fileName = "file:rules/Decision.xls";  
        System.setProperty("OPENRULES_MODE", "Solve");  
        Decision decision = new Decision("DefineEmployeeSchedule", fileName);  
        decision.put("MaxSolutions", "30");  
        decision.put("Minimize", "Total Cost");  
        decision.execute();  
        decision.execute("PrintSolution");  
    }  
}
```

Decision Results

...

Found a solution with Total Cost[8700]
 Found a solution with Total Cost[8650]
 Found a solution with Total Cost[8600]
 Found a solution with Total Cost[8550]
 Found a solution with Total Cost[8500]
 Found a solution with Total Cost[8450]
 Found a solution with Total Cost[8400]
 Found a solution with Total Cost[8350]
 Found a solution with Total Cost[8300]
 Found a solution with Total Cost[8250]
 Found a solution with Total Cost[8200]
 Found a solution with Total Cost[8150]
 Found a solution with Total Cost[8100]
 Found a solution with Total Cost[8100]

*** Execution Profile ***

Number of Choice Points: 94360

Number of Failures: 94333

Occupied memory: 93172496

Execution time: 14885 msec

===== Optimal Solution =====							
	M	T	W	T	F	S	S
FT	5	8	9	10	14	14	12
PT	0	0	0	0	2	4	0
COST: 8100							
=====							

Smarter Search Strategies

- Adding time limits for:
 - Search of one solution
 - The overall search
- CP solvers provide many search strategies for selecting variables and values to try first, e.g.
 - Notorious “N-Queens” problem: using a selector `MIN_DOMAIN_MIN_VALUE` improves performance 1,000 times
- CP/LP tools provide different optimization options that may be tried without changing a decision model

Conclusion

- Combination of BR and CP/LP tools creates a powerful while intuitive decision modeling and optimization framework
- Many practical Decision Optimization problems may be successfully modeled and solved by subject matter experts using off-the-shelf CP/LP tools such as [OpenRules Rule Solver](#)
- The [JSR-331](#) standard gives all BR vendors an opportunity to add a true optimization component to their product offerings

Q&A

Web: www.OpenRules.com

Email:

support@openrules.com

jacobfeldman@openrules.com