=== Decision Management Community ===

Challenge July 2024 "Smart Investment"

Solution with OpenRules Rule Solver

by Ian Detinich, iandetinich@gmail.com

Problem Statement

The problem was defined at https://dmcommunity.org/challenge-july-2024/ as a Challenge July-2024:

Smart Investment Solutions

A client of an investment firm has $10000 available for investment. He has instructed that his money be invested in particular stocks, so that no more than $5000 is invested in any one stock but at least $1000 be invested in each stock. He has further instructed the firm to use its current data and invest in the manner that maximizes his overall gain during a one-year period. The stocks, the current price per share and the firm's predicted stock price a year from now are summarized below:

Stock	Current Price	Projected Price 1 year
ABC	$25	$35
XYZ	$50	$60
TTT	$100	$125
LMN	$25	$40

You task is to create a decision model that can be used to make a smart investment while satisfying the client requirements for different combinations of stocks. Send your solutions to DecisionManagementCommunity@gmail.com.

Introduction

As a student of Seton Hall University, I have already submitted my solution using Excel Solver which I used for similar optimization problems in my class "Quantitative Decision Making". This summer I am interning at OpenRules, Inc. I was assigned to work with OpenRules Rule Solver, which from my understanding is a perfect fit for such problems.

OpenRules recommends defining the problem in an Excel file that contains the problem Glossary with different decision variables, rules, and solution search methods. So, I created an Excel file "DecisionModelInvest.xls" available here.

Problem Definition

First, I attempted to present major problem concepts in the following Glossary:

Glossary glossary				
Variables	**Business Concept**	**Attribute**	**Type**	**Used As**
Total Amount to Invest	**Investment**	totalAmount	int	in
Min to Invest		minInvest	int	in
Max to Invest		maxInvest	int	in
Stocks		stocks	Stock[]	in
Overall Gain		gain	int	out
Stock Name	**Stock**	name	String	in
Current Price		currentPrice	int	in
Projected Price		projectedPrice	int	in

Here we used the business concept "Investment" which refers to our input data such as "Total Amount to Invest", "Min to Invest", "Max to Invest", and our only output variable "Overall Gain." Each Investment contains the array "Stocks" of the type Stock[] where each Stock is defined by its Name, Price, and Projected Price.

Having this Glossary, allowed me to define the test case from the Challenge in the table "DecisionTest":

DecisionTest testCases					
#	ActionDefine	ActionDefine	ActionDefine	ActionDefine	ActionExpect
Test ID	**Total Amount to Invest**	**Min to Invest**	**Max to Invest**	**Stocks**	**Overall Gain**
1	10000	1000	5000	stocks4	4650

All stocks are defined in a separate table “DecisionData”:

DecisionData Stock stocks4		
Stock Name	Current Price	Projected Price
ABC	25	35
XYZ	50	60
TTT	100	125
LMN	25	40

As I already knew (from my Excel Solver’s solution) that the optimal Overall Gain should be $4650, I put this expected value in the column “ActionExpect” as shown above.

Defining Decision Variables

For this problem, key constrained (unknown) variables represent how many stocks we want to purchase. Their values vary from 0 to a possible maximum investment. So first, we should define and calculate an intermediate decision variable “Possible Maximum Investment”. It can be calculated as the Total Amount to Invest divided by the Cheapest Stock Price. In OpenRules it can be done simply in the following decision table:

Decision DefinePossibleMaximumInvestment	
Action	Action
Cheapest Stock Price	**Possible Maximum Investment**
Min of Current Price of Stocks	Total Amount to Invest / Cheapest Stock Price

We can also note the temporary variables “Cheapest Stock Price” and “Total Amount to Invest” which can be added to the Glossary. Now, we are ready to define our unknown variable using the Rule Solver column “SolverDefineVariables”:

Decision DefineStockVariables [for each Stock in Stocks]			
SolverDefineVariables			
Variable Name	**Method**	**Par 1**	**Par 2**
Stock Name	"New Variable"	0	"Possible Maximum Investment"
"Stock Variables"	"Add Variable"	Stock Name	

This table for each Stock in the array of Stocks applies two rules:

1) The first rule creates a New Variable with the same name as the stock defined from 0 to “Possible Maximum Investment”
2) The second rule adds this variable to the array “Stock Variables” which we will need to define as “Overall Gain Variable”.

To define investment Min/Max constraints, we will need to define for each stock the corresponding investment constrained variables. We may add their definition to the above table as follows:

Decision DefineStockVariables [for each Stock in Stocks]				
SolverDefineVariables				
Variable Name	**Method**	**Par 1**	**Par 2**	**Par 3**
Stock Name	"New Variable"	0	"Possible Maximum Investment"	
"Stock Variables"	"Add Variable"	Stock Name		
Investment in {{Stock Name}}	"Variable Operator Value"	Stock Name	"*"	"Current Price"
"Stock Investment Variables"	"Add Variable"	Investment in {{Stock Name}}		

Post investment Min/Max investment constraints for each stock using the following table:

Decision PostMinMaxConstraints [for each Stock in Stocks]				
SolverPostConstraints				
Constraint Name	**Constraint Type**	**Par 1**	**Par 2**	**Par 3**
"Min Constraint"	"Variable Operator Value"	Investment in {{Stock Name}}	">="	"Min to Invest"
"Max Constraint"		Investment in {{Stock Name}}	"<="	"Max to Invest"

Then, I used Rule Solver's methods "Sum" and "Scalar Product" to define constrained variables that represent the Total Investment and Overall Gain:

Decision DefineTotalInvestmentVariables			
SolverDefineVariables			
Variable Name	**Method**	**Par 1**	**Par 2**
"Total Investment Variable"	"Sum"	"Stock Investment Variables"	
"Overall Gain Variable"	"Scalar Product"	"Stock Variables"	"Stock Profits"

To limit the Total Investment, I posted the following constraint:

Decision PostOverallGainConstraint				
SolverPostConstraints				
Constraint Name	**Constraint Type**	**Par 1**	**Par 2**	**Par 3**
"Total Investment <= 10K"	"Variable Operator Value"	"Total Investment Variable"	"<="	"Total Amount to Invest"

Problem Resolution

OpenRules Rule Solver usually recommends creating two tables:

- "**Define**" that defines problem variables and constraints
- "**Solve**" that defines the problem objective and calls predefined search methods

The table "Define" invokes all the above tables:

Decision Define
ActionExecute
Decision Tables
DefinePossibleMaximumInvestment
DefineStockProfits
DefineStockVariables
DefineTotalInvestmentVariables
PostMinMaxConstraints
PostOverallGainConstraint

The table "Solve" should do at least 3 things:

1) Set "Overall Gain Variable" as the optimization objective
2) Invoke the default method "SolverMaximize" that will maximize this objective
3) Assigned the found optimal value of the constrained variable "Overall Gain Variable" to our resulting output variable "Overall Gain".

Here is my initial implementation of the table "Solve":

Decision Solve			
SolverSetObjective	ActionExecute	SolverAssignSolution	
Objective	**Actions**	Business Variable Name	Solver Variable Name
"Overall Gain Variable"	SolverMaximize	"Overall Gain"	Overall Gain Variable

Execution Results

Next, I executed this decision model against our test data using the standard file "test.bat". It produced the expected results:

```
Optimal solution is found. Objective: 4650
Solution #241:
        ABC[120] Investment in ABC[3000] XYZ[20] Investment in XYZ[1000] TTT[10]
Investment in TTT[1000] LMN[200] Investment in LMN[5000] Total Investment Variable
[10000]
        Overall Gain Variable[4650] -Overall Gain Variable[-4650]
*** Execution Profile ***
Number of Choice Points: 11911
Number of Failures: 11053
Execution time: 6294 msec
        Variables:
            Overall Gain: 0 --> 4650
```

As expected, the Overall Gain was defined as 4650, and investments in different stocks corresponded to my Excel Solver solution.

Advanced Decision Model

While my decision model produced the expected results, it took 6 seconds to find the optimal solution after finding 240 other solutions. When I consulted with OpenRules experts, they recommended improving my decision model in several directions:

1) Try a different search algorithm called “Dichotomize”
2) Add for each stock the output variables “Recommended Number of Shares to Buy”
3) Add Explanations to the search
4) Deploy the decision model as an AWS Lambda function to be able to invoke it using JSON data.

Below I’ll describe how I did it.

First, I expanded the Glossary by adding an array called “Solver Explanations” to Investment and the decision variable “Recommended Number of Shares to Buy” to Stock. They will be Used as output attributes of my decision model:

Glossary glossary				
Variables	**Business Concept**	**Attribute**	**Type**	**Used As**
Total Amount to Invest	**Investment**	totalAmount	int	in
Min to Invest		minInvest	int	in
Max to Invest		maxInvest	int	in
Stocks		stocks	Stock[]	in
Solver Explanations		explanations	String[]	out
Overall Gain		gain	int	out
Stock Name	**Stock**	name	String	in
Current Price		currentPrice	int	in
Projected Price		projectedPrice	int	in
Recommended Number of Shares to Buy		recommendedNumberOf	int	out
Cheapest Stock Price	Temp	cheapestStockPrice	int	
Possible Maximum Investment		possibleInvestmentMax	int	
Stock Profits		stockProfits	int[]	

OpenRules has a special column labeled “SolverSettings” that can be used inside a regular decision table. So, I added the table “DefineSearchParameters”:

Decision DefineSearchParameters	
SolverSettings	
Setting	**Value**
"Objective"	"Overall Gain Variable"
"Optimization Strategy"	"Dichotomize"

I then also moved the assignment of Overall Gain to a separate table:

Decision AssignOverallGain	
SolverAssignSolution	
Business Variable Name	Solver Variable Name
"Overall Gain"	Overall Gain Variable

To assign the optimal values of all stocks to the variable “Recommended Number of Shares to Buy” I used the following decision table:

Decision AssignRecommendedShares [for each Stock in Stocks]	
SolverAssignSolution	
Business Variable Name	Solver Variable Name
"Recommended Number of Shares to Buy"	Stock Name

To save Solver's explanations (if any) in my array "Explanations", I used the following table:

Decision SaveExplanations
SolverSaveExplanations
Name of Array with Explanations
"Solver Explanations"

And finally, I modified the table "Solve" as follows:

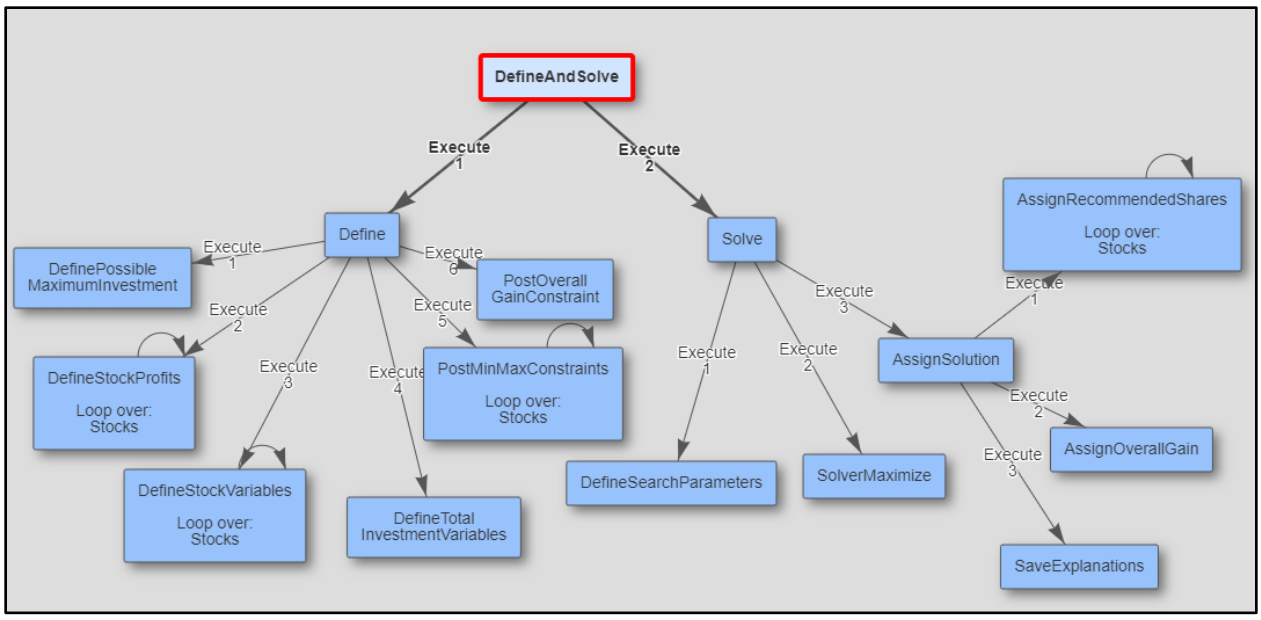
Decision Solve
ActionExecute
Actions
DefineSearchParameters
SolverMaximize
AssignSolution

I obtained the same results.

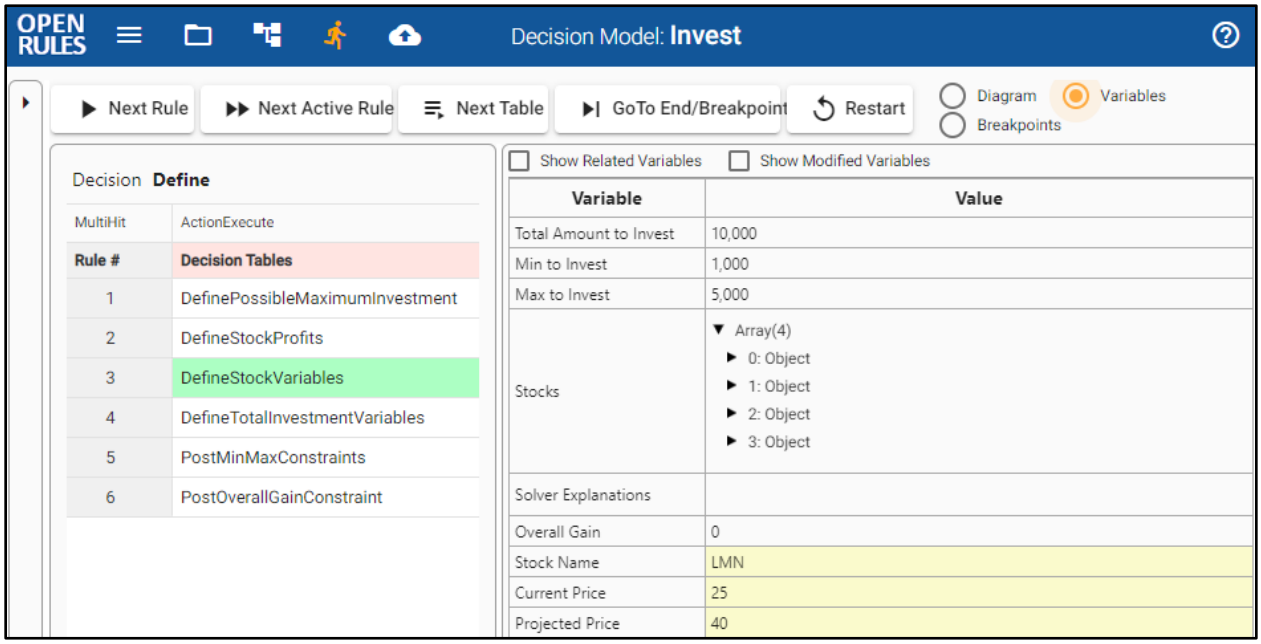
```
Optimal solution is found. Objective: -4650
Solution #9:
        ABC[120] Investment in ABC[3000] XYZ[20] Investment in XYZ[1000] TTT[10] Investment in
 TTT[1000] LMN[200] Investment in LMN[5000] Total Investment Variable[10000]
        Overall Gain Variable[4650] -Overall Gain Variable[-4650]
*** Execution Profile ***
Number of Choice Points: 1533
Number of Failures: 1522
Execution time: 141 msec
```

This time the search took only 141 milliseconds instead of 6 seconds, and the optimization strategy "Dichotomize" (specified in the above table "DefineSearchParameters") found an optimal solution from the ninth attempt.

I also tried to visualize my decision model using OpenRules IDE. It automatically built a graphical diagram of my decision model:



I was able to use OpenRules Debugger to execute all the rules of my decision model one by one and analyze the values of all decision variables. Here is one of the Debugger's snapshots:



Deploying the Decision Model as an AWS Lambda Function

I was advised to deploy my decision model on the cloud using one of the default deployment mechanisms supported by OpenRules. I tried to deploy my decision model as a RESTful decision service available from Amazon AWS via JSON interface. I was told that AWS Lambda function is one of the most popular choices for many production-level decision services. Here is what I did.

I added AWS Lambada deployment properties to the default OpenRules configuration file "project.properties":

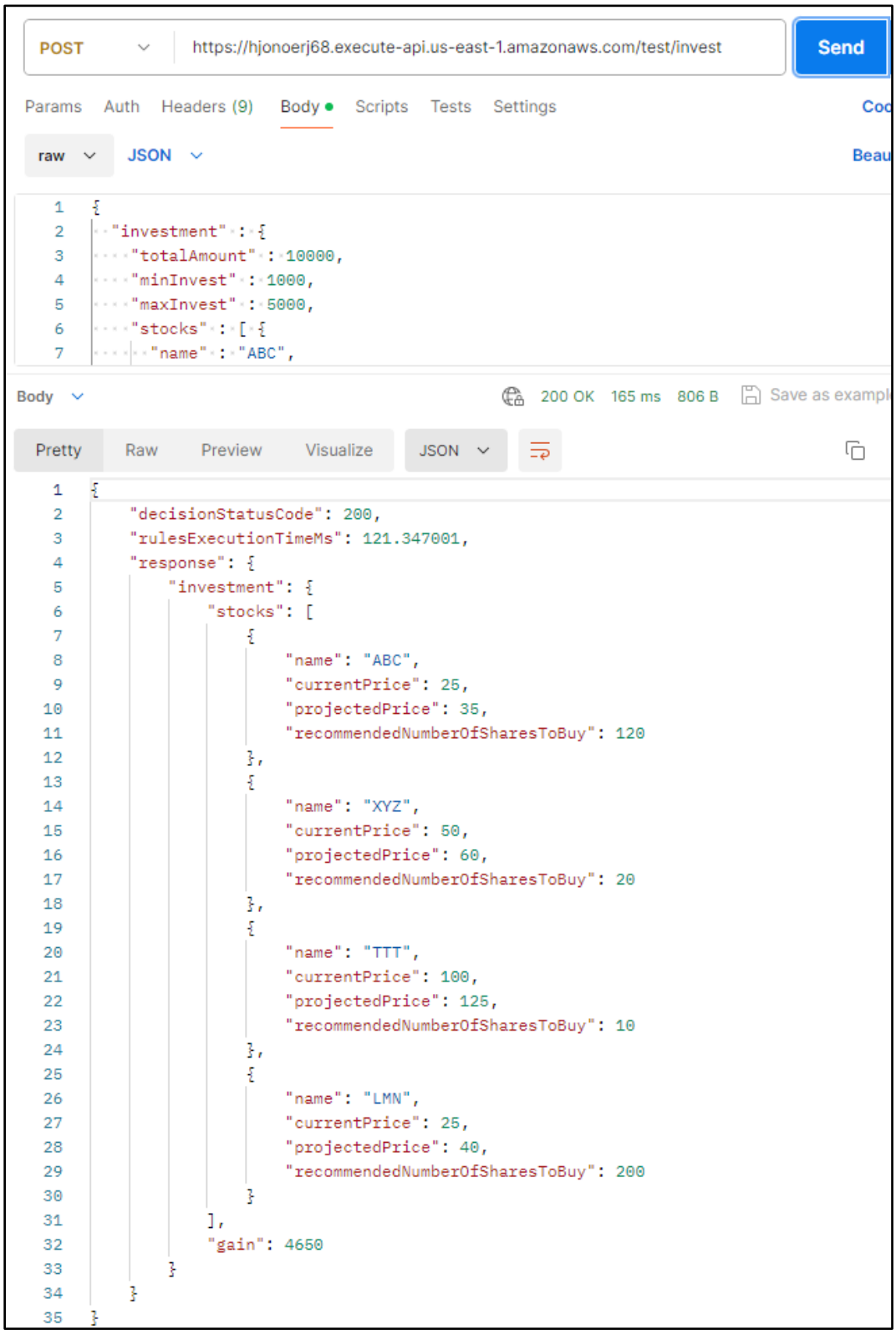
```
model.file="rules/DecisionModelInvest.xls"
report=On
trace=On

# deployment properties
deployment=aws-lambda
aws.lambda.bucket=openrules-demo-lambda-bucket
aws.api.stage=test
aws.lambda.region=us-east-1
```

I also copied all necessary dependencies from the standard project "VacationDaysLambda" to my the file "pom.xml". I also copied the batch file "VacationDaysLambda/deployLambda.bat" and ran it. Within 30 seconds it deployed my project to AWS cloud and gave me the endpoint URL https://hjonoerj68.execute-api.us-east-1.amazonaws.com/test/invest.

OpenRules also automatically converted my test case from Excel (see the beginning of this article) to JSON in the folder "target/jsons":


```
{
  "investment" : {
    "totalAmount" : 10000,
    "minInvest" : 1000,
    "maxInvest" : 5000,
    "stocks" : [ {
      "name" : "ABC",
      "currentPrice" : 25,
      "projectedPrice" : 35,
      "recommendedNumberOfSharesToBuy" : 0
    }, {
      "name" : "XYZ",
      "currentPrice" : 50,
      "projectedPrice" : 60,
      "recommendedNumberOfSharesToBuy" : 0
    }, {
      "name" : "TTT",
      "currentPrice" : 100,
      "projectedPrice" : 125,
      "recommendedNumberOfSharesToBuy" : 0
    }, {
      "name" : "LMN",
      "currentPrice" : 25,
      "projectedPrice" : 40,
      "recommendedNumberOfSharesToBuy" : 0
    } ],
    "gain" : 0
  }
}
```

Now I was ready to use POSTMAN to test my decision model. Here is my POSTMAN view:



As a result, I got a decision service from the AWS cloud that was accessible through a standard REST interface. This service is accessible from any computer. Importantly, the best outcomes were achieved with an overall gain of 4650 and an execution time of 165 milliseconds from start to finish.