

Trucks Dispatching Schedule

Overview

- Data Inputting
- Implementation Detail
 - Create Data
 - Build Model
 - Solve Model
- User Satisfaction Concern

Data Input

- Number of truck types
- Number of warehouses
- Number of trucks for each type
- Loading capacity for each type

Implementation – Strategy

```
Sub truckpickup()
```

```
Dim c As Range, numTruck As Range, numUnits As Range  
Dim TotalUnits As Integer 'An integer variable to store the total number of items from all warehouses  
Dim numType As Integer 'An integer variable to store the total number of types of trucks  
Dim numwarehouse As Integer 'An integer variable to store the total number of warehouses  
Dim Base As Range 'Set the upper left cell of the Matrix as the base cell
```

```
Call CreateData(c, numTruck, numUnits, TotalUnits, numType, numwarehouse)
```

```
Call buildModel(c, numTruck, numUnits, TotalUnits, numType, numwarehouse, Base)
```

```
Call SolveModel(Base, numTruck, numType, numwarehouse)
```

```
End Sub
```

Implementation – Create Data

- Range Variable
 - Number of trucks for each truck type
 - Number of unites available at each warehouse
 - Loading capacity for each truck type
- Variable Initialization
 - The matrix of variable (T_{ij}) , number of trucks of each type sending to each warehouse
 - Y_j , number of units picked up at each warehouse j

Implementation – Create Data

```
Sub CreateData(ByRef c As Range, ByRef numTruck As Range, ByRef numUnits As Range, _  
    ByRef TotalUnits As Integer, ByRef numType As Integer, ByRef numwarehouse As Integer)
```

```
'Get the number of types of trucks again for later use, Here we count the serial number generated  
'by Macro GetReady_ which will surely be correct. enabling us to use end(XlToRight).Columns.count  
numType = Range(Cells(2, 2), Cells(2, 2).End(xlToRight)).Columns.Count
```

```
'Get the total number of warehouses again for later use_Same as above  
numwarehouse = Range(Cells(12, 1), Cells(12, 1).End(xlDown)).Rows.Count
```

```
Set numTruck = Range(Cells(3, 2), Cells(3, 1 + numType)) 'Create a range for all types of trucks  
Set c = Range(Cells(4, 2), Cells(4, 1 + numType)) 'Create a range for all capacity of all types of trucks  
Set numUnits = Range(Cells(12, 2), Cells(11 + numwarehouse, 2)) 'Create a range for all items of all warehouses
```

Implementation – Create Data

'A nested loop to fill in 0 in the Matrix, So as to initialize

Dim W As Integer, T As Integer, Base2 As Range

Set Base2 = Cells(11, 7)

For W = 1 To numwarehouse

For T = 1 To numType + 2

Base2(W, T) = 0

Next T

Next W

Cells(13 + numwarehouse, 1) = "Total" 'Create a title

'Calculate the sum of all available items. Store the result in variable "TotalUnits"

For j = 1 To numwarehouse

TotalUnits = TotalUnits + Cells(11 + j, 2)

Next j

Cells(13 + numwarehouse, 2) = TotalUnits 'Display the total number of units in all warehouses

Implementation – Build Model

- Construct variable matrix
 - For number of trucks of each type sending to each warehouse
- Set formula
 - SUM to get Y_j , number of units picked up at each warehouse j
 - SUM to get T_i , total number of trucks dispatched for each truck type i
 - SUMPRODUCT to get $c_i * T_{ij}$, total capacity of trucks dispatched to each warehouse j
- Construct objective cell
 - SUM to get total Y as objective value

Implementation – Build Model

'Creates a two dimensional range

```
Set Base = Range(Cells(11, 7), Cells(11, 7).Offset(numwarehouse - 1, numType - 1))
```

'Put the formula for calculating the total capacity of trucks sent to each warehouse at the end of each row referring to each warehouse

Dim i As Integer

```
For i = 1 To numwarehouse
```

```
Base(i, numType + 2).Formula = "=Sumproduct(" & c.Address & "," & Range(Base(i, 1), Base(i, numType)).Address & ")"
```

```
Next i
```

'Put the formula for suming up T_{ij} under each column referring to each type of truck i

```
Base(numwarehouse + 1, 0) = "Each"
```

```
Base(numwarehouse + 2, 0) = "Total"
```

```
For j = 1 To numType
```

```
Base(numwarehouse + 2, j).Formula = "=SUM(" & Base(1, j).Address & ":" & Base(numwarehouse, j).Address & ")"
```

```
Next j
```

Implementation – Build Model

'Calculate the remaining items of each warehouse

Dim m As Integer

For m = 1 To numwarehouse

Cells(11 + m, 4).Formula = "=" & Cells(11 + m, 2).Address & "-" & Cells(10 + m, 7 + numType).Address & ")"

Next m

'Calculate the Obj.Func.

Base(numwarehouse + 4, numType) = "Objective :"

Base(numwarehouse + 4, numType + 1).Formula = "=SUM(" & Base(1, 1 + numType).Address _

& ":" & Base(numwarehouse, 1 + numType).Address & ")"

Implementation – Solve Model

- Record Solver solving process and Solver option setting
 - To ensure an integer solution within tolerance
 - To ensure a acceptable run time
- Generalize the cells' address so as to fit on our model

Implementation – Solve Model

```
Sub SolveModel(ByRef Base As Range, ByRef numTruck As Range, ByVal numType As Integer, ByVal numwarehouse As Integer)
    'Assign the Obj.Func.Cell
    Dim ObjectiveCell As Range
    Set ObjectiveCell = Base(numwarehouse + 4, numType + 1)

    'A range referring to all variable cells
    Dim Var As Range
    Set Var = Range(Base(1, 1), Base(1, 1).Offset(numwarehouse - 1, numType))

    'Number of each type truck i sent
    Dim Tij As Range
    Set Tij = Range(Base(numwarehouse + 2, 1), Base(numwarehouse + 2, 1).End(xlToRight))

    'Number of each type of truck available(Ti)
    Dim Ti As Range
    Set Ti = Range(numTruck(1, 1), numTruck(1, 1).End(xlToRight))

    'Number of trucks sent to warehouse j
    Dim Yj As Range
    Set Yj = Range(Base(1, numType + 1), Base(1, numType + 1).End(xlDown))

    'Number of available items in each warehouse
    Dim Dj As Range
    Set Dj = Range(numTruck(10, 1), numTruck(10, 1).End(xlDown))

    'Theoretical capacity
    Dim CiTij As Range
    Set CiTij = Range(Base(1, numType + 2), Base(1, numType + 2).End(xlDown))

    'Range for the Tij variables only
    Dim TijOnly As Range
    Set TijOnly = Range(Base(1, 1), Base(1, 1).Offset(numwarehouse - 1, numType - 1))
End Sub
```

Implementation – Solve Model

```
'Reset Solver  
SolverReset
```

```
'Set Objective Function Cell  
SolverOk SetCell:=ObjectiveCell.Address, MaxMinVal:=1, ValueOf:=0, ByChange:=Var.Address, _  
Engine:=2, EngineDesc:="Simplex LP"
```

```
'Number-of-Available-Trucks Constraints  
SolverAdd CellRef:=Tij.Address, Relation:=1, FormulaText:=Ti.Address
```

```
'Yj<=Dj  
SolverAdd CellRef:=Yj.Address, Relation:=1, FormulaText:=Dj.Address
```

```
'Yj<=CiTij  
SolverAdd CellRef:=Yj.Address, Relation:=1, FormulaText:=CiTij.Address
```

```
'Logical Constraints  
SolverAdd CellRef:=TijOnly.Address, Relation:=4, FormulaText:="integer"
```

```
'Set solver options  
SolverOptions MaxTime:=100, Iterations:=100, Precision:=1e-06, _  
Convergence:=0.0001, StepThru:=False, Scaling:=True, AssumeNonNeg:=True, _  
Derivatives:=1
```

```
'Call Solver  
infeasible = Application.Run("SolverSolve", True)  
If infeasible Then  
    MsgBox "An integer solution within tolerance is found. It is possible that better integer solutions exist"  
Else  
    MsgBox "Optimal solution found!"  
End If
```

```
'Do not display Solver Solution Box  
SolverSolve UserFinish:=True
```

```
'Keep the Optimal Solution  
SolverFinish KeepFinal:=1
```

```
' Clear unnecessary figures from the worksheet for a clean user interface(Clear away the CiTij column used  
CiTij.Select  
Selection.ClearContents  
CiTij(0, 1).Select  
Selection.ClearContents  
CiTij(-1, 1).Select  
Selection.ClearContents
```

User Satisfaction Concern

- Add in only two buttons to do ALL

[illegible]

User Satisfaction Concern

- Click “Start” button to fill in the number of trucks type

The image shows a software interface with a grid background. At the top, there are two buttons: "Start" and "Continue". Below the "Start" button, a modal dialog box is displayed. The dialog box has a title bar that says "ENTER NUMBER OF TYPES". Inside the dialog, the text reads "The number of types of trucks you have in depot now is :". To the right of this text are two buttons: "OK" (blue) and "Cancel" (white). Below the text is a text input field containing the number "6".

User Satisfaction Concern

- and the number of warehouses as well

The image shows a user interface with a grid background. At the top, there are two buttons: "Start" and "Continue". Below them, a dialog box titled "ENTER NUMBER OF WAREHOUSES" is displayed. The dialog box contains the text "How many of warehouses are there" and a text input field with the number "8" entered. To the right of the input field are two buttons: "OK" and "Cancel".

User Satisfaction Concern

- Then user will be asked to fill in all scheduling information

[illegible]

User Satisfaction Concern

- Then click “Continue” to obtain the result

[illegible]

User Satisfaction Concern

- Implementation Details
- Create another Macro to implement the “Start” button
 - Display the format to facilitate user fill in information
 - Include code for clearing all previous content before user start
- Close solver outcome box and inform user the solution details
 - If the solver can get optimal solution: “Optimal Solution Found”
 - Otherwise, “An integer solution found with tolerance”
- All information displayed is necessary and sufficient
- Clean and Concise