

Restore the average-distance Solver model

Restore this model. Remove the distance-bound and shortfall constraints used for the compromise.

Solver Parameters

Set Objective:

To: ☐ Max ☒ Min

By Changing Variable Cells:

Subject to the Constraints:

Solving Method:

Only **B7:C7** change during a solve. The macro writes the current limit into **B8**.

Keep the same GRG accuracy settings.

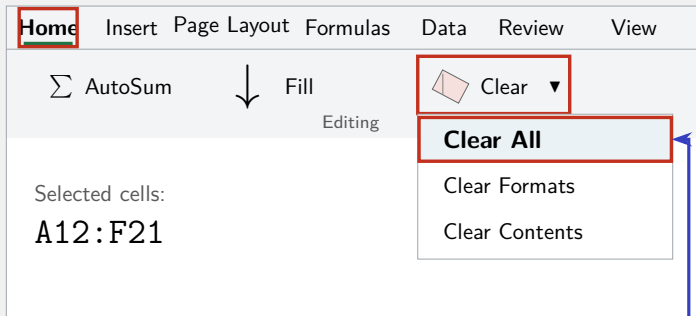
Select the cells to clear

A12:F21		f_x				
	A	B	C	D	E	F
11	Maximum (km)					
12						
13	Longest distance bound					
14	Objective for distance					
15						
16	Objective	Current	Best	Shortfall		
17	Average					
18	Maximum bound					
19						
20	Shortfall bound q					
21	Objective for shortfall					

Select A12:F21.

Reuse the old z , q and shortfall area; keep row 11.

Clear the old contents and formats



Home → Clear → Clear All.

Clear All removes the old formulas, values and percentage formats.

Label the result columns

A12:F12		fx				
	A	B	C	D	E	F
11	Maximum (km)					
12	Limit (km)	x	y	Average	Maximum	Solver
13						
14						
⋮						
26						



In A12:F12, enter the headings shown.

Leave **A13:F26** empty for the **14 results**, one per connection limit.

Select the numerical result cells

B13:E26		fx				
	A	B	C	D	E	F
11	Maximum (km)					
12	Limit (km)	x	y	Average	Maximum	Solver
13						
14						
⋮						
26						

Select B13:E26.

Cmd + 1 opens **Format Cells**.

These columns will store the coordinates and distances.

Use four decimal places

Format Cells

Number Alignment Font Border Fill Protection

Category:

- General
- Number**
- Currency
- Accounting
- Date

Decimal places: 4

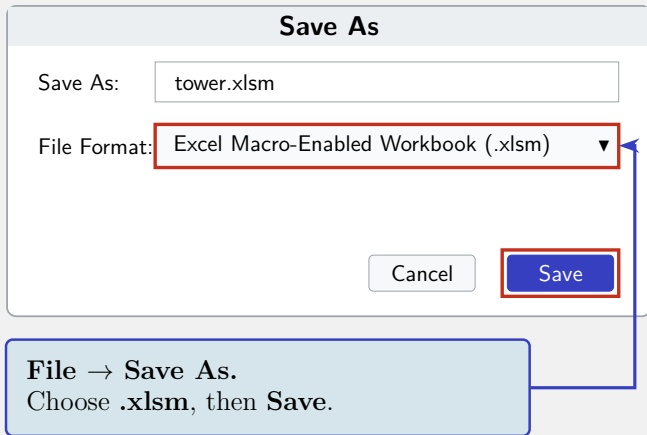
☐ Use 1000 Separator (,)

Cancel OK

Choose **Number**, set **4** decimal places, then **OK**.

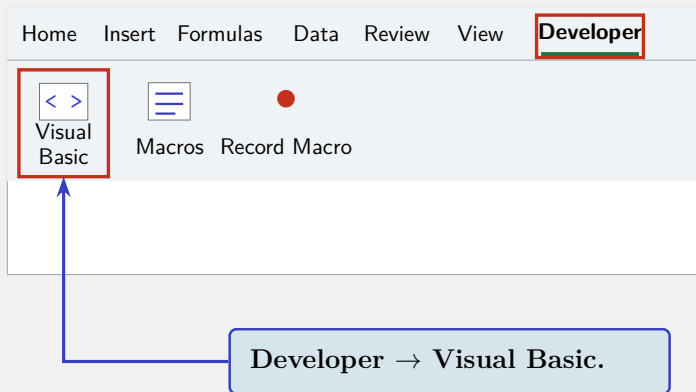
B13:E26: display numbers with four decimal places.

Save a workbook that can store VBA



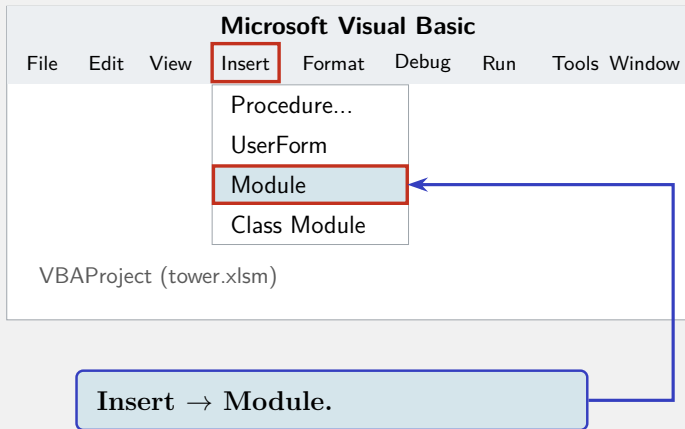
The **.xlsm** format keeps the macro together with the worksheet.

Open the VBA editor



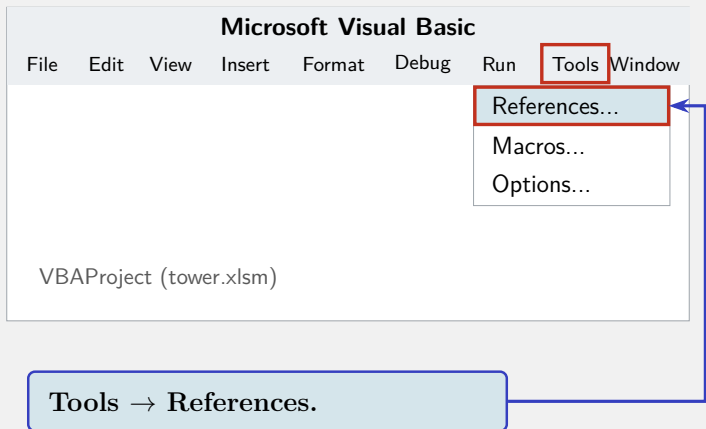
The VBA editor is where you will enter the macro.

Create a module for the macro



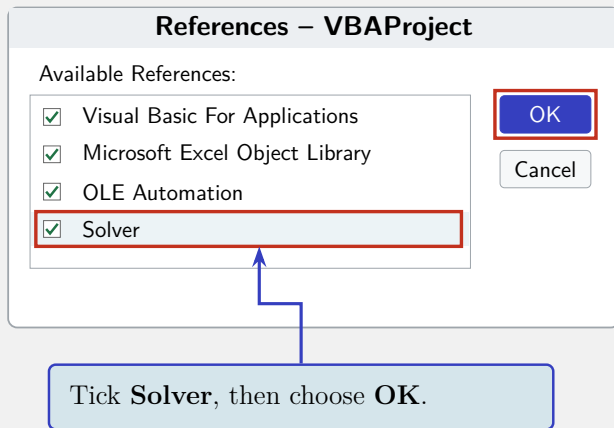
A module provides a blank page for the VBA instructions.

Open the VBA references



Here you can make the Solver commands available to the macro.

Enable Solver commands in VBA



The macro can now call **SolverSolve** and **SolverFinish**.

Name the macro and its stored numbers

```
Sub CompareTowerRange()
    Dim limit As Long, r As Long, result As Integer

End Sub
```

Sub begins the macro. Select this name from **Developer** → **Macros** when running the completed code.

limit stores the allowed kilometres. **r** stores the result row. **result** stores Solver's reply as a number.

Do not run yet: we are still adding the instructions.

Clear the table and choose a starting location

```
Sub CompareTowerRange()  
    Dim limit As Long, r As Long, result As Integer  
    Range("A13:F26").ClearContents  
    Range("B7").Value = 27  
    Range("C7").Value = 30
```

ClearContents removes the old answers from A13:F26. The headings in row 12 stay.

Start at (27, 30).
Its longest link is
26.9258 km, so
it respects even
the first limit of 27
km.

End Sub

Do not run yet: we are still adding the instructions.

Repeat the solve for each connection limit

```
Sub CompareTowerRange()  
    Dim limit As Long, r As Long, result As Integer  
    Range("A13:F26").ClearContents  
    Range("B7").Value = 27  
    Range("C7").Value = 30  
    For limit = 27 To 40
```

For uses 27, 28, 29, . . . , 40. With no Step written, it adds 1 each time. There are **14 solves**.

Next limit returns to the loop with the next number. After 40, the loop ends.

```
Next limit ◀
End Sub
```

Do not run yet: we are still adding the instructions.

Put the current limit into the worksheet

```
Sub CompareTowerRange()  
    Dim limit As Long, r As Long, result As Integer  
    Range("A13:F26").ClearContents  
    Range("B7").Value = 27  
    Range("C7").Value = 30  
    For limit = 27 To 40  
        Range("B8").Value = limit  
        Calculate  
    Next limit  
End Sub
```

Write the current limit into **B8**. Solver keeps it fixed while choosing the coordinates.

Calculate updates formulas in the open workbooks. The sheet is ready for this solve.

Do not run yet: we are still adding the instructions.

Run Solver without a results window

```
Sub CompareTowerRange()  
    Dim limit As Long, r As Long, result As Integer  
    Range("A13:F26").ClearContents  
    Range("B7").Value = 27  
    Range("C7").Value = 30  
    For limit = 27 To 40  
        Range("B8").Value = limit  
        Calculate  
        result = SolverSolve(UserFinish:=True)  
    Next limit  
End Sub
```

SolverSolve is like clicking Solve. It uses the model already entered: minimize B10 by changing B7:C7.

UserFinish:=True hides the final results window. The loop can continue without waiting for a click. The number returned goes into **result**.

Do not run yet: we are still adding the instructions.

Stop and show the limit if Solver has a problem

```
Sub CompareTowerRange()  
    Dim limit As Long, r As Long, result As Integer  
    Range("A13:F26").ClearContents  
    Range("B7").Value = 27  
    Range("C7").Value = 30  
    For limit = 27 To 40  
        Range("B8").Value = limit  
        Calculate  
        result = SolverSolve(UserFinish:=True)  
        If result < 0 Or result > 2 Then  
            MsgBox "Check Solver for limit " & limit  
            Exit Sub  
        End If  
    Next limit  
End Sub
```

Continue for replies **0, 1 or 2**. They report that the constraints are satisfied. We explain and check these replies after the full macro.

For any other reply, **MsgBox** shows which limit needs checking. **Exit Sub** stops the whole macro. Earlier answers remain in the table.

Do not run yet: we are still adding the instructions.

Keep the location and update its distances

```
Sub CompareTowerRange()  
    Dim limit As Long, r As Long, result As Integer  
    Range("A13:F26").ClearContents  
    Range("B7").Value = 27  
    Range("C7").Value = 30  
    For limit = 27 To 40  
        Range("B8").Value = limit  
        Calculate  
        result = SolverSolve(UserFinish:=True)  
        If result < 0 Or result > 2 Then  
            MsgBox "Check Solver for limit " & limit  
            Exit Sub  
        End If  
        SolverFinish KeepFinal:=1  
        Calculate  
    Next limit  
End Sub
```

KeepFinal:=1 keeps the coordinates selected by Solver. **Calculate** then updates the distances before we record them.

The next solve starts from this location. Its limit is larger, so the previous location is still allowed.

Do not run yet: we are still adding the instructions.

Choose the row and record the tested limit

```
Sub CompareTowerRange()  
    Dim limit As Long, r As Long, result As Integer  
    Range("A13:F26").ClearContents  
    Range("B7").Value = 27  
    Range("C7").Value = 30  
    For limit = 27 To 40  
        Range("B8").Value = limit  
        Calculate  
        result = SolverSolve(UserFinish:=True)  
        If result < 0 Or result > 2 Then  
            MsgBox "Check Solver for limit " & limit  
            Exit Sub  
        End If  
        SolverFinish KeepFinal:=1  
        Calculate  
        r = 13 + (limit - 27)  
        Cells(r, 1).Value = limit  
    Next limit  
End Sub
```

13 is the first result row. **limit - 27** counts the steps after 27 km.

27 km → row 13
28 km → row 14
:
40 km → row 26.

Cells(r, 1) means row r, column A. Write the current allowed limit there.

Do not run yet: we are still adding the instructions.

Copy the two coordinates into adjacent cells

```
Sub CompareTowerRange()  
    Dim limit As Long, r As Long, result As Integer  
    Range("A13:F26").ClearContents  
    Range("B7").Value = 27  
    Range("C7").Value = 30  
    For limit = 27 To 40  
        Range("B8").Value = limit  
        Calculate  
        result = SolverSolve(UserFinish:=True)  
        If result < 0 Or result > 2 Then  
            MsgBox "Check Solver for limit " & limit  
            Exit Sub  
        End If  
        SolverFinish KeepFinal:=1  
        Calculate  
        r = 13 + (limit - 27)  
        Cells(r, 1).Value = limit  
        Cells(r, 2).Resize(1, 2).Value = _  
            Range("B7:C7").Value  
    Next limit  
End Sub
```

Cells(r, 2) starts at column B. **Resize(1, 2)** selects one row and two columns: B and C in row r.

Copy the two numbers from B7:C7 into those two cells. The **underscore** continues the same instruction on the next line.

Do not run yet: we are still adding the instructions.

Save both distances and the Solver reply

```
Sub CompareTowerRange()  
    Dim limit As Long, r As Long, result As Integer  
    Range("A13:F26").ClearContents  
    Range("B7").Value = 27  
    Range("C7").Value = 30  
    For limit = 27 To 40  
        Range("B8").Value = limit  
        Calculate  
        result = SolverSolve(UserFinish:=True)  
        If result < 0 Or result > 2 Then  
            MsgBox "Check Solver for limit " & limit  
            Exit Sub  
        End If  
        SolverFinish KeepFinal:=1  
        Calculate  
        r = 13 + (limit - 27)  
        Cells(r, 1).Value = limit  
        Cells(r, 2).Resize(1, 2).Value = _  
            Range("B7:C7").Value  
        Cells(r, 4).Value = Range("B10").Value  
        Cells(r, 5).Value = Range("B11").Value  
        Cells(r, 6).Value = result  
    Next limit  
End Sub
```

Copy the average into column D, the maximum into E and Solver's reply into F. **.Value** stores the numbers, so later solves do not overwrite these answers.

One row now contains one complete result. **Next limit** moves on to the next optimization.

The macro is complete. The next slide gives the full code to copy and run.

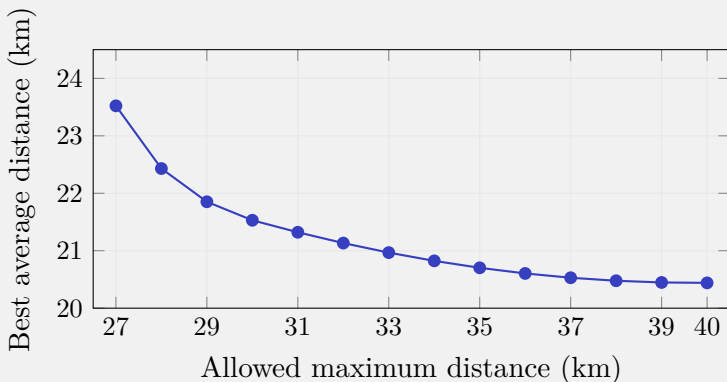
Run the complete macro

```
Sub CompareTowerRange()  
    Dim limit As Long, r As Long, result As Integer  
    Range("A13:F26").ClearContents  
    Range("B7").Value = 27  
    Range("C7").Value = 30  
    For limit = 27 To 40  
        Range("B8").Value = limit  
        Calculate  
        result = SolverSolve(UserFinish:=True)  
        If result < 0 Or result > 2 Then  
            MsgBox "Check Solver for limit " & limit  
            Exit Sub  
        End If  
        SolverFinish KeepFinal:=1  
        Calculate  
        r = 13 + (limit - 27)  
        Cells(r, 1).Value = limit  
        Cells(r, 2).Resize(1, 2).Value = _  
            Range("B7:C7").Value  
        Cells(r, 4).Value = Range("B10").Value  
        Cells(r, 5).Value = Range("B11").Value  
        Cells(r, 6).Value = result  
    Next limit  
End Sub
```

Run **CompareTowerRange**. Check that all **14** rows are filled.

More allowed range can reduce the best average

Create an **XY scatter chart with lines and markers**. Use **A13:A26** for x and **D13:D26** for y ; add the series by selecting these two ranges separately.



A larger limit allows every previous location and possibly new ones. The best average therefore **cannot increase**.

The Dunder Mifflin Company problem

Open **The Dunder Mifflin Company problem** on Moodle.



6 research projects

25 engineers

\$1.7 million to invest

The Dunder Mifflin Company problem

Dunder Mifflin: allocate the research budget

Dunder Mifflin must allocate next year's research and development budget among **six proposed projects**.

The company can hire up to **25 engineers** and spend up to **\$1.7 million** on the selected projects' fixed start-up costs.

With x_i engineers, project i succeeds with probability

$$p_i = \frac{x_i}{x_i + \epsilon_i},$$

where $\epsilon_i > 0$ is a given project-specific parameter. More engineers increase the chance of success.

Each project's **expected net present value (NPV)** is its NPV if successful multiplied by p_i . Failure contributes zero.

Choose projects and team sizes to **maximize total expected NPV**. Assign engineers only to funded projects.

Choose the projects and their teams

**25 engineers
available**

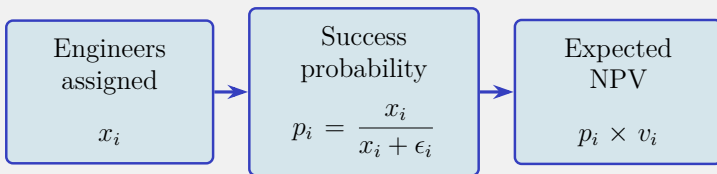
**Start-up budget
\$1.7 million**

Project	Start-up cost	NPV if successful	ϵ_i
1	325	750	3.1
2	200	120	2.5
3	490	900	4.5
4	125	400	5.6
5	710	1100	8.2
6	240	800	8.5

Choose which projects to fund
and how many engineers to assign.

Maximize total expected NPV. Table amounts: \$1,000.

From engineers to expected NPV



ϵ_i is given for each project. For a fixed positive team size, a smaller ϵ_i means a higher chance of success.

v_i is the NPV if successful; failure gives zero.

No engineers: zero success probability and zero expected NPV.

Calculate the expected NPV of project 1

6 engineers

$$\epsilon_1 = 3.1$$

NPV if successful

750 (\$1,000)

$$p_1 = \frac{6}{6 + 3.1} = 65.9341\%$$

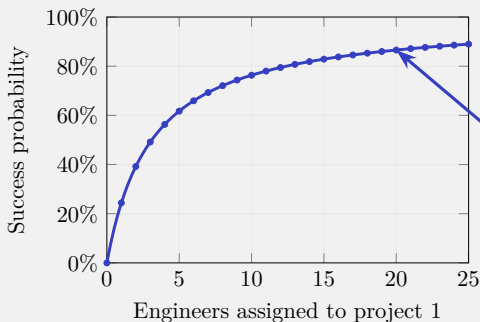
Expected NPV

$$\frac{6}{9.1} \times 750 = 494.5055$$

Approximately **\$494,505.49.**

Expected NPV weights the successful outcome by its probability.

Why use GRG Nonlinear?



Extra engineers improve the chance of success, but each increase is smaller.

The formula is smooth and **nonlinear**.

Use GRG Nonlinear with integer constraints on team sizes.

Create the Dunder Mifflin worksheet

	A	B	C	D	E
1					
2					
3					
< > Dunder Mifflin +					

Then double-click
the tab and type:
Dunder Mifflin.

Click + to add a
worksheet.

Use this worksheet for the project choices, formulas and Solver model.

Label the project choices

A1:D1	<i>fx</i>	Project		
	A	B	C	D
1	Project	Fund?	Engineers	Link

Enter these headings in A1:D1.

B and C will hold the decisions; D will link the two decisions.

Label the probability and financial columns

E1:I1		fx			
	E	F	G	H	I
1	€	Success probability	NPV if successful	Expected NPV	Start-up cost

Enter these headings in E1:I1.

E, G and I hold data; F and H will contain formulas.

Enter the project numbers and ϵ

A2:A7	f_x	
	A	
1	Project	
2	1	
3	2	
4	3	
5	4	
6	5	
7	6	

E2:E7	f_x	
	E	
1	ϵ	
2	3.1	
3	2.5	
4	4.5	
5	5.6	
6	8.2	
7	8.5	

Enter the values shown
in A2:A7 and E2:E7.

ϵ is the given number used in the success probability formula.

Enter project values and start-up costs

G2:G7		fx			
	G		H		I
1	NPV if successful		Expected NPV		Start-up cost
2	750				325
3	120				200
4	900				490
5	400				125
6	1100				710
7	800				240

Enter the values shown
in **G2:G7** and **I2:I7**.

All amounts are in **\$1,000**; for example, 750 means \$750,000.

Enter the available engineers and budget

B9:C9		fx	
	B	C	
7			
8			
9	Available	25	

H9:I9		fx	
	H	I	
7		240	
8			
9	Budget	1700	

Enter the labels and limits:

B9: Available C9: 25

H9: Budget I9: 1700

Row 8 will hold totals; row 9 gives the available resources.

Initialize the project decisions

B2:C7		fx	0
	A	B	C
1	Project	Fund?	Engineers
2	1	0	0
3	2	0	0
4	3	0	0
5	4	0	0
6	5	0	0
7	6	0	0

Enter 0 in B2:C7.
Solver changes these 12 cells.

Fund?: 1 = yes; 0 = no.

Engineers: whole numbers, zero or more.

Solver chooses the funded projects and their teams.

Why do we need the Link column?

B2:C2		fx	
	A	B	C
1	Project	Fund?	Engineers
2	1	0	6

Six engineers are assigned to a project that **has not** been funded.

The budget and staffing limits alone do not prevent this.

We must explicitly require that **only funded projects can receive engineers**.

Link the team size to project funding

D2	<i>fx</i>	=C2-\$C\$9*B2		
	B		C	D
1	Fund?		Engineers	Link
2	0		0	0

In **D2**, enter =C2-\$C\$9*B2.

\$C\$9 fixes the reference to the available **25 engineers**.

Require $\text{Link} = \text{Engineers} - 25 \times \text{Fund?} \leq 0$ in Solver.

Fund? = 0

$0 \leq \text{Engineers} \leq 0$.

No engineers assigned.

Fund? = 1

$0 \leq \text{Engineers} \leq 25$.

Engineers may be assigned.

The 25 engineers are shared across **all projects**.

Link makes the team-size decision depend on the funding decision.

Copy the funding constraint to every project

D2:D7		<i>fx</i>	=C2-\$C\$9*B2	
	B		C	D
1	Fund?		Engineers	Link
2	0		0	0
3	0		0	0
4	0		0	0
5	0		0	0
6	0		0	0
7	0		0	0
8				
9	Available		25	

Copy D2 down to D7.
Solver will require **D2:D7** $\leq$ 0.

With $D \leq 0$: no funding forces zero engineers; funding allows up to 25.

Calculate each probability of success

F2	f_x	$=C2/(C2+E2)$		
	C	D	E	F
1	Engineers	Link	€	Success probability
2	0	0	3.1	0
3	0	0	2.5	0
4	0	0	4.5	0
5	0	0	5.6	0
6	0	0	8.2	0
7	0	0	8.5	0

F2: $=C2/(C2+E2)$
Copy F2 down to F7.

More engineers raise success probability; with none, it is zero.

Display the probabilities as percentages

Select F2:F7.
Press **Cmd+1**.

Format Cells

Number	Alignment	Font	Border	Fill
Category:	Sample			
General	0.00%			
Number				
Currency	Decimal places: 2			
Percentage				
Cancel OK				

F2:F7	
	F
1	Success probability
2	0.00%
3	0.00%
4	0.00%
5	0.00%
6	0.00%
7	0.00%

Percentage
2 decimal places
Click **OK**.

Only the display changes; calculations keep full precision.

Calculate each expected NPV

H2	<i>fx</i>	=F2*G2	
	F	G	H
1	Success probability	NPV if successful	Expected NPV
2	0.00%	750	0
3	0.00%	120	0
4	0.00%	900	0
5	0.00%	400	0
6	0.00%	1100	0
7	0.00%	800	0

H2: =F2*G2
Copy H2 down to H7.



No extra Fund? multiplier: unfunded projects already give zero.

Display expected NPVs with two decimals

Select H2:H7.
Press **Cmd+1**.

Format Cells	
<u>N</u> umber	A
Category:	Sample
General	0.00
Number	
Currency	
Percentage	
Decimal places:	2
Cancel OK	

H2:H7	
	H
1	Expected NPV
2	0.00
3	0.00
4	0.00
5	0.00
6	0.00
7	0.00

Number
2 decimal places
Click **OK**.

Amounts remain in **\$1,000**; only their display changes.

Label the totals and the remaining budget

B8		<i>fx</i>	Used						
	B		C	D	E	F	G	H	I
8	Used					Totals			
9	Available		25					Budget	1700
10								Funds left	

In B8, enter
Used

In F8, enter
Totals

In H10, enter
Funds left

These labels identify the totals and the unused budget.

Add the engineers assigned to all projects

C8	<i>fx</i>	=SUM(C2:C7)
	B	C
1	Fund?	Engineers
2	0	0
3	0	0
4	0	0
5	0	0
6	0	0
7	0	0
8	Used	0
9	Available	25

In C8, enter

=SUM(C2:C7)

Solver will keep the total in C8 at or below the 25 available in C9.

Add the expected NPVs

H8	<i>fx</i>	=SUM(H2:H7)		
	F	G	H	I
7	0.00%	800	0.00	240
8	Totals		0.00	
9			Budget	1700
10			Funds left	

In H8, enter

=SUM(H2:H7)

H8 is the objective: Solver will maximize total expected NPV.

Calculate the start-up funds used

I8	<i>fx</i>	=SUMPRODUCT(B2:B7,I2:I7)		
	F	G	H	I
7	0.00%	800	0.00	240
8	Totals		0.00	0
9			Budget	1700
10			Funds left	

In I8, enter

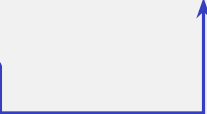
=SUMPRODUCT(B2:B7,I2:I7)

Only projects with **Fund?** = 1 contribute their start-up cost.

Calculate the funds left

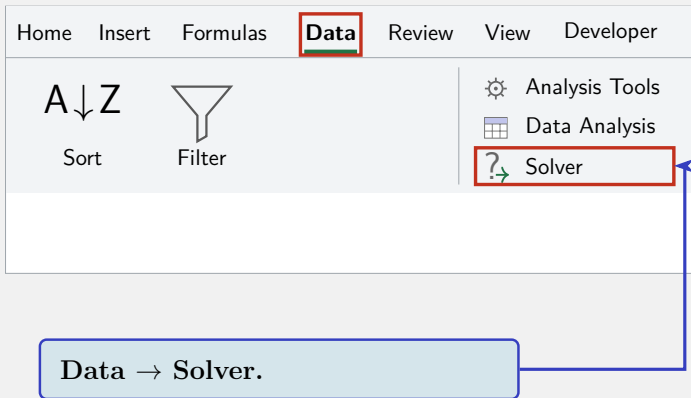
I10	<i>fx</i>	=I9-I8		
	F	G	H	I
7	0.00%	800	0.00	240
8	Totals		0.00	0
9			Budget	1700
10			Funds left	1700

In I10, enter
=I9-I8



Funds left equal the budget minus the start-up funds used.

Open Solver for the project model



Use the **Dunder Mifflin** worksheet.

Set the objective and the decision cells

Solver Parameters

Set Objective:

To: ☒ Max ☐ Min ☐ Value Of:

By Changing Variable Cells:

Maximize H8; changing cells: B2:C7.
B2:B7: Fund? C2:C7: Engineers.

Solver chooses both the funded projects and their team sizes.

Require a yes or no decision for each project

Add Constraint

Cell Reference:		Constraint:
\$B\$2:\$B\$7	bin▼ ≤ = ≥ int bin dif	binary
OK	Cancel	Add

Click **Add** in Solver.
B2:B7 → **bin** → **Add**.

bin restricts each Fund? cell to **0** or **1**.

Require whole numbers of engineers

Add Constraint

Cell Reference:		Constraint:
\$C\$2:\$C\$7	int▼ ≤ = ≥ int bin dif	integer
OK	Cancel	Add

First, set **By Changing Variable Cells** to **B2:C7**.

Then add **C2:C7** → **int** → **Add**.

C2:C7 must be decision cells before you can require integer values.

Assign engineers only to funded projects

Add Constraint

Cell Reference:	Constraint:
\$D\$2:\$D\$7	$\leq$ ▼ 0

OK Cancel Add

Add **D2:D7** $\leq$ 0.

The diagram illustrates the process of adding a constraint in Excel. It shows a dialog box titled 'Add Constraint' with two main sections: 'Cell Reference' and 'Constraint'. The 'Cell Reference' field contains '\$D\$2:\$D\$7'. The 'Constraint' field contains a less-than-or-equal-to symbol followed by a dropdown arrow and the value '0'. Below these fields are three buttons: 'OK', 'Cancel', and 'Add'. The 'Add' button is highlighted with a red border. A blue arrow points from the 'Add' button to a light blue box below the dialog, which contains the text 'Add **D2:D7** $\leq$ 0.'

An unfunded project must have zero engineers.

Limit the engineers and the start-up spending

Add Constraint

Cell Reference:	Constraint:
\$C\$8	\$C\$9
≤	▼

Add Constraint

Cell Reference:	Constraint:
\$I\$8	\$I\$9
≤	▼

Add **C8** ≤ **C9**; then **I8** ≤ **I9**.

At most **25 engineers** and **\$1.7 million** in start-up funds.

Choose GRG Nonlinear and nonnegative decisions

Solver Parameters

Set Objective: ☒ Max

By Changing Variable Cells:

Subject to the Constraints:

\$B\$2:\$B\$7 = binary
\$C\$2:\$C\$7 = integer
\$D\$2:\$D\$7 <= 0
\$C\$8 <= \$C\$9
\$I\$8 <= \$I\$9

Add

Change

Delete

☒ Make Unconstrained Variables Non-Negative

Solving Method:

Tick **Non-Negative**.
Choose **GRG Nonlinear**; open **Options**.

GRG handles the nonlinear success-probability formulas.

Keep the integer restrictions and tighten accuracy

Options

All MethodsGRG NonlinearEvolutionary

Constraint Precision:0.000001

☒ Use Automatic Scaling

Solving with Integer Constraints

☐ Ignore Integer Constraints

Integer Optimality (%):0

Keep **Ignore Integer Constraints** unchecked.
Use the other settings shown.

A positive percentage can stop the integer search early.

Set the GRG stopping accuracy

The screenshot shows the 'Options' dialog box for the Solver. Under the 'All Methods' section, 'GRG Nonlinear' is selected. The 'Convergence' value is set to 0.00000001. The 'OK' button is highlighted with a red box, and a blue arrow points from the text box below to it.

Options		
All Methods	GRG Nonlinear	Evolutionary
Convergence:	0.00000001	
Cancel		OK

GRG Nonlinear: set Convergence to 0.00000001. Then OK.

A smaller value asks GRG to refine the solution further.

Run Solver and keep its solution

The diagram illustrates the steps to run the Solver and keep its solution. It features two dialog boxes: 'Solver Parameters' and 'Solver Results'. In the 'Solver Parameters' dialog, the 'Solve' button is highlighted with a red box. A blue arrow points from this button to the 'Keep Solver Solution' option in the 'Solver Results' dialog, which is also highlighted with a red box. Another blue arrow points from the 'Keep Solver Solution' option to a light blue instruction box. In the 'Solver Results' dialog, the 'OK' button is also highlighted with a red box. The instruction box contains the text: 'Click **Solve**. When finished, choose **Keep Solver Solution** → **OK**.'

Solver Parameters

Solve

Solver Results

☒ Keep Solver Solution

☐ Restore Original Values

OK

Click **Solve**. When finished, choose **Keep Solver Solution** → **OK**.

The selected projects and team sizes remain in **B2:C7**.

The best project selection and allocation

Project	Fund?	Engineers	Success probability	Expected NPV
1	1	6	65.9341%	494.5055
2	0	0	0%	0
3	1	7	60.8696%	547.8261
4	1	3	34.8837%	139.5349
5	1	9	52.3256%	575.5814
6	0	0	0%	0
Total		25		1757.4479

Total expected NPV: **\$1,757,447.86.**

Fund projects 1, 3, 4 and 5; assign all 25 engineers.

Check the engineers and the budget

	B	C	H	I
8	Used	25	1757.4479	1650
9	Available	25	Budget	1700
10			Funds left	50

$$325 + 490 + 125 + 710 = 1650$$

Start-up costs stay below the **1700** budget.

Amounts in the worksheet are in \$1,000: 50 means \$50,000 remaining.

Could GRG stop at a worse local maximum?

In general, yes.

Nonlinear objectives may have several peaks.

In this problem

Expected NPV rises
and flattens as
the team grows.

**Allow fractions
temporarily**

Concave objective.
Linear constraints.

$$f_i(x_i) = \frac{v_i x_i}{x_i + \epsilon_i}, \quad f_i''(x_i) = \frac{-2v_i \epsilon_i}{(x_i + \epsilon_i)^3} < 0.$$

v_i : NPV if successful; ϵ_i : fixed project parameter.

Here, a continuous local maximum is also a global maximum.

Is the global optimum unique?

Yes: the best allocation is unique.

	Engineers in projects 1–6	Expected NPV (\$)
Best	6, 0, 7, 3, 9, 0	1,757,447.86
Next best	5, 0, 7, 3, 10, 0	1,754,719.54

The next-best plan earns
\$2,728.32 less in expected NPV.

An unused engineer could improve a funded project at no extra start-up cost.

Therefore a best plan uses all 25 engineers.

Uniqueness was checked independently, not inferred from GRG.

Network flows with Excel Solver

Build and solve networks in Excel.

Car shipments

Choose quantities.

Aid deliveries

Respect capacities.

Navigation

Choose a route.

Excel Solver

The Bavarian Motor Company problem

The **Bavarian Motor Company (BMC)** manufactures expensive luxury cars in **Hamburg, Germany**, and exports them for sale in **Italy**.

The cars are shipped from Hamburg to the ports of **Genova** and **Trieste**.

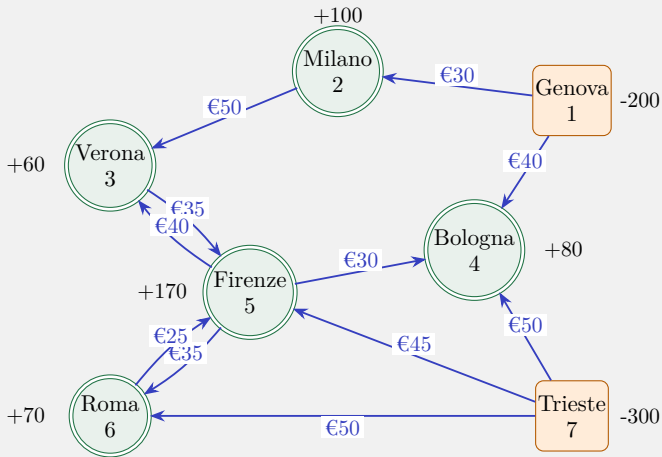
From these ports, the cars are transported **by rail or truck** to distributors in **Milano, Verona, Bologna, Firenze and Roma**.



Meet distributor orders at minimum transport cost.

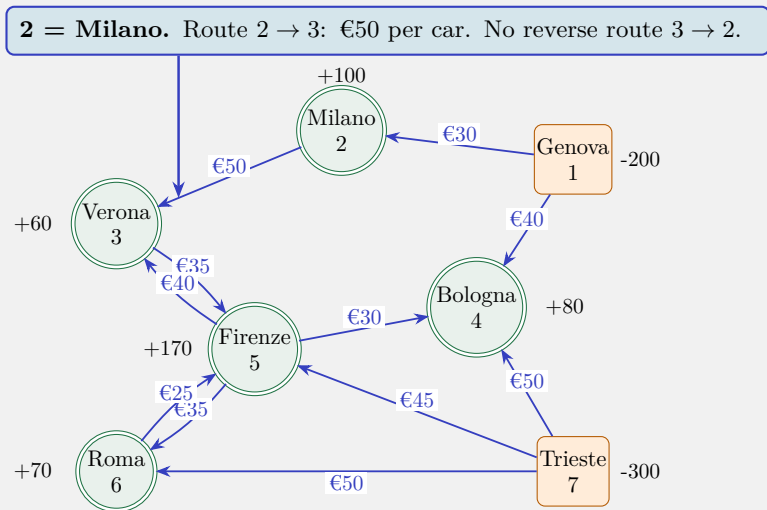
BMC: deliver cars to the distributors

Orange ports supply cars. Green cities require cars.



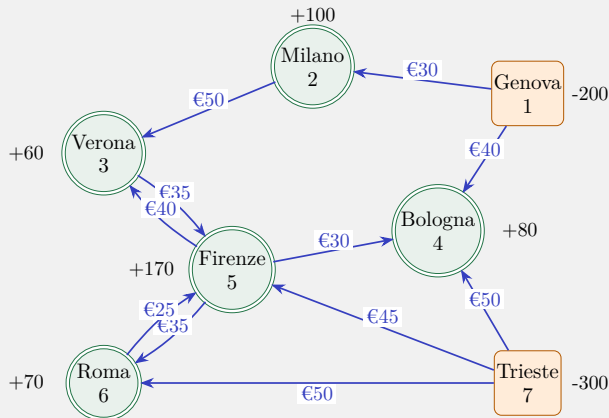
Meet all distributor orders at minimum cost. A green city may keep the cars it needs and forward other cars along its outgoing arrows.

Read the nodes and directed routes



A car may travel only in the direction shown by an arrow.

Read supply and demand on the graph



Negative = available cars: Genova -200, Trieste -300.
Positive = cars required: Milano needs 100 cars.

Total supply is 500 cars; total demand is 480. The least-cost plan must satisfy all five distributors.

Prepare the complete shipment worksheet

fx

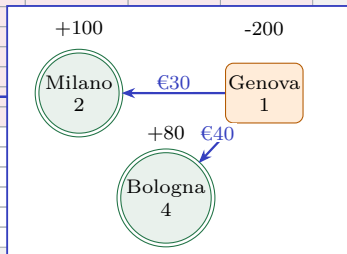
	A	B	C	D	E	F	G	H	I	J	K	L
1	BMC: car distribution											
2												
3	Ship	From			To		Cost (€)	Node			Net flow	Required
4												
5												
6												
7												
8												
9												
10												
11												
12												
13												
14												
15												
16						Total cost						

Create both tables on one worksheet, with these headings.

The left table records routes and shipment quantities. The right table records the balance at each city.

Translate the Genova routes into table rows

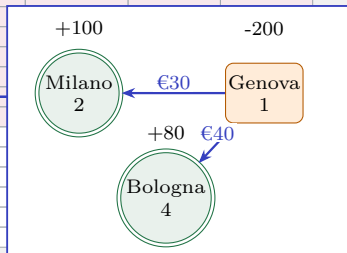
fx												
	A	B	C	D	E	F	G	H	I	J	K	L
1	BMC: car distribution											
2												
3	Ship		From			To	Cost (€)			Node	Net flow	Required
4			1	Genova	2	Milano	30					
5			1	Genova	4	Bologna	40					
6												
7												
8												
9												
10												
11												
12												
13												
14												
15												
16						Total cost						



Enter one row per arrow: origin, destination and cost per car.

Choose the shipment quantities

fx												
	A	B	C	D	E	F	G	H	I	J	K	L
1	BMC: car distribution											
2												
3	Ship			From		To	Cost (€)			Node	Net flow	Required
4	0	1		Genova	2	Milano	30					
5	0	1		Genova	4	Bologna	40					
6	0											
7	0											
8	0											
9	0											
10	0											
11	0											
12	0											
13	0											
14	0											
15												
16						Total cost						



Enter 0 in **B4:B14**. Solver will choose these shipment quantities.

Add the Milano route

fx												
	A	B	C	D	E	F	G	H	I	J	K	L
1	BMC: car distribution											
2												
3	Ship		From		To	Cost (€)				Node	Net flow	Required
4	0	1	Genova	2	Milano	30						
5	0	1	Genova	4	Bologna	40						
6	0	2	Milano	3	Verona	50						
7	0											
8	0											
9	0											
10	0											
11	0											
12	0											
13	0											
14	0											
15												
16						Total cost						

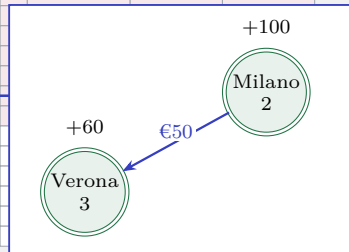
+100

Milano
2

+60

Verona
3

€50



Milano → Verona becomes row 6: **2, Milano, 3, Verona, 50.**

Give opposite directions separate rows

fx												
	A	B	C	D	E	F	G	H	I	J	K	L
1	BMC: car distribution											
2												
3	Ship		From		To	Cost (€)				Node	Net flow	Required
4	0	1	Genova	2	Milano	30						
5	0	1	Genova	4	Bologna	40						
6	0	2	Milano	3	Verona	50						
7	0	3	Verona	5	Firenze	35						
8	0	5	Firenze	3	Verona	40						
9	0											
10	0											
11	0											
12	0											
13	0											
14	0											
15												
16						Total cost						

+60

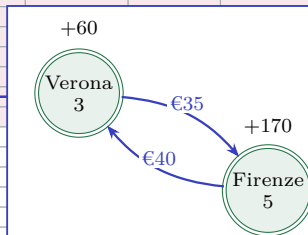
Verona
3

€35

€40

+170

Firenze
5

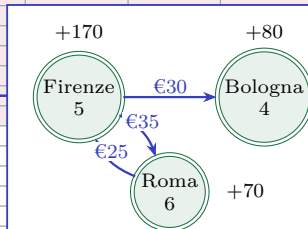


Enter both directions between Verona and Firenze: their costs differ.

Add the remaining Firenze and Roma routes

fx												
	A	B	C	D	E	F	G	H	I	J	K	L
1	BMC: car distribution											
2												
3	Ship		From		To	Cost (€)				Node	Net flow	Required
4	0	1	Genova	2	Milano	30						
5	0	1	Genova	4	Bologna	40						
6	0	2	Milano	3	Verona	50						
7	0	3	Verona	5	Firenze	35						
8	0	5	Firenze	3	Verona	40						
9	0	5	Firenze	4	Bologna	30						
10	0	5	Firenze	6	Roma	35						
11	0	6	Roma	5	Firenze	25						
12	0											
13	0											
14	0											
15												
16						Total cost						

```
graph LR; F5((Firenze 5)) -- "€30" --> B4((Bologna 4)); F5 -- "€35" --> R6((Roma 6)); R6 -- "€25" --> F5; S170[+170] --- F5; S80[+80] --- B4; S70[+70] --- R6; D170[-170] --- F5; D80[-80] --- B4; D70[-70] --- R6
```

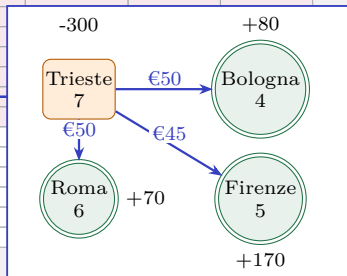


Add one row for each remaining Firenze and Roma route.

Complete the routes from Trieste

fx												
	A	B	C	D	E	F	G	H	I	J	K	L
1	BMC: car distribution											
2												
3	Ship		From			To	Cost (€)			Node	Net flow	Required
4	0	1	Genova	2		Milano	30					
5	0	1	Genova	4		Bologna	40					
6	0	2	Milano	3		Verona	50					
7	0	3	Verona	5		Firenze	35					
8	0	5	Firenze	3		Verona	40					
9	0	5	Firenze	4		Bologna	30					
10	0	5	Firenze	6		Roma	35					
11	0	6	Roma	5		Firenze	25					
12	0	7	Trieste	4		Bologna	50					
13	0	7	Trieste	5		Firenze	45					
14	0	7	Trieste	6		Roma	50					
15												
16						Total cost						

```
graph LR; T[Trieste 7] -- "€50" --> B((Bologna 4)); T -- "€50" --> R((Roma 6)); T -- "€45" --> F((Firenze 5)); B --- B80[+80]; R --- R70[+70]; F --- F70[+70]; T --- T300[-300];
```



Complete rows 12–14: Trieste ships to Bologna, Firenze and Roma.

Calculate the total shipment cost

G16 <i>fx</i> =SUMPRODUCT(B4:B14,G4:G14)												
	A	B	C	D	E	F	G	H	I	J	K	L
1	BMC: car distribution											
2												
3	Ship		From		To	Cost (€)				Node	Net flow	Required
4	0	1	Genova	2	Milano	30						
5	0	1	Genova	4	Bologna	40						
6	0	2	Milano	3	Verona	50						
7	0	3	Verona	5	Firenze	35						
8	0	5	Firenze	3	Verona	40						
9	0	5	Firenze	4	Bologna	30						
10	0	5	Firenze	6	Roma	35						
11	0	6	Roma	5	Firenze	25						
12	0	7	Trieste	4	Bologna	50						
13	0	7	Trieste	5	Firenze	45						
14	0	7	Trieste	6	Roma	50						
15												
16					Total cost	0						

In G16, enter =SUMPRODUCT(B4:B14,G4:G14)

Each quantity is multiplied by its cost per car; SUMPRODUCT adds the eleven route costs.

Enter the city requirements beside the routes

fx												
	A	B	C	D	E	F	G	H	I	J	K	L
1	BMC: car distribution											
2												
3	Ship		From		To	Cost (€)				Node	Net flow	Required
4	0	1	Genova	2	Milano	30		1	Genova			-200
5	0	1	Genova	4	Bologna	40		2	Milano			100
6	0	2	Milano	3	Verona	50		3	Verona			60
7	0	3	Verona	5	Firenze	35		4	Bologna			80
8	0	5	Firenze	3	Verona	40		5	Firenze			170
9	0	5	Firenze	4	Bologna	30		6	Roma			70
10	0	5	Firenze	6	Roma	35		7	Trieste			-300
11	0	6	Roma	5	Firenze	25						
12	0	7	Trieste	4	Bologna	50						
13	0	7	Trieste	5	Firenze	45						
14	0	7	Trieste	6	Roma	50						
15												
16					Total cost	0						

Enter the seven city IDs, names and signed requirements in **I4:J10** and **L4:L10**.

Keep column K for net flow. Negative requirements are available stock; positive requirements are distributor orders.

Calculate a balance for every city

fx												
	A	B	C	D	E	F	G	H	I	J	K	L
1	BMC: car distribution											
2												
3	Ship		From		To		Cost (€)		Node		Net flow	Required
4	0	1	Genova	2	Milano	30	1	Genova				-200
5	0	1	Genova	4	Bologna	40	2	Milano				100
6	0	2	Milano	3	Verona	50	3	Verona				60
7	0	3	Verona	5	Firenze	35	4	Bologna				80
8	0	5	Firenze	3	Verona	40	5	Firenze				170
9	0	5	Firenze	4	Bologna	30	6	Roma				70
10	0	5	Firenze	6	Roma	35	7	Trieste				-300
11	0	6	Roma	5	Firenze	25						
12	0	7	Trieste	4	Bologna	50						
13	0	7	Trieste	5	Firenze	45						
14	0	7	Trieste	6	Roma	50						
15												
16					Total cost	0						

For each city, enter **incoming cars – outgoing cars** in K.

Cars received and then shipped onward do not count as cars kept by that distributor.

Choose the correct balance constraint

Net flow = incoming – outgoing.

b is the node's requirement: **negative for supply, positive for demand.**

Total supply and demand	Balance constraint
Total supply > total demand	Net flow $\geq b$
Total supply = total demand	Net flow = b
Total supply < total demand	Net flow $\leq b$

Supply: 200 cars

$$b = -200$$

Demand: 100 cars

$$b = +100$$

Equal totals: use all stock and fill every order exactly.

Supply exceeds demand: the BMC case

Net flow = incoming – outgoing.

b is the node's requirement: **negative** for supply, **positive** for demand.

Total supply and demand	Balance constraint
Total supply > total demand	Net flow $\geq b$
Total supply = total demand	Net flow = b
Total supply < total demand	Net flow $\leq b$



BMC: 500 cars available, 480 requested.

Meet every order; unused cars stay at the ports.

Port: net flow ≥ -200 means it can send at most 200 cars.

Customer: net flow ≥ 100 means it keeps at least 100 cars.

Supply is smaller than demand

Net flow = incoming – outgoing.

b is the node's requirement: **negative** for supply, **positive** for demand.

Total supply and demand	Balance constraint
Total supply > total demand	Net flow $\geq b$
Total supply = total demand	Net flow = b
Total supply < total demand	Net flow $\leq b$



There are not enough cars to fill every order.

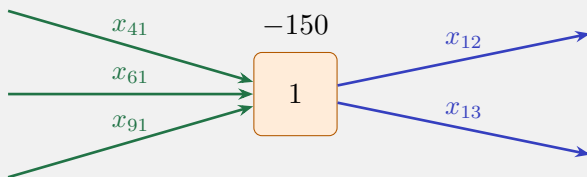
Deliver all available cars at minimum cost.

Ports: net flow = –stock. **Customers:** $0 \leq \text{net flow} \leq \text{order}$.

Transit nodes: net flow = 0. Incoming equals outgoing.

Read the balance of a supply node

x_{ij} is the number of cars shipped from node i to node j .



$$x_{41} + x_{61} + x_{91} - (x_{12} + x_{13}) \geq -150$$

This separate example has 150 cars initially at node 1. Outgoing cars may exceed incoming cars by at most 150.

Calculate net flow in K4

K4	fx													
	A	B	C	D	E	F	G	H	I	J	K	L		
1	BMC: car distribution													
2														
3	Ship		From		To	Cost (€)		Node		Net flow	Required			
4	0	1	Genova	2	Milano	30		1 Genova			-200			
5	0	1	Genova	4	Bologna	40		2 Milano			100			
6	0	2	Milano	3	Verona	50		3 Verona			60			
7	0	3	Verona	5	Firenze	35		4 Bologna			80			
8	0	5	Firenze	3	Verona	40		5 Firenze			170			
9	0	5	Firenze	4	Bologna	30		6 Roma			70			
10	0	5	Firenze	6	Roma	35		7 Trieste			-300			
11	0	6	Roma	5	Firenze	25								
12	0	7	Trieste	4	Bologna	50								
13	0	7	Trieste	5	Firenze	45								
14	0	7	Trieste	6	Roma	50								
15														
16					Total cost	0								

In K4, enter:

```
=SUMIF($E$4:$E$14,I4,$B$4:$B$14)
-SUMIF($C$4:$C$14,I4,$B$4:$B$14)
```

The first SUMIF adds shipments arriving at node I4; the second subtracts shipments leaving it.

Match the city ID to the destination column

K4	fx											
	A	B	C	D	E	F	G	H	I	J	K	L
1	BMC: car distribution											
2												
3	Ship		From		To		Cost (€)	Node		Net flow	Required	
4	0	1	Genova	2	Milano	30	1	Genova				-200
5	0	1	Genova	4	Bologna	40	2	Milano				100
6	0	2	Milano	3	Verona	50	3	Verona				60
7	0	3	Verona	5	Firenze	35	4	Bologna				80
8	0	5	Firenze	3	Verona	40	5	Firenze				170
9	0	5	Firenze	4	Bologna	30	6	Roma				70
10	0	5	Firenze	6	Roma	35	7	Trieste				-300
11	0	6	Roma	5	Firenze	25						
12	0	7	Trieste	4	Bologna	50						
13	0	7	Trieste	5	Firenze	45						
14	0	7	Trieste	6	Roma	50						
15												
16					Total cost	0						

Find I4 in the destination range E4:E14.

Find I4 in the destination range E4:E14.

`SUMIF($E$4:$E$14,I4,$B$4:$B$14)`

The dollar signs keep the route ranges fixed when the formula is copied.

Add the matching shipment quantities

K4	fx											
	A	B	C	D	E	F	G	H	I	J	K	L
1	BMC: car distribution											
2												
3	Ship		From		To	Cost (€)				Node	Net flow	Required
4	0	1	Genova	2	Milano	30		1	Genova			-200
5	0	1	Genova	4	Bologna	40		2	Milano			100
6	0	2	Milano	3	Verona	50		3	Verona			60
7	0	3	Verona	5	Firenze	35		4	Bologna			80
8	0	5	Firenze	3	Verona	40		5	Firenze			170
9	0	5	Firenze	4	Bologna	30		6	Roma			70
10	0	5	Firenze	6	Roma	35		7	Trieste			-300
11	0	6	Roma	5	Firenze	25						
12	0	7	Trieste	4	Bologna	50						
13	0	7	Trieste	5	Firenze	45						
14	0	7	Trieste	6	Roma	50						
15												
16					Total cost	0						

For each destination matching I4, add the shipment in the same row of B4:B14.

For each destination matching **I4**, add the shipment in the same row of **B4:B14**.

SUMIF(\$E\$4:\$E\$14,I4,\$B\$4:\$B\$14)

This adds the incoming cars. Genova has no incoming routes, so its incoming total is 0.

Subtract the outgoing cars

K4	fx											
	A	B	C	D	E	F	G	H	I	J	K	L
1	BMC: car distribution											
2												
3	Ship		From		To	Cost (€)				Node	Net flow	Required
4	0	1	Genova	2	Milano	30		1	Genova			-200
5	0	1	Genova	4	Bologna	40		2	Milano			100
6	0	2	Milano	3	Verona	50		3	Verona			60
7	0	3	Verona	5	Firenze	35		4	Bologna			80
8	0	5	Firenze	3	Verona	40		5	Firenze			170
9	0	5	Firenze	4	Bologna	30		6	Roma			70
10	0	5	Firenze	6	Roma	35		7	Trieste			-300
11	0	6	Roma	5	Firenze	25						
12	0	7	Trieste	4	Bologna	50						
13	0	7	Trieste	5	Firenze	45						
14	0	7	Trieste	6	Roma	50						
15												
16					Total cost	0						

Match I4 to the origins in C4:C14. Add the shipment on those rows.

Match **I4** to the origins in **C4:C14**. Add the shipments on those rows.

SUMIF(\$C\$4:\$C\$14,I4,\$B\$4:\$B\$14)

This adds the outgoing cars. Subtract it from the incoming total to obtain net flow.

Copy the balance formula to every city

K4	fx Net flow = incoming - outgoing											
	A	B	C	D	E	F	G	H	I	J	K	L
1	BMC: car distribution											
2												
3	Ship		From		To		Cost (€)		Node		Net flow	Required
4	0	1	Genova	2	Milano	30	1	Genova		0	-200	
5	0	1	Genova	4	Bologna	40	2	Milano		0	100	
6	0	2	Milano	3	Verona	50	3	Verona		0	60	
7	0	3	Verona	5	Firenze	35	4	Bologna		0	80	
8	0	5	Firenze	3	Verona	40	5	Firenze		0	170	
9	0	5	Firenze	4	Bologna	30	6	Roma		0	70	
10	0	5	Firenze	6	Roma	35	7	Trieste		0	-300	
11	0	6	Roma	5	Firenze	25						
12	0	7	Trieste	4	Bologna	50						
13	0	7	Trieste	5	Firenze	45						
14	0	7	Trieste	6	Roma	50						
15												
16				Total cost		0						

Fill K4 down to K10. I4 becomes I5, I6, ...; the three dollar-marked ranges stay fixed.

All balances are initially 0 because all shipment quantities are 0. Solver will update them through column B.

Minimize the worksheet total cost

	A	B	C	D	E	F	G	H
1	BMC: car distribution							
2								
3	Ship		From		To	Cost (€)		
4	0	1	Genova	2	Milano	30		
5	0	1	Genova	4	Bologna	40		
6	0	2	Milano	3	Verona	50		
7	0	3	Verona	5	Firenze	35		
8	0	5	Firenze	3	Verona	40		
9	0	5	Firenze	4	Bologna	30		
10	0	5	Firenze	6	Roma	35		
11	0	6	Roma	5	Firenze	25		
12	0	7	Trieste	4	Bologna	50		
13	0	7	Trieste	5	Firenze	45		
14	0	7	Trieste	6	Roma	50		
15								
16					Total cost	0		

Solver Parameters
Set Objective: **\$G\$16**
To: ☐ Max ☒ Min
By Changing Variable Cells:
\$B\$4:\$B\$14
Subject to the Constraints:
\$B\$4:\$B\$14 = integer
\$K\$4:\$K\$10 >= \$L\$4:\$L\$10
☒ Make variables non-negative
Method: Simplex LP Options **Solve**

Data → Solver: set objective **G16** and select **Min**.

Let Solver choose whole shipment quantities

	A	B	C	D	E	F	G	H
1	BMC: car distribution							
2								
3	Ship		From		To	Cost (€)		
4	0	1	Genova	2	Milano	30		
5	0	1	Genova	4	Bologna	40		
6	0	2	Milano	3	Verona	50		
7	0	3	Verona	5	Firenze	35		
8	0	5	Firenze	3	Verona	40		
9	0	5	Firenze	4	Bologna	30		
10	0	5	Firenze	6	Roma	35		
11	0	6	Roma	5	Firenze	25		
12	0	7	Trieste	4	Bologna	50		
13	0	7	Trieste	5	Firenze	45		
14	0	7	Trieste	6	Roma	50		
15								
16	Total cost						0	

Solver Parameters

Set Objective:

To: ☐ Max ☒ Min

By Changing Variable Cells:

Subject to the Constraints:

☒ Make variables non-negative

Method:

Changing cells: **B4:B14**. Add the constraint **B4:B14 = integer**.

Apply the city balances in Solver

Solver Parameters

Set Objective:

To: ☐ Max ☒ Min

By Changing Variable Cells:

Subject to the Constraints:

☒ Make variables non-negative

Method:

	H	I	J	K	L
1					
2					
3					
4					
5					
6					
7					
8					
9					
10					
11					
12					
13					
14					
15					
16					

	Node	Net flow	Required
1	Genova	0	-200
2	Milano	0	100
3	Verona	0	60
4	Bologna	0	80
5	Firenze	0	170
6	Roma	0	70
7	Trieste	0	-300

Total cost 0

Add $K4:K10 \geq L4:L10$ to meet all signed city requirements.

Request the best integer shipment plan

fx									
	A	B	C	D	E	F	G	H	
1	BMC: car distribution								
2									
3	Ship		From		To	Cost (€)			
4	0	1	Genova	2	Milano	30			
5	0	1	Genova	4	Bologna	40			
6	0	2	Milano	3	Verona	50			
7	0	3	Verona	5	Firenze	35			
8	0	5	Firenze	3	Verona	40			
9	0	5	Firenze	4	Bologna	30			
10	0	5	Firenze	6	Roma	35			
11	0	6	Roma	5	Firenze	25			
12	0	7	Trieste	4	Bologna	50			
13	0	7	Trieste	5	Firenze	45			
14	0	7	Trieste	6	Roma	50			
15									
16					Total cost	0			

Solver Options
Integer Optimality (%):
Set 0.
Click **OK** to return to Solver.

Use **Integer Optimality = 0%** to request the best integer solution.

Use the Options button in Solver to set Integer Optimality.

Solve the linear shipment model

fx

	A	B	C	D	E	F	G	H
1	BMC: car distribution							
2								
3	Ship		From		To	Cost (€)		
4	0	1	Genova	2	Milano	30		
5	0	1	Genova	4	Bologna	40		
6	0	2	Milano	3	Verona	50		
7	0	3	Verona	5	Firenze	35		
8	0	5	Firenze	3	Verona	40		
9	0	5	Firenze	4	Bologna	30		
10	0	5	Firenze	6	Roma	35		
11	0	6	Roma	5	Firenze	25		
12	0	7	Trieste	4	Bologna	50		
13	0	7	Trieste	5	Firenze	45		
14	0	7	Trieste	6	Roma	50		
15								
16					Total cost	0		

Solver Parameters

Set Objective:

To: ☐ Max ☒ Min

By Changing Variable Cells:

Subject to the Constraints:

☒ Make variables non-negative

Method:

Select **non-negative variables** and **Simplex LP**, then **Solve**.

After solving, choose **Keep Solver Solution** and click **OK**. The model is linear in the shipment quantities.

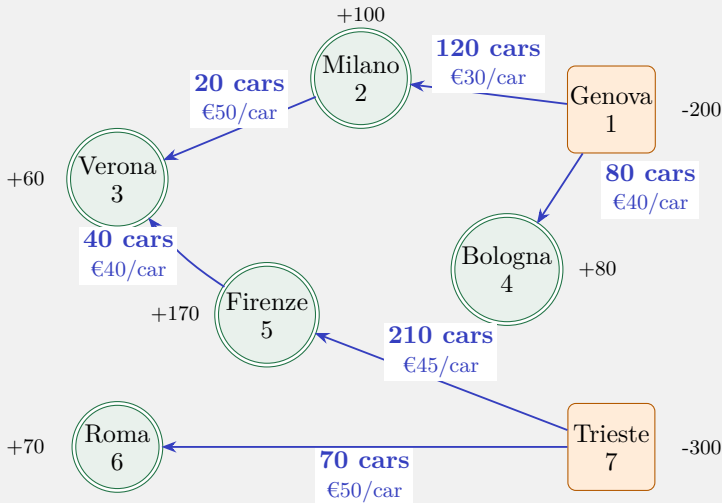
Read the optimal plan in the complete worksheet

fx												
	A	B	C	D	E	F	G	H	I	J	K	L
1	BMC: car distribution											
2												
3	Ship		From		To		Cost (€)		Node		Net flow	Required
4	120	1	Genova	2	Milano	30		1	Genova	-200	-200	
5	80	1	Genova	4	Bologna	40		2	Milano	100	100	
6	20	2	Milano	3	Verona	50		3	Verona	60	60	
7	0	3	Verona	5	Firenze	35		4	Bologna	80	80	
8	40	5	Firenze	3	Verona	40		5	Firenze	170	170	
9	0	5	Firenze	4	Bologna	30		6	Roma	70	70	
10	0	5	Firenze	6	Roma	35		7	Trieste	-280	-300	
11	0	6	Roma	5	Firenze	25						
12	0	7	Trieste	4	Bologna	50						
13	210	7	Trieste	5	Firenze	45						
14	70	7	Trieste	6	Roma	50						
15												
16				Total cost			22.350					

Minimum total cost: €22,350. Trieste ships 280 of its 300 cars; 20 remain.

The five distributor balances exactly match their orders: 100, 60, 80, 170 and 70 cars.

Follow the optimal car shipments



Minimum cost: €22,350. Twenty cars stay at Trieste.

Arrows show the positive shipments and their unit costs. Routes assigned zero are omitted.

The satellite navigation problem

Compare the fastest route with the most scenic route.



Open **The satellite navigation problem** on Moodle.

The navigation task

The navigator assigns each road a **travel time** and a **scenic score**. A higher score means better views.

Fastest route

Minimize the total hours.

Most scenic route

Maximize the total scenic points.

Follow a complete route along the arrows.

Torino

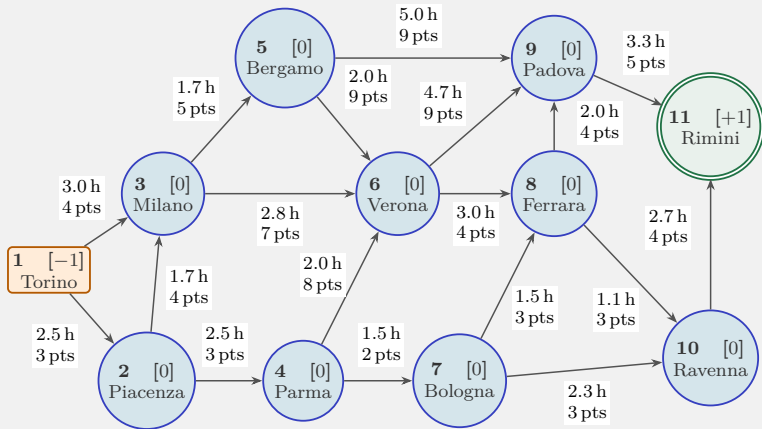


Rimini

Find both routes. Add the time and scenic points of every road selected along each route.

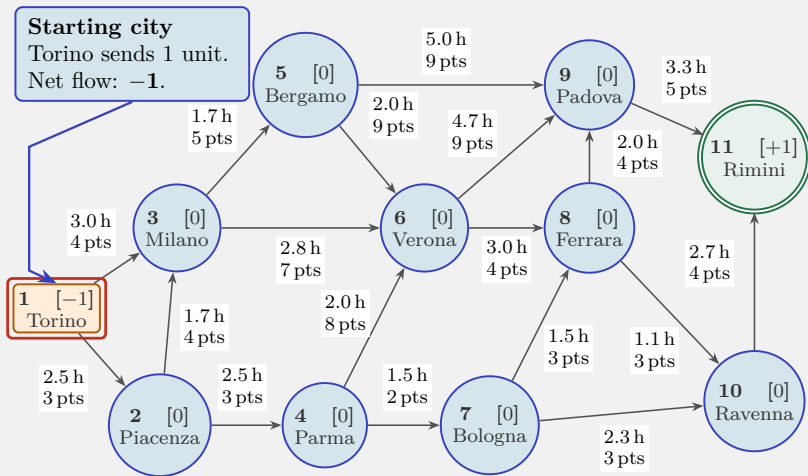
Choose a route from Torino to Rimini

Each road: travel time (h) and scenic score (pts)



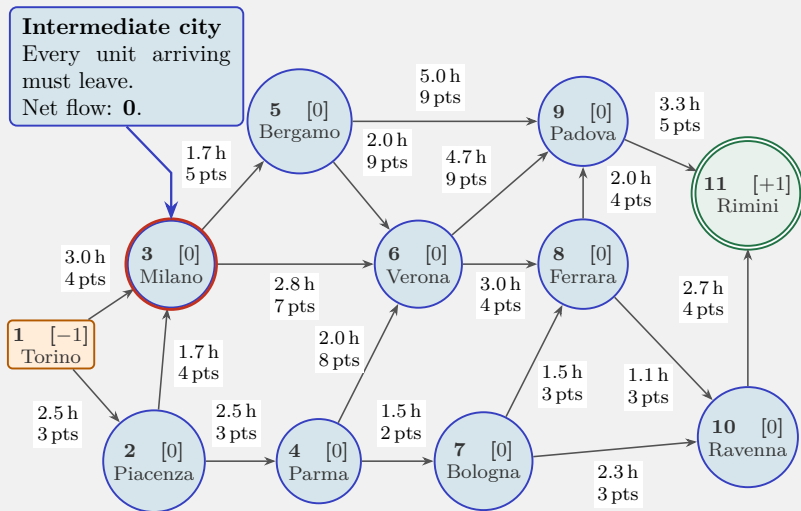
Orange: start. **Blue:** transit. **Green double border:** destination. Brackets show required inflow minus outflow. One unit travels from Torino to Rimini.

The starting city sends one unit



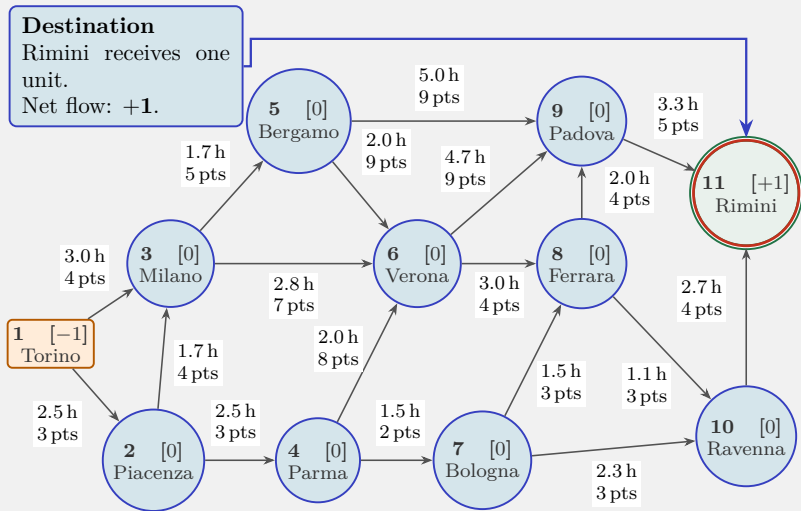
One unit represents the driver's journey. It must follow a continuous route.
Net flow means inflow minus outflow.

Intermediate cities pass the unit on



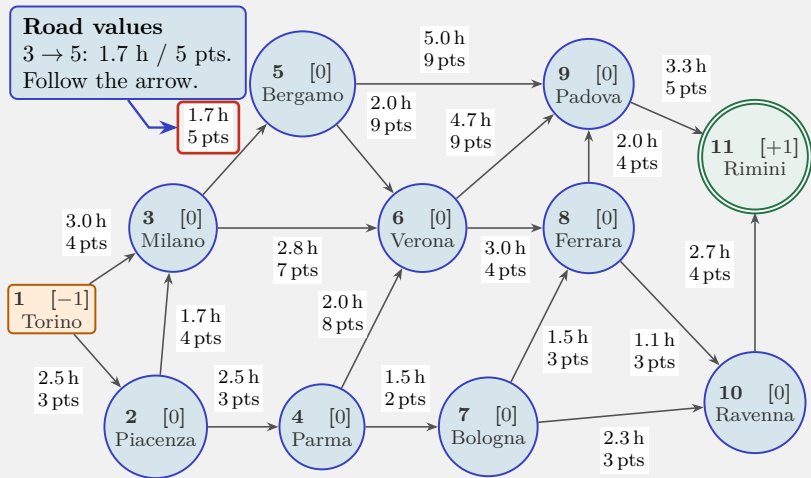
One unit represents the driver's journey. It must follow a continuous route.
Net flow means inflow minus outflow.

The destination receives one unit



One unit represents the driver's journey. It must follow a continuous route.
Net flow means inflow minus outflow.

Read the travel time and scenic score



Find the **fastest route** and the **most scenic route**. Add the values of every road selected.

Enter the complete road and city tables

	A	B	C	D	E
1	Use	From	To	Time (h)	Scenic
2	0	1	2	2.5	3
3	0	1	3	3.0	4
4	0	2	3	1.7	4
5	0	2	4	2.5	3
6	0	3	5	1.7	5
7	0	3	6	2.8	7
8	0	4	6	2.0	8
9	0	4	7	1.5	2
10	0	5	6	2.0	9
11	0	5	9	5.0	9
12	0	6	8	3.0	4
13	0	6	9	4.7	9
14	0	7	8	1.5	3
15	0	7	10	2.3	3
16	0	8	9	2.0	4
17	0	8	10	1.1	3
18	0	9	11	3.3	5
19	0	10	11	2.7	4
20	Total time				
21	Scenic points				

	G	H	I	J
1	Node	City	Net flow	Required
2	1	Torino		-1
3	2	Piacenza		0
4	3	Milano		0
5	4	Parma		0
6	5	Bergamo		0
7	6	Verona		0
8	7	Bologna		0
9	8	Ferrara		0
10	9	Padova		0
11	10	Ravenna		0
12	11	Rimini		1

New sheet: enter both tables.

A2:A19 = 0 initially.

Use = 1 selects a road.

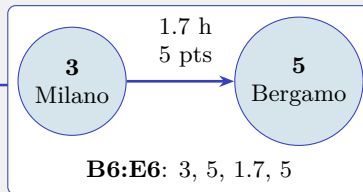
Leave flows and totals blank.

Keep the road table in A:E and the city table in G:J. Use = 1 selects a road;
Use = 0 excludes it.

Translate each road into a worksheet row

	A	B	C	D	E
1	Use	From	To	Time (h)	Scenic
2	0	1	2	2.5	3
3	0	1	3	3.0	4
4	0	2	3	1.7	4
5	0	2	4	2.5	3
6	0	3	5	1.7	5
7	0	3	6	2.8	7
8	0	4	6	2.0	8
9	0	4	7	1.5	2
10	0	5	6	2.0	9
11	0	5	9	5.0	9
12	0	6	8	3.0	4
13	0	6	9	4.7	9
14	0	7	8	1.5	3
15	0	7	10	2.3	3
16	0	8	9	2.0	4
17	0	8	10	1.1	3
18	0	9	11	3.3	5
19	0	10	11	2.7	4
20	Total time				
21	Scenic points				

	G	H	I	J
1	Node	City	Net flow	Required
2	1	Torino		-1
3	2	Piacenza		0
4	3	Milano		0
5	4	Parma		0
6	5	Bergamo		0
7	6	Verona		0
8	7	Bologna		0
9	8	Ferrara		0
10	9	Padova		0
11	10	Ravenna		0
12	11	Rimini		1



From and To identify the arrow; Time and Scenic copy its two labels. The other rows follow the same rule.

Calculate all city balances

	A	B	C	D	E
1	Use	From	To	Time (h)	Scenic
2	0	1	2	2.5	3
3	0	1	3	3.0	4
4	0	2	3	1.7	4
5	0	2	4	2.5	3
6	0	3	5	1.7	5
7	0	3	6	2.8	7
8	0	4	6	2.0	8
9	0	4	7	1.5	2
10	0	5	6	2.0	9
11	0	5	9	5.0	9
12	0	6	8	3.0	4
13	0	6	9	4.7	9
14	0	7	8	1.5	3
15	0	7	10	2.3	
16	0	8	9	2.0	
17	0	8	10	1.1	
18	0	9	11	3.3	
19	0	10	11	2.7	
20	Total time				
21	Scenic points				

	G	H	I	J
1	Node	City	Net flow	Required
2	1	Torino	0	-1
3	2	Piacenza	0	0
4	3	Milano	0	0
5	4	Parma	0	0
6	5	Bergamo	0	0
7	6	Verona	0	0
8	7	Bologna	0	0
9	8	Ferrara	0	0
10	9	Padova	0	0
11	10	Ravenna	0	0
12	11	Rimini	0	1

In **I2**, enter this formula; fill down to **I12**.
`=SUMIF($C$2:$C$19,G2,$A$2:$A$19)`
`-SUMIF($B$2:$B$19,G2,$A$2:$A$19)`

The first SUMIF adds selected roads arriving at a city; the second adds selected roads leaving it.

Calculate travel time and scenic points

	A	B	C	D	E
1	Use	From	To	Time (h)	Scenic
2	0	1	2	2.5	3
3	0	1	3	3.0	4
4	0	2	3	1.7	4
5	0	2	4	2.5	3
6	0	3	5	1.7	5
7	0	3	6	2.8	7
8	0	4	6	2.0	8
9	0	4	7	1.5	2
10	0	5	6	2.0	9
11	0	5	9	5.0	9
12	0	6	8	3.0	4
13	0	6	9	4.7	9
14	0	7	8	1.5	3
15	0	7	10	2.3	3
16	0	8	9	2.0	4
17	0	8	10	1.1	3
18	0	9	11	3.3	5
19	0	10	11	2.7	4
20	Total time			0.0	
21	Scenic points			0	

	G	H	I	J
1	Node	City	Net flow	Required
2	1	Torino	0	-1
3	2	Piacenza	0	0
4	3	Milano	0	0
5	4	Parma	0	0
6	5	Bergamo	0	0
7	6	Verona	0	0
8	7	Bologna	0	0
9	8	Ferrara	0	0
10	9	Padova	0	0
11	10	Ravenna	0	0
12	11	Rimini	0	1

D20: total hours

=SUMPRODUCT(A2:A19,D2:D19)

D21: scenic points

=SUMPRODUCT(A2:A19,E2:E19)

SUMPRODUCT adds each road value multiplied by Use. Roads with Use = 0 contribute nothing.

Find the fastest route

Solver Parameters

Set Objective:

To: ☐ Max ☒ Min

By Changing Variable Cells:

Subject to the Constraints:

\$A\$2:\$A\$19 = binary

\$I\$2:\$I\$12 = \$J\$2:\$J\$12

Add

Change

Delete

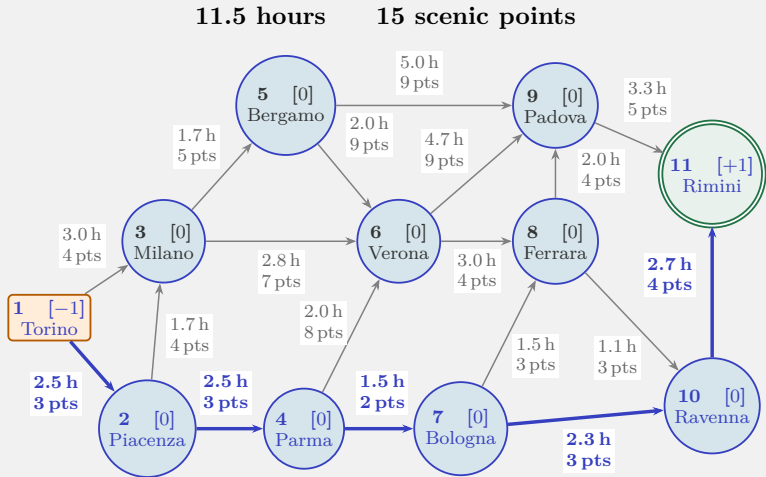
☒ Make Unconstrained Variables Non-Negative

Solving Method:

Use: binary.
Net flow = Required.
Integer Optimality
(%): 0.
Click **Solve**.

Use **Simplex LP**: the objective and constraints are linear. Keep the final solution.

The fastest route



Torino → Piacenza → Parma → Bologna → Ravenna → Rimini.
Copy **D20** to **H15** using **Paste Special** → **Values**: save 11.5.

Find the most scenic route

Solver Parameters

Set Objective:

To: ☒ Max ☐ Min

By Changing Variable Cells:

Subject to the Constraints:

\$A\$2:\$A\$19 = binary	Add Change Delete
\$I\$2:\$I\$12 = \$J\$2:\$J\$12	

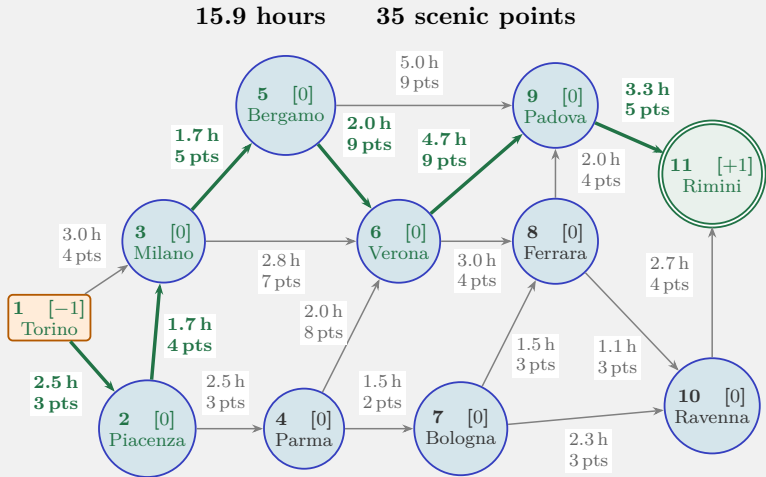
☒ Make Unconstrained Variables Non-Negative

Solving Method:

Objective
D21, Max.
Keep all constraints.
Click **Solve**.

Keep **Simplex LP**, integer constraints and **Integer Optimality (%) = 0**.

The most scenic route



Torino → Piacenza → Milano → Bergamo → Verona → Padova → Rimini.
Copy **D21** to **H16** using **Paste Special** → **Values**: save 35.

Choose a compromise route

Which route minimizes the **larger relative shortfall**, giving equal importance to travel time and scenery?

Travel time

Extra time compared with the fastest route:

$$a = (T - 11.5)/11.5.$$

Scenery

Points lost compared with the most scenic route:

$$b = (35 - S)/35.$$

$$\min_{x,q} q \quad \text{subject to } a \leq q, \ b \leq q, \ q \geq 0.$$

Keep the binary road decisions and all city balances.
Both weights are fixed at 1.

T and S depend on the selected roads. Solver changes those roads and q to reduce the worse percentage.

Prepare the complete objective comparison table

	G	H	I
14	Measure	Best alone	Shortfall
15	Travel time	11.5	38.26%
16	Scenic points	35	0.00%
17	Shortfall limit q	100.00%	
18	Objective q	100.00%	

H17: enter 1.

H18: enter =H17.

Format as percentages.

I15: =(D20-H15)/H15

I16: =(H16-D21)/H16

Format as percentages.

Enter the labels in G14:I18. Keep H15 = 11.5 and H16 = 35 as fixed values; they must not change with the route.

Minimize the worst shortfall

Solver Parameters

Set Objective:

To: ☐ Max ☒ Min

By Changing Variable Cells:

Subject to the Constraints:

\$A\$2:\$A\$19 = binary

\$I\$2:\$I\$12 = \$J\$2:\$J\$12

\$I\$15:\$I\$16 ≤ \$H\$17

Add

Change

Delete

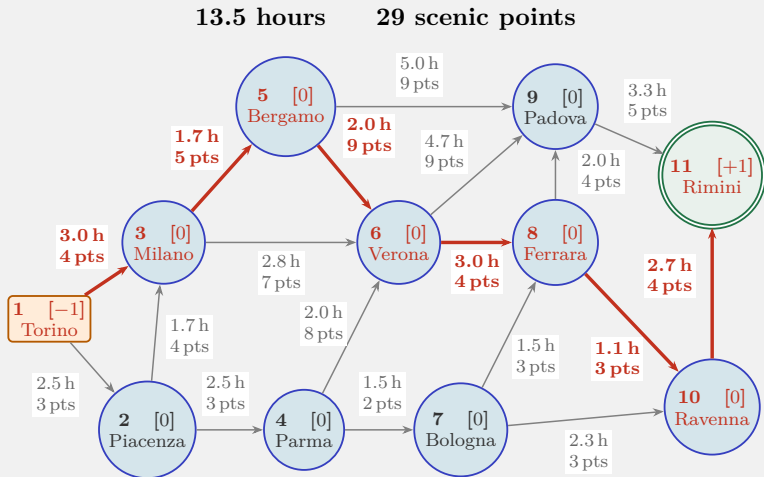
☒ Make Unconstrained Variables Non-Negative

Solving Method:

Add **I15:I16 ≤ H17**.
Minimize **H18**.
Binary: A2:A19 only.
Click **Solve**.

Keep **Simplex LP** and **Integer Optimality (%) = 0**. The limit q may contain decimals.

The compromise route



Torino → Milano → Bergamo → Verona → Ferrara → Ravenna → Rimini.

Check the compromise result

Route	Hours	Points	Time loss	Scenic loss	Worst loss
Fastest	11.5	15	0.00%	57.14%	57.14%
Most scenic	15.9	35	38.26%	0.00%	38.26%
Compromise	13.5	29	17.39%	17.14%	17.39%

$$(13.5 - 11.5)/11.5 = \mathbf{17.3913\%}$$

$$(35 - 29)/35 = \mathbf{17.1429\%}$$

The compromise minimizes the larger percentage. It does not minimize their sum.

Why the compromise is optimal

A better route would need $q < 4/23 \approx \mathbf{17.3913\%}$.

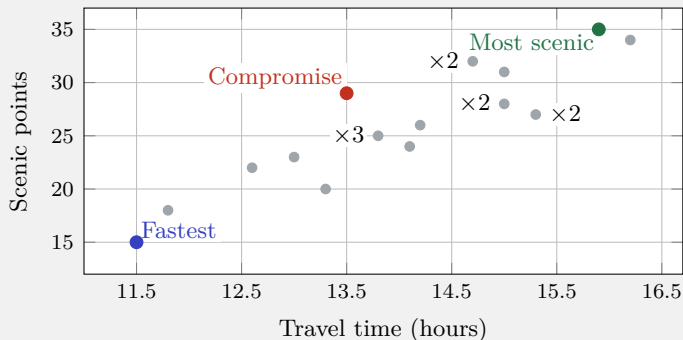
Travel time
 $T < 13.5$ hours

Scenic score
 $S > 28.913$ points

Every route below **13.5 hours** has at most **23 scenic points**. No route meets both requirements.

All 20 complete routes were checked. The selected route achieves the smallest possible worst shortfall.

Travel time and scenery across all routes



The **20 complete routes** give **15 distinct time–score pairs**. Labels $\times 2$ and $\times 3$ show how many routes share a point. The fastest, most scenic and compromise routes are unique.