

## Example 4.1

### Static Workforce Scheduling Models

## Background Information

- A post office requires different numbers of full-time employees on different days of the week.
- The number of full-time employees required each day is given in this table.

#### Employee Requirements for Post Office

Monday	17
Tuesday	13
Wednesday	15
Thursday	19
Friday	14
Saturday	16
Sunday	11

[4.2](#) | [4.3](#) | [4.4](#) | [4.5](#) | [4.6](#) | [4.7](#)



## Background Information -- continued

- Union rules state that each full-time employee must work 5 consecutive days and then receive 2 days off.
- For example, an employee who works Monday to Friday must be off Saturday and Sunday.
- The post office wants to meet its daily requirements using only full-time employees.
- Its objective is to minimize the number of full-time employees that must be hired.

[4.2](#) | [4.3](#) | [4.4](#) | [4.5](#) | [4.6](#) | [4.7](#)



## Solution

- To model the Post Office problem with a spreadsheet, we must keep track of the following:
  - Number of employees starting work on each day of the week
  - Number of employees working each day
  - Total number of employees
- It is important to keep track of the number of employees starting work each day, because this is the only way to incorporate the fact that workers work 5 consecutive days.

[4.2](#) | [4.3](#) | [4.4](#) | [4.5](#) | [4.6](#) | [4.7](#)



## POSTAL.XLS

- This file shows the spreadsheet model for this problem.
- The spreadsheet figure on the next slide shows the model.

[4.2](#) | [4.3](#) | [4.4](#) | [4.5](#) | [4.6](#) | [4.7](#)

	A	B	C	D	E	F	G	H	
1	<b>Post Office scheduling model - nonoptimal solution shown</b>								
2									
3	Number starting their five-day shift on various days							<b>Range names used:</b>	
4	Mon	5						Starting - B4:B10	
5	Tue	5						Available - B21:H21	
6	Wed	5						MinReqd - B23:H23	
7	Thu	5						TotEmployees - B25	
8	Fri	5							
9	Sat	5							
10	Sun	5							
11									
12	Number working on various days (along top) who started their shift on various days (along side)								
13		Mon	Tue	Wed	Thu	Fri	Sat	Sun	
14	Mon	5	5	5	5	5			
15	Tue		5	5	5	5	5		
16	Wed			5	5	5	5	5	
17	Thu	5			5	5	5	5	
18	Fri	5	5			5	5	5	
19	Sat	5	5	5			5	5	
20	Sun	5	5	5	5			5	
21	Available	25	25	25	25	25	25	25	
22		>=	>=	>=	>=	>=	>=	>=	
23	Min required	17	13	15	13	14	16	11	
24									
25	Total employees	35							

[4.2](#) | [4.3](#) | [4.4](#) | [4.5](#) | [4.6](#) | [4.7](#)



## Developing the Model

- To form this spreadsheet, proceed as follows.
  - **Daily requirements.** Enter the number of employees needed on each day of the week in the MinReqd range.
  - **Employees beginning each day.** Enter any trial values for the number of employees beginning work on each day of the week in the Starting range.
  - **Employees on hand each day.** The important key to this solution is to realize that the numbers in the Starting range do not represent the number of workers who will show up each day. As an example, the number who start on Monday work Monday through Friday. Therefore, enter the formula **=B\$4** in cell B14 and copy it across to cell F14. Proceed similarly for rows 15-20, being careful to take “wrap arounds” into account.

[4.2](#) | [4.3](#) | [4.4](#) | [4.5](#) | [4.6](#) | [4.7](#)



## Developing the Model -- continued

- After completing these rows calculate the total number who show up each day by entering the formula **=SUM(B14:B20)** in cell B21 and copying across to cell H21.
- **Total employees.** Calculate the total number of employees in cell I5 with the formula **=SUM(Starting)** in the TotEmployees cell.
- At this point, you might want to try rearranging the numbers in the Starting range to see if you can “guess” an optimal solution. It’s not that easy.

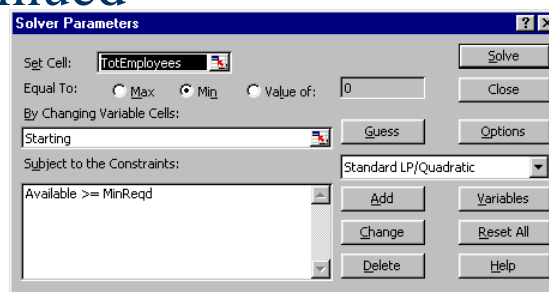
[4.2](#) | [4.3](#) | [4.4](#) | [4.5](#) | [4.6](#) | [4.7](#)

## Developing the Model -- continued

- Using the Solver – Now invoke the Solver and specify the following.
  - **Objective.** Choose the TotEmployees cell as the target cell to minimize.
  - **Changing cells.** Choose the Starting range as the changing cells.
  - **Daily requirement constraint.** Enter the constraint Avail $\geq$ MinReqd. This constraint ensures that enough people are working each day. After completing these steps, the Solver dialog box should appear as shown on the next slide.

[4.2](#) | [4.3](#) | [4.4](#) | [4.5](#) | [4.6](#) | [4.7](#)

## Developing the Model -- continued



- **Specify nonnegativity and optimize.** Under Solver Options, check the nonnegativity box and use the LP algorithm to obtain the optimal solution shown on the following slide.

[4.2](#) | [4.3](#) | [4.4](#) | [4.5](#) | [4.6](#) | [4.7](#)

	A	B	C	D	E	F	G	H
1	<b>Post Office scheduling model - optimal noninteger solution</b>							
2								
3	Number starting their five-day shift on various days							
4	Mon	6.33						
5	Tue	5.00						
6	Wed	0.33						
7	Thu	7.33						
8	Fri	0.00						
9	Sat	3.33						
10	Sun	0.00						
11								
12	Number working on various days (along top) who started their shift on various days (along side)							
13		Mon	Tue	Wed	Thu	Fri	Sat	Sun
14	Mon	6.33	6.33	6.33	6.33	6.33		
15	Tue		5.00	5.00	5.00	5.00	5.00	
16	Wed			0.33	0.33	0.33	0.33	0.33
17	Thu	7.33			7.33	7.33	7.33	7.33
18	Fri	0.00	0.00			0.00	0.00	0.00
19	Sat	3.33	3.33	3.33			3.33	3.33
20	Sun	0.00	0.00	0.00	0.00			0.00
21	Available	17.00	14.67	15.00	19.00	19.00	16.00	11.00
22		>=	>=	>=	>=	>=	>=	>=
23	Min required	17	13	15	19	14	16	11
24								
25	Total employees	22.33						

4.2 | 4.3 | 4.4 | 4.5 | 4.6 | 4.7

## Developing the Model -- continued

- This optimal solution requires the number of employees starting work on some days to be a fraction.
- Because part-time employees are not allowed, this solution is unrealistic.
- We will now show how to solve the post office model when the number of employees beginning work each day must be an integer.

4.2 | 4.3 | 4.4 | 4.5 | 4.6 | 4.7


## Using Solver with Integer Constraints

- In the Solver dialog box, add the constraint Starting=Integer.
- To do this simply select “int” instead of <=,=, or >= in the Add Constraint dialog.
- Now when you solve, you obtain the solution shown on the following slide.
- As we see, the post office needs to hire 23 full-time employees. This solution reveals an aspect of some modeling problems.

[4.2](#) | [4.3](#) | [4.4](#) | [4.5](#) | [4.6](#) | [4.7](#)

	A	B	C	D	E	F	G	H
1	<b>Post Office scheduling model</b>							
2								
3	Number starting their five-day shift on various days							
4	Mon	6						
5	Tue	6						
6	Wed	0						
7	Thu	7						
8	Fri	0						
9	Sat	4						
10	Sun	0						
11								
12	Number working on various days (along top) who started their shift on various days (along side)							
13		Mon	Tue	Wed	Thu	Fri	Sat	Sun
14	Mon	6	6	6	6	6		
15	Tue		6	6	6	6	6	
16	Wed			0	0	0	0	0
17	Thu	7			7	7	7	7
18	Fri	0	0			0	0	0
19	Sat	4	4	4			4	4
20	Sun	0	0	0	0			0
21	Available	17	16	16	19	19	17	11
22		>=	>=	>=	>=	>=	>=	>=
23	Min required	17	13	15	19	14	16	11
24								
25	Total employees	23						

[4.2](#) | [4.3](#) | [4.4](#) | [4.5](#) | [4.6](#) | [4.7](#)



## Using Solver with Integer Constraints -- continued

- Because of irregular daily requirements and the constraint on consecutive days off, no solution can exactly match available workers to daily requirements.
- All solutions have surplus workers some days. Sometimes the optimal solution to a modeling problem is not the “perfect” solution you were hoping for.
- By the way, you may get a different schedule that is still optimal. This is a case of **multiple optimal solutions** and is not at all uncommon in LP problems.


[4.2](#) | [4.3](#) | [4.4](#) | [4.5](#) | [4.6](#) | [4.7](#)



## Using Solver with Integer Constraints -- continued

- As we see in this example, it is easy to add integer constraints to an LP model.
- However, you should be aware that this makes the problem much more difficult to solve mathematically.
- In fact, Solver uses a different algorithm called **branch and bound** for integer models.
- Our advice is to use integer constraints sparingly.

[4.2](#) | [4.3](#) | [4.4](#) | [4.5](#) | [4.6](#) | [4.7](#)



## Using Solver with Integer Constraints -- continued

- One other comment about integer constraints concerns Solver's **Tolerance** setting.
- As Solver searches for the best integer solution, it is often able to find "good" solutions fairly quickly, but it often has to spend a lot of time finding slightly better solutions.
- A nonzero tolerance setting allows it to quit early. The default tolerance setting is 0.05. This means that if Solver finds a feasible solution that is guaranteed to have an objective value no more than 5% from the optimal value, it will quit and report this "good" solution.

[4.2](#) | [4.3](#) | [4.4](#) | [4.5](#) | [4.6](#) | [4.7](#)



## Sensitivity Analysis

- The most obvious type of sensitivity analysis involves examining how the work schedule and the total number of employees change as the number of employees required each day changes.
- Suppose the number of employees needed each day of the week increases by 2, 4, 6. How does this change the total number of employees needed?
- We can answer this by using the SolverTable add-in, but we first have to alter the model slightly as shown on the next slide.

[4.2](#) | [4.3](#) | [4.4](#) | [4.5](#) | [4.6](#) | [4.7](#)

	A	B	C	D	E	F	G	H	I	J	K	L	
1	<b>Post Office scheduling model</b>												
2													
3	Number starting their five-day shift on various days												
4	Mon	6											
5	Tue	6											
6	Wed	0											
7	Thu	7											
8	Fri	0											
9	Sat	4											
10	Sun	0											
11													
12	Min required	17	13	15	19	14	16	11	Extra per day (for sensitivity)				0
13													
14	Number working on various days (along top) who started their shift on various days (along side)												
15		Mon	Tue	Wed	Thu	Fri	Sat	Sun					
16	Mon	6	6	6	6	6							
17	Tue		6	6	6	6	6						
18	Wed			0	0	0	0	0					
19	Thu	7			7	7	7	7					
20	Fri	0	0			0	0	0					
21	Sat	4	4	4			4	4					
22	Sun	0	0	0	0			0					
23	Available	17	16	16	19	19	17	11					
24		>=	>=	>=	>=	>=	>=	>=					
25	Final min required	17	13	15	19	14	16	11					
26													
27	Total employees	23											
28													
29	<b>Sensitivity of total number of employees to extra required each day</b>												
30		Extra	Total										
31			=B\$27										
32		0	23										
33		2	25										
34		4	28										
35		6	31										

4.2 | 4.3 | 4.4 | 4.5 | 4.6 | 4.7

## Sensitivity Analysis -- continued

- The problem is that we want to increase each of the daily minimal required values by the same amount. Therefore, we move the original requirements up to row 12, enter a trial value for the extra number required per day in the Extra cell, and enter the formula **=B12+Extra** in cell B25, which is then copied across.
- Now we can use the SolverTable option, using Extra cell as the single input, letting it vary from 0 to 6 in increments of 2, and specifying the TotEmployees cell as the single output cell.

4.2 | 4.3 | 4.4 | 4.5 | 4.6 | 4.7



## Sensitivity Analysis -- continued

- The results appear in rows 32-35 of the optimal solution.
- When the requirement increases by 2 each day, only 2 extra employees are necessary. However, when the requirement increases by 4 each day, more than 4 extra employees are necessary. The same is true when the requirement increases by 6 each day.

[4.2](#) | [4.3](#) | [4.4](#) | [4.5](#) | [4.6](#) | [4.7](#)



## Creating a “Fair” Schedule for Employees

- The Solver solution to the example requires 6 employees to start work on Monday, 6 on Tuesday, 7 on Thursday, and 4 on Saturday.
- Certainly the 4 employees who begin work on Saturday will be unhappy, because they never get a weekend off!
- The Solver solution can be used to implement a “fairer” schedule that treats all employees in an equal fashion.

[4.2](#) | [4.3](#) | [4.4](#) | [4.5](#) | [4.6](#) | [4.7](#)



## Creating a “Fair” Schedule for Employees -- continued

- We simply rotate the schedules of the employees over a 23 week period.
- To see how this is done, consider the following schedule.
  - Weeks 1-6: Start on Monday
  - Weeks 7-12: Start on Tuesday
  - Weeks 13-19: Start on Thursday
  - Weeks 20-23: Start on Saturday

[4.2](#) | [4.3](#) | [4.4](#) | [4.5](#) | [4.6](#) | [4.7](#)



## Creating a “Fair” Schedule for Employees -- continued

- Employee 1 follows this schedule for a 23 week period. Employee 2 starts with week 2 of this schedule and follows the schedule through week 23.
- For the last week of the 23-week period, employee 2 follows week 1 of the schedule. We continue in this fashion to generate a 23-week schedule for each employee.
- This method of scheduling treats each employee equally.

[4.2](#) | [4.3](#) | [4.4](#) | [4.5](#) | [4.6](#) | [4.7](#)



## Modeling Issues

1. This example is a **static** scheduling model, because we assume that the post office faces the same situation each week. In reality, demands change over time, workers take vacations in the summer, and so on, so the post office does not face the same situation each week. **Dynamic** scheduling models will be discussed later.
2. If you wanted to set up a weekly scheduling model for a supermarket or a restaurant, the number of variables could be very large and the computer might have difficulty finding an exact solution. In this situation *heuristic* methods can be used to find a good solution.

[4.2](#) | [4.3](#) | [4.4](#) | [4.5](#) | [4.6](#) | [4.7](#)



## Modeling Issues -- continued

3. Our model can be easily expanded to handle part-time employees, the use of overtime, and alternative objective functions such as maximizing the number of weekend days off received by employees.
4. How did we determine the number of workers needed each day? Perhaps the post office wants to have enough employees to assure that 95% of all letters are sorted within an hour. To determine the number of employees that are needed to provide adequate service the post office would need to use queuing theory.

[4.2](#) | [4.3](#) | [4.4](#) | [4.5](#) | [4.6](#) | [4.7](#)