Case Report: Stocking Sets

Prepared for

Brenda Sims
Kitchen Showroom Floor Saleswoman
Furniture City

Date: November 23, 2016

by Gurpal Bisra, Maja Peralta, and Barend Lotter
Centre for Operations Excellence

1 Executive Summary

Brenda Sims, the saleswoman on the kitchen showroom floor of Furniture City, is concerned her company is losing customer goodwill and business because of stock-outs. Brenda discovered 80% of her orders for kitchen sets couldn't be filled as the local warehouse was overstocked with unpopular items and understocked with popular ones. A new inventory policy is required, with data-driven solutions pitched to management, to prevent customer delays and revenue loss due to wasted warehouse space. Our services were requested to formulate an integer programming model to maximize the total number of kitchen sets Furniture City stocks in the local warehouse and, consequently, their number of customer orders.

This report discusses the binary integer programming models we formulated under various scenarios and our recommendations for Brenda Sims to optimize their warehouse stocking strategy to influence management to accept a new inventory policy. We first considered how to maximize the number of kitchen sets Furniture City can stock in the local warehouse given the current space requirements. All analysis re based on the assumptions that inventory is replenished immediately.

For this scenario, we found that the maximum number of kitchen sets that could be stocked is four. The solution we provided is not unique and various combinations of items can be chosen to obtain a maximum objective value of four. Next, we reformulated our integer programming model, to include additional space that was made available by discontinuing the nursery set range. We found that the additional space enabled us to stock one more kitchen set (five in total). Furthermore, we also evaluated the potential inventory if all the space previously assigned to the nursery sets was made available for extra kitchen sets. We found that this additional space enabled us to stock a maximum of six kitchen sets. Finally, we discuss how the inventory policy will be affected by the kitchen sets which cannot be replenished immediately.

Our analysis was conducted while meeting the following requirements:

1. **Binary Constraint:** The values of x_i and y are binary.
2. **Light Fixtures:** A maximum of two different styles of light fixtures can be in stock.
3. **Cabinets:** A maximum of two different styles of cabinets can be in stock.
4. **Countertops:** A maximum of three different styles of countertops can be in stock.
5. **Sinks:** A maximum of two different styles of sinks can be in stock.
6. **Dishwashers and Ranges:** A combination of four different styles of dishwashers and ranges can be held in stock.
7. **Kitchen Set Tiles:** A maximum of two different styles of tiles can be in stock.
8. **Kitchen Set Wallpaper:** A maximum of two different styles of wallpaper can be in stock.
9. **Kitchen Set:** All items composing a kitchen set must be in stock before selling the kitchen set.

Table of Contents

Table of Figures

Table of Tables

2 Introduction

Brenda Sims, the saleswoman on the kitchen showroom floor of Furniture City, is concerned her company is losing customer goodwill and business because of stock-outs. Brenda discovered that over 80% of her orders for kitchen sets couldn't be filled because the local warehouse was overstocked with unpopular items and understocked with popular ones. A new inventory policy is required, with data-driven solutions pitched to management, to prevent customer delays and revenue loss due to wasted warehouse space. Our services were requested to formulate an integer programming model to maximize the total number of kitchen sets Furniture City stocks in the local warehouse and, consequently, their number of customer orders.

This report discusses the binary integer programming models we formulated under various scenarios and our recommendations for Brenda Sims to optimize their warehouse stocking strategy to influence management to accept a new inventory policy. We first considered how to maximize the number of kitchen sets, and consequently the number of customer orders, that Furniture City stocks in the local warehouse. We then noted how many different kitchen sets were in stock and how many of each feature and style should be stocked to meet customer demands. Next, we reformulated our integer programming model, to factor discontinuing nursery sets, to determine the optimal inventory policy for the kitchen department. As per request of Ms. Sims, we determined how the additional testing space provided to the kitchen department obtained from the nursery department helps Furniture City. Finally, we discuss how the inventory policy will be affected by the kitchen sets which cannot be replenished immediately.

Our analysis was conducted while meeting the following requirements:

1. **Binary Constraint:** All decision variables are binary (1 or 0).
2. **Light Fixtures:** A maximum of two different styles of light fixtures can be in stock.
3. **Cabinets:** A maximum of two different styles of cabinets can be in stock.
4. **Countertops:** A maximum of three different styles of countertops can be in stock.
5. **Sinks:** A maximum of two different styles of sinks can be in stock.
6. **Dishwashers and Ranges:** A combination of four different styles of dishwashers and ranges can be held in stock.
7. **Kitchen Set Tiles:** A maximum of two different styles of tiles can be in stock.
8. **Kitchen Set Wallpaper:** A maximum of two different styles of wallpaper can be in stock.
9. **Kitchen Set:** All items composing a kitchen set must be in stock before selling the kitchen set.

Furthermore, we made the following assumptions to conduct our analysis:

1. when a customer orders a kitchen set, all the items composing that kitchen set are replenished at the local warehouse immediately.

2.1 Information Provided

Rationale

Brenda Simms recommends changing local warehouse stocking policy for the following reasons: (1) Daniel Holbrook, an expeditor at the local warehouse for Furniture City, consistently discovered items which have been unstocked the past year; (2) obsolete items are overstocked such as pea-green wallpaper; (3) revenue loss is occurring due to wasted warehouse space; and (4) Furniture City is losing customer trust.

Current Management Policy

Daniel has confirmed the inventory problem has occurred because management had a policy of stocking every furniture item on the showroom floor in the local warehouse. Management only replenished inventory every three months, and when inventory was replenished, management ordered every item regardless of if it had been sold. Daniel was unsuccessful in relaying this problem to management previously. Hence, Brenda feels responsibility to sell her inventory policy to management.

Brenda's Plan

Using her extensive sales experience, Brenda decided that the most effective sales strategy would be to use her kitchen department as a model for the new inventory policy. She would identify all kitchen sets comprising 85 percent of customers' orders. Given the fixed amount of warehouse space allocated to the kitchen department, she would identify the items Furniture City should stock in order to satisfy the greatest number of customer orders. She would then calculate the revenue from satisfying customer orders under the new inventory policy, using the bottom line to persuade management to accept her policy.

Analyzed Kitchen Sets

Since management is more likely to accept a data-driven solution, Brenda first analyzed her records over the past three years. She determined that 20 kitchen sets were responsible for 85 percent of the customer orders. These 20 kitchen sets were composed of up to eight features in a variety of styles. Brenda listed each feature and its popular styles in Table 1.

TABLE 1: Top 20 selling kitchen sets composed of up to eight features in a variety of styles.

Floor Tiles	Wallpaper	Light Fixtures	Cabinets
(T1) White textured tile	(W1) Plain ivory paper	(L1) One large rectangular frosted fixture	(C1) Light solid wood
(T2) Ivory textured	(W2) Ivory paper with dark purple pinstripes	(L2) Three small square frosted fixtures	(C2) Dark solid wood cabinets
(T3) White checkered tile with blue trim	(W3) Blue paper with marble texture	(L3) One large oval frosted fixture	(C3) Light wood cabinets with glass doors
(T4) White checkered tile with light yellow trim	(W4) Light yellow paper with marble texture	(L4) Three small frosted globe fixtures	(C4) Dark wood cabinets with glass doors

Countertops	Dishwashers	Sinks	Ranges
(O1) Plain light wood countertops	(D1) White energy-saving dishwasher	(S1) Sink with separate hot and cold water taps	(R1) White electric oven
(O2) Stained light wood countertops	(D2) Ivory energy-saving dishwasher	(S2) Divided sink with separate hot and cold water taps and garbage disposal	(R2) Ivory electric oven
(O3) White lacquer-coated countertops		(S3) Sink with one hot and cold water tap	(R3) White gas oven
(O4) Ivory lacquer-coated countertops		(S4) Divided sink with one hot and cold water tap and garbage disposal	(R4) Ivory gas oven

Brenda then created a table showing the 20 kitchen sets and the particular features composing each set. To simplify the table, she used the codes shown in parentheses above to represent the particular feature and style. The table is given below. For example, kitchen set 1 consists of floor tile T2, wallpaper W2, light fixture L4, cabinet C2, countertop O4, dishwasher D2, sink S2, and range R2. Notice that sets 14 through 20 do not contain dishwashers. Brenda knew she had only a limited amount of warehouse space allocated to the kitchen department. The warehouse could hold 50 square feet of tile and 12 rolls of wallpaper in the inventory bins. The inventory shelves could hold two light fixtures, two cabinets, three countertops, and two sinks. Dishwashers and ranges are similar in size, so Furniture City stored them in similar locations. The warehouse floor could hold a total of four dishwashers and ranges.

TABLE 2: 20 kitchen sets and the particular features composing each set.

	T1	T2	T3	T4	W1	W2	W3	W4	L1	L2	L3	L4	C1	C2	C3	C4	O1	O2	O3	O4	D1	D2	S1	S2	S3	S4	R1	R2	R3	R4
Set 1		1				1						1		1						1		1		1				1		
Set 2		1			1				1							1				1		1				1		1		
Set 3	1						1			1			1				1				1				1				1	
Set 4			1				1				1				1				1		1		1				1			
Set 5				1				1	1				1					1			1			1			1			
Set 6		1				1				1						1				1		1			1					1
Set 7	1						1					1			1			1			1		1				1			
Set 8		1			1						1		1				1					1			1					1
Set 9		1			1					1					1			1				1		1				1		
Set 10	1				1				1				1						1		1					1			1	
Set 11			1		1						1				1		1				1		1						1	
Set 12		1				1			1					1				1				1				1		1		
Set 13				1				1			1				1		1				1			1					1	
Set 14				1				1				1	1						1				1				1			
Set 15			1				1		1				1				1								1				1	
Set 16			1				1					1	1						1					1			1			
Set 17	1							1		1					1				1							1			1	
Set 18		1					1				1			1						1			1					1		
Set 19		1						1				1				1				1				1						1
Set 20		1					1		1				1					1							1					1

Every kitchen set always includes exactly 20 square feet of tile and exactly five rolls of wallpaper. Therefore, 20 square feet of a particular style of tile and five rolls of a particular style of wallpaper are required for the styles to be in stock.

3 Analysis

3.1 Binary Integer Programming Model [Question (a)]

Considering the aforementioned considerations in the Introduction of this report, we formulated the following binary integer programming model to maximize the total number of kitchen sets Furniture City stocks in the local warehouse and, consequently, their number of customer orders.

Inventory Items: Based on Table 1, the inventory items will be referred to as follows:

Floor Tiles (T_i) where i ϵ 1, 2, 3, 4;

Wallpaper (W_i) where i ϵ 1, 2, 3, 4;

Light Fixtures (L_i) where i ϵ 1, 2, 3, 4;

Cabinets (C_i) where i ϵ 1, 2, 3, 4;

Countertops (O_i) where i ϵ 1, 2, 3, 4;

Dishwashers (D_i) where i ϵ 1, 2;

Sinks (S_i) where i ϵ 1, 2, 3, 4;

Ranges (R_i) where i ϵ 1, 2, 3, 4.

Binary Decision Variables: Two types of binary decision variables are used: x_i represents the whether a complete kitchen set is in kept in stock; and y_i represents whether a specific item is in stock.

Complete Kitchen Sets:

Whether each of the 20 kitchen sets (i.e. whose items are detailed in Table 1), x_i where i = 1, 2… 20, are selected for being stocked in the local warehouse by letting x_i equal to:

1	denote the kitchen set is selected.	[indicator]
0	denote the kitchen set not selected.	[indicator]

Stocked Items:

Likewise, additional decision variables include whether or not each of the 30 kitchen sets items (i.e. as detailed in Table 1 and corresponding to the inventory items described above), y_i where i = 1, 2… 30, are selected for being stocked in the local warehouse by letting y_i equal to:

1	denote the kitchen item is selected.	[indicator]
0	denote the kitchen item not selected.	[indicator]

Constraints: The following constraints define the binary integer programming model to find the optimal solution:

1. **Binary Constraint:** The values of x_i and y are binary.

2. **Light Fixtures:** A maximum of two different styles of light fixtures can be in stock.
   $L_1 + L_2 + L_3 + L_4 \leq 2$ [light fixtures]

3. **Cabinets:** A maximum of two different styles of cabinets can be in stock.
   $C_1 + C_2 + C_3 + C_4 \leq 2$ [cabinets]

4. **Countertops:** A maximum of three different styles of countertops can be in stock.
   $O_1 + O_2 + O_3 + O_4 \leq 3$ [countertops]

5. **Sinks:** A maximum of two different styles of sinks can be in stock.
   $S_1 + S_2 + S_3 + S_4 \leq 2$ [sinks]

6. **Dishwashers and Ranges:** A combination of four different styles of dishwashers and ranges can be held in stock.
   $D_1 + D_2 + R_1 + R_2 + R_3 + R_4 \leq 4$ [dishwashers and ranges]

7. **Kitchen Set Tiles:** A maximum of two different styles of tiles can be in stock. This occurs since each set requires 20 feet and the warehouse can only hold 50 square feet of tile.
   $T_1 + T_2 + T_3 + T_4 \leq 2$ [tiles]

8. **Kitchen Set Wallpaper:** A maximum of two different styles of wallpaper can be in stock. This occurs since the warehouse can only hold 12 rolls of wallpaper and each kitchen set requires 5 rolls.
   $W_1 + W_2 + W_3 + W_4 \leq 2$ [sinks]

9. **Kitchen Set:** All items composing a kitchen set must be in stock before selling the kitchen set.
   $8{*}x_1 \leq T_2 + W_2 + L_4 + C_2 + O_4 + D_2 + S_2 + R_2$ [sets]
   $8{*}x_2 \leq T_2 + W_1 + L_1 + C_4 + O_4 + D_2 + S_4 + R_2$ [sets]
   $8{*}x_3 \leq T_1 + W_3 + L_2 + C_1 + O_1 + D_1 + S_3 + R_3$ [sets]
   $8{*}x_4 \leq T_3 + W_3 + L_3 + C_3 + O_3 + D_1 + S_1 + R_1$ [sets]
   $8{*}x_5 \leq T_4 + W_4 + L_1 + C_1 + O_2 + D_1 + S_2 + R_1$ [sets]

   $8{*}x_6 \leq T_2 + W_2 + L_2 + C_4 + O_4 + D_2 + S_3 + R_4$ [sets]
   $8{*}x_7 \leq T_1 + W_3 + L_4 + C_3 + O_2 + D_1 + S_1 + R_1$ [sets]
   $8{*}x_8 \leq T_2 + W_1 + L_3 + C_1 + O_1 + D_2 + S_3 + R_4$ [sets]
   $8{*}x_9 \leq T_2 + W_1 + L_2 + C_3 + O_2 + D_2 + S_2 + R_2$ [sets]
   $8{*}x_{10} \leq T_1 + W_1 + L_1 + C_1 + O_3 + D_1 + S_4 + R_3$ [sets]

   $8{*}x_{11} \leq T_3 + W_1 + L_3 + C_3 + O_1 + D_1 + S_1 + R_3$ [sets]
   $8{*}x_{12} \leq T_2 + W_2 + L_1 + C_2 + O_2 + D_2 + S_4 + R_2$ [sets]

$8*x_{13} \leq T_4 + W_4 + L_3 + C_3 + O_1 + D_1 + S_2 + R_3$ [sets]

$7*x_{14} \leq T_4 + W_4 + L_4 + C_1 + O_3 + S_1 + R_1$ [sets]

$7*x_{15} \leq T_3 + W_3 + L_1 + C_1 + O_1 + S_3 + R_3$ [sets]

$7*x_{16} \leq T_3 + W_3 + L_4 + C_1 + O_3 + S_2 + R_1$ [sets]

$7*x_{17} \leq T_1 + W_4 + L_2 + C_3 + O_3 + S_4 + R_3$ [sets]

$7*x_{18} \leq T_2 + W_3 + L_3 + C_2 + O_4 + S_1 + R_2$ [sets]

$7*x_{19} \leq T_2 + W_4 + L_4 + C_4 + O_4 + S_2 + R_4$ [sets]

$7*x_{20} \leq T_2 + W_3 + L_1 + C_1 + O_2 + S_3 + R_4$ [sets]

Objective Function:

The objective is to maximize the total number of kitchen sets Furniture City stocks in the local warehouse and, consequently, their number of customer orders. Therefore, we seek to:

max Z = $\{ \sum_{i=1}^{20} x_i \}$ [sets]

Optimal Solution of Original Formulation

Question (b): Optimal Solution to Maximize the Number of Kitchen Sets from Part (a).

Using the Gurobi Solver built-in to AMPL, we find that the maximum number of kitchen set to stock is **four**. Although it is not a unique optimal solution, one solution is to stock the items listed Table 3 below. The four complete sets that are stocked with this combination of items are sets {2, 8, 15, and 20}. The constructed binary integer programming model can be found as an AMPL output is found in Figure 1 in Appendix A.

TABLE 3: Different kitchen items to be selected for stock.

T1	0	W1	1	L1	1	C1	1
T2	1	W2	0	L2	0	C2	0
T3	1	W3	1	L3	1	C3	0
T4	0	W4	0	L4	0	C4	1
O1	1	D1	0	S1	0	R1	0
O2	1	D2	1	S2	0	R2	1
O3	0			S3	1	R3	1
O4	1			S4	1	R4	1

3.2 Integer Programming Model for Discontinuing Nursery Set [Question (c)]

Constraint Information for Discontinuing Nursery Sets

Per request of the client, we reformulated our integer programming model, to factor discontinuing nursery sets, to determine the optimal inventory policy for the kitchen department. Constraint number six was changed as follows:

6. **Dishwashers:** Both styles of dishwashers can be accommodated
   {redundant constraint}
   $D_1 + D_2 \leq 2$ [dishwashers]

7. **Ranges:** Three of the four ranges can be stocked.
   $R_1 + R_2 + R_3 + R_4 \leq 3$ [ranges]

3.3 Optimal Solution for Updated Formulation without Nursery Set

Using the Gurobi Solver built-in to AMPL, we find that the maximum number of kitchen set to stock is **five**. Although it is not a unique optimal solution, one solution is to stock the items listed Table 3 below. The five complete sets that are stocked with this combination of items are sets {4, 8, 11, 15, and 20}. The constructed binary integer programming model can be found as an AMPL output is found in Figure 2 in Appendix B.

TABLE 4: Different kitchen items to be selected for stock once the nursery set is discontinued.

T1	0	W1	1	L1	1	C1	1
T2	1	W2	0	L2	0	C2	0
T3	1	W3	1	L3	1	C3	1
T4	0	W4	0	L4	0	C4	0
O1	1	D1	1	S1	1	R1	1
O2	1	D2	1	S2	0	R2	0
O3	1			S3	1	R3	1
O4	0			S4	0	R4	1

3.4 Integer Programming Model for Testing Space Conditions [Question (d)]

Constraint Information for Testing Space Formulation

Per request of the client, we reformulated our integer programming model, to factor in all the additional space provided for the trial. Constraint number six was changed as follows:

1. **Light Fixtures:** A maximum of three different styles of light fixtures can be in stock.
   $L_1 + L_2 + L_3 + L_4 \leq 3$ [light fixtures]

2. **Cabinets:** A maximum of three different styles of cabinets can be in stock.
   $C_1 + C_2 + C_3 + C_4 \leq 3$ [cabinets]

3. **Countertops:** A maximum of four different styles of countertops can be in stock.
   {redundant constraint}
   $O_1 + O_2 + O_3 + O_4 \leq 4$ [countertops]

4. **Sinks:** A maximum of four different styles of sinks can be in stock.
   {redundant constraint}
   $S_1 + S_2 + S_3 + S_4 \leq 4$ [sinks]

3.5 Optimal Solution for Testing Space Formulation

Using the Gurobi Solver built-in to AMPL, we find that the maximum number of kitchen set to stock is **six**. Although it is not a unique optimal solution, one solution is to stock the items listed Table 3 below. The six complete sets that are stocked with this combination of items are sets {3, 8, 9, 10, 18, and 20}. The constructed binary integer programming model can be found as an AMPL output is found in Figure 3 in Appendix B.

TABLE 5: Different kitchen items to be selected for stock once the nursery set is discontinued and all space is available

T1	1	W1	1	L1	1	C1	1
T2	1	W2	0	L2	1	C2	1
T3	0	W3	1	L3	1	C3	1
T4	0	W4	0	L4	0	C4	0
O1	1	D1	1	S1	1	R1	0
O2	1	D2	1	S2	1	R2	1
O3	1			S3	1	R3	1
O4	1			S4	1	R4	1

3.6 Inventory Policy [Question (e)]

If the items were not replenished immediately the number of kitchen sets in stock would decrease dramatically as soon as one item was removed. In this case, it may be better to redefine what is meant by a complete set. If items were not replenished immediately then complete set should only be considered complete if it has dedicated items in stock; items that do not also form part of other sets. With this assumption, the model suggested in this report will not work.

The assumption of immediate replenishment might work if the warehouse uses a Just In Time policy with some type of inventory buffer nearby. Items that are sold can then be immediately replaced by the items in the buffer and the items in the buffer can then be reordered .

4 Appendices

Appendix A. Representation of the Binary Integer Programming Model in AMPL with the Optimal Solution

FIGURE 1: Optimum solution for (a).

kitchen.dat

```
param  rows   :=     20;
param  cols   :=     30;

set    tile                 :=     1      2      3      4;
set    paper                :=     5      6      7      8;
set    light                :=     9      10     11     12;
set    cabinet              :=     13     14     15     16;
set    counter              :=     17     18     19     20;
set    washer_range  :=     21     22     27     28     29     30;
set    sink                 :=     23     24     25     26;

param group [*,*]:
       #      T1     T2     T3                          R2     R3     R4
              1      2      3      …      …      …      28     29     30 :=
       1      0      1      0      …      …      …       1     0      0
       2      …      …      …      …      …      …      …      …      …
       …      …      …      …      …      …      …      …      …      …
       20     0      1      …      …      …      …      0      0      1;
```

kitchen.mod###

```
param  cols;
param  rows;

set row                     := 1..rows;
set col                     := 1..cols;
param group                  {i in row, j in col};

set    tile;
set    paper;
set    light;
set    cabinet;
set    counter;
set    washer_range;
set    sink;


###################
#DECISION VARIABLES
###################
# CHOOSE THE KITCHEN SETS TO STOCK AND THE ASSOCIATED INVENTORY
var kitchen   {row}  binary;
var item             {col}  binary;
```

```
###################
#OBJECTIVE FUNCTION:  MAXIMIZE THE NUMBER OF KITCHEN SETS STOCKED
###################
maximize STOCK: sum{i in row}  kitchen[i];


############
#CONSTRAINTS
############
#ALL ITEMS MUST BE STOCKED FOR A SET TO BE AVAILABLE:
s.t. COMPLETE {i in row}:
              sum{j in col} (item[j]*group[i,j]) >= kitchen[i]*sum{j in col} group[i,j];


#TILES may not take up more than 50 sqr. ft.  Complete sets require 20 sqr ft of tile
s.t. TILE: sum{i in tile} item[i] <= 50/20;

#WALLPAPER may not take be more than 12 rolls.  Complete set require
s.t. PAPER: sum{i in paper} item[i] <= 12/5;

#LIGHTS a maximum of two
s.t. LIGHTS: sum{i in light} item[i] <= 2;

#CABINET a maximum of two
s.t. CABINET: sum{i in cabinet} item[i] <= 2;

#COUNTER a maximum of three
s.t. COUNTER: sum{i in counter} item[i] <= 3;

#SINK a maximum of two
s.t. SINK: sum{i in sink} item[i] <= 2;

#DISHWASHER plus RANGES a maximum of four
s.t. WASHER_RANGE: sum{i in washer_range} item[i] <= 4;
```

kitchen.run###

```
reset;
model kitchen.mod;
data  kitchen.dat;
option solver gurobi;
solve;
display STOCK;
display kitchen;
display item;
```

Results (b)

```
ampl: include kitchen.run;
Gurobi 7.0.0: optimal solution; objective 4
909 simplex iterations
65 branch-and-cut nodes
STOCK = 4

kitchen [*] :=
 1  0
```

```
 2  1
 3  0
 4  0
 5  0
 6  0
 7  0
 8  1
 9  0
10  0
11  0
12  0
13  0
14  0
15  1
16  0
17  0
18  0
19  0
20  1
;

item [*] :=
 1 0    4 0    7 1   10 0   13 1   16 1   19 0   22 1   25 1   28 1
 2 1    5 1    8 0   11 1   14 0   17 1   20 1   23 0   26 1   29 1
 3 1    6 0    9 1   12 0   15 0   18 1   21 0   24 0   27 0   30 1
;
```

Appendix B. Variants of the Binary Integer Programming Model in AMPL with the Optimal Solutions

FIGURE 2: Optimum solution for question (c).

kitchen2.dat

```
param  rows   :=     20;
param  cols   :=     30;

set    tile                 :=     1      2      3      4;
set    paper                :=     5      6      7      8;
set    light                :=     9      10     11     12;
set    cabinet              :=     13     14     15     16;
set    counter              :=     17     18     19     20;
set    washer               :=     21     22;
set    sink                 :=     23     24     25     26;
set    range                :=     27     28     29     30;

param group [*,*]:
       #      T1     T2     T3                          R2     R3     R4
              1      2      3      …      …      …      28     29     30 :=
       1      0      1      0      …      …      …       1     0      0
       2      …      …      …      …      …      …      …      …      …
       …      …      …      …      …      …      …      …      …      …
       20     0      1      …      …      …      …      0      0      1;
```

kitchen2.mod###

```
param  cols;
param  rows;

set row                     := 1..rows;
set col                     := 1..cols;
param group                  {i in row, j in col};

set    tile;
set    paper;
set    light;
set    cabinet;
set    counter;
set    washer;
set range;
set    sink;


###################
#DECISION VARIABLES
###################
# CHOOSE THE KITCHEN SETS TO STOCK AND THE ASSOCIATED INVENTORY
var kitchen   {row}  binary;
var item             {col}  binary;
```

```
###################
#OBJECTIVE FUNCTION:  MAXIMIZE THE NUMBER OF KITCHEN SETS STOCKED
###################
maximize STOCK: sum{i in row}  kitchen[i];


############
#CONSTRAINTS
############
#ALL ITEMS MUST BE STOCKED FOR A SET TO BE AVAILABLE:
s.t. COMPLETE {i in row}:
              sum{j in col} (item[j]*group[i,j]) >= kitchen[i]*sum{j in col} group[i,j];


#TILES may not take up more than 50 sqr. ft.  Complete sets require 20 sqr ft of tile
s.t. TILE: sum{i in tile} item[i] <= 50/20;

#WALLPAPER may not take be more than 12 rolls.  Complete set require
s.t. PAPER: sum{i in paper} item[i] <= 12/5;

#LIGHTS a maximum of two
s.t. LIGHTS: sum{i in light} item[i] <= 2;

#CABINET a maximum of two
s.t. CABINET: sum{i in cabinet} item[i] <= 2;

#COUNTER a maximum of three
s.t. COUNTER: sum{i in counter} item[i] <= 3;

#SINK a maximum of two
s.t. SINK: sum{i in sink} item[i] <= 2;

#DISHWASHER a maximum of two
s.t. WASHER: sum{i in washer} item[i] <= 2;

#RANGES a maximum of three
s.t. RANGE: sum{i in range} item[i] <= 3;
```

kitchen2.run###

```
reset;
model kitchen2.mod;
data  kitchen2.dat;
option solver gurobi;
solve;
display STOCK;
display kitchen;
display item;
```

Results (c)

```
ampl: include kitchen2.run;
Gurobi 7.0.0: optimal solution; objective 5
897 simplex iterations
77 branch-and-cut nodes
STOCK = 5

kitchen [*] :=
```

```
 1  0
 2  0
 3  0
 4  1
 5  0
 6  0
 7  0
 8  1
 9  0
10  0
11  1
12  0
13  0
14  0
15  1
16  0
17  0
18  0
19  0
20  1
;

item [*] :=
 1 0    4 0    7 1   10 0   13 1   16 0   19 1   22 1   25 1   28 0
 2 1    5 1    8 0   11 1   14 0   17 1   20 0   23 1   26 0   29 1
 3 1    6 0    9 1   12 0   15 1   18 1   21 1   24 0   27 1   30 1
```

FIGURE 3: Optimum solution for question (d).

kitchen3.dat

```
param  rows   :=     20;
param  cols   :=     30;

set    tile                :=     1      2      3      4;
set    paper               :=     5      6      7      8;
set    light               :=     9      10     11     12;
set    cabinet             :=     13     14     15     16;
set    counter             :=     17     18     19     20;
set    washer              :=     21     22;
set    sink                :=     23     24     25     26;
set    range               :=     27     28     29     30;

param group [*,*]:
       #      T1     T2     T3                          R2     R3     R4
              1      2      3      …      …      …      28     29     30 :=
       1      0      1      0      …      …      …       1     0      0
       2      …      …      …      …      …      …      …      …      …
       …      …      …      …      …      …      …      …      …      …
       20     0      1      …      …      …      …      0      0      1;
```

kitchen3.dat

```
param  cols;
param  rows;
```

```
set row                     := 1..rows;
set col                     := 1..cols;
param group                  {i in row, j in col};

set    tile;
set    paper;
set    light;
set    cabinet;
set    counter;
set    washer;
set range;
set    sink;


###################
#DECISION VARIABLES
###################
# CHOOSE THE KITCHEN SETS TO STOCK AND THE ASSOCIATED INVENTORY
var kitchen   {row}  binary;
var item             {col}  binary;


###################
#OBJECTIVE FUNCTION:  MAXIMIZE THE NUMBER OF KITCHEN SETS STOCKED
###################
maximize STOCK: sum{i in row}  kitchen[i];


############
#CONSTRAINTS
############
#ALL ITEMS MUST BE STOCKED FOR A SET TO BE AVAILABLE:
s.t. COMPLETE {i in row}:
              sum{j in col} (item[j]*group[i,j]) >= kitchen[i]*sum{j in col} group[i,j];


#TILES may not take up more than 50 sqr. ft.  Complete sets require 20 sqr ft of tile
s.t. TILE: sum{i in tile} item[i] <= 50/20;

#WALLPAPER may not take be more than 12 rolls.  Complete set require
s.t. PAPER: sum{i in paper} item[i] <= 12/5;

#LIGHTS a maximum of two
s.t. LIGHTS: sum{i in light} item[i] <= 3;

#CABINET a maximum of two
s.t. CABINET: sum{i in cabinet} item[i] <= 3;

#COUNTER a maximum of three
s.t. COUNTER: sum{i in counter} item[i] <= 4;

#SINK a maximum of two
s.t. SINK: sum{i in sink} item[i] <= 4;

#DISHWASHER  a  maximum of two
s.t. WASHER: sum{i in washer} item[i] <= 2;

#RANGES a maximum of three
```

```
s.t. RANGE: sum{i in range} item[i] <= 3;
```

kitchen3.run###

```
reset;
model kitchen2.mod;
data  kitchen2.dat;
option solver gurobi;
solve;
display STOCK;
display kitchen;
display item;
```

Results (d)

```
ampl: include kitchen3.run;
Gurobi 7.0.0: optimal solution; objective 6
403 simplex iterations
36 branch-and-cut nodes
STOCK = 6

kitchen [*] :=
 1  0
 2  0
 3  1
 4  0
 5  0
 6  0
 7  0
 8  1
 9  1
10  1
11  0
12  0
13  0
14  0
15  0
16  0
17  0
18  1
19  0
20  1
;

item [*] :=
 1 1    4 0    7 1   10 1   13 1   16 0   19 1   22 1   25 1   28 1
 2 1    5 1    8 0   11 1   14 1   17 1   20 1   23 1   26 1   29 1
 3 0    6 0    9 1   12 0   15 1   18 1   21 1   24 1   27 0   30 1
```